VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

VYUŽITÍ ČASOVÝCH ZNAČEK JAZYKA JAVASCRIPT PRO IDENTIFIKACI POČÍTAČE

COMPUTER IDENTIFICATION USING JAVASCRIPT TIMESTAMPS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MICHAL JIREŠ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. LIBOR POLČÁK, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Jireš Michal**
Program: Informační technologie
Název: **Využití časových značek jazyka JavaScript pro identifikaci počítače**
Computer Identification Using JavaScript Timestamps
Kategorie: Web

Zadání:

1. Seznamte se s možnostmi identifikace počítačů v síťovém prostředí na základě odchylky jeho vnitřních hodin a programem PC Fingerprinter.
2. Seznamte se s metodami modifikace času implementovanými v rozšířeních JS Restrictor a Chrome Zero.
3. Navrhněte program v jazyce JavaScript, který bude počítat odchylku v měření času lokálního počítače vůči referenčnímu serveru.
4. Implementujte návrh jako rozšíření testovací stránky projektu JavaScript Restrictor.
5. Otestujte implementaci a ověřte chování rozšíření JavaScript Restrictor.
6. Zhodnoťte výsledky projektu a jeho dopady.

Literatura:

- Tadayoshi Kohno, Andre Broido, K.C. Claffy: Remote Physical Device Fingerprinting. IEEE Transactions on Dependable and Secure Computing, s. 93-108, 2005.
- Jakub Jirásek: Využití časových informací pro identifikaci počítače, diplomová práce, Brno, FIT VUT v Brně, 2012.
- Libor Polčák a Barbora Franková: Clock-Skew-Based Computer Identification: Traps and Pitfalls. Journal of Universal Computer Science. 2015, roč. 21, č. 9, s. 1210-1233. ISSN 0948-6968.
- Schwarzm Michael aj. JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks. Network and Distributed Systems Security Symposium 2018. San Diego, CA, USA.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání včetně vypracování technické zprávy.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Polčák Libor, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 14. května 2020
Datum schválení: 21. října 2019

Abstrakt

Tato práce se zabývá identifikací počítačů na základě posunu jejich vnitřních hodin. Tento posun bude zjišťován pomocí časových značek JavaScriptu. Cílem práce je vytvořit webovou stránku, která se pomocí výpočtu posunu bude snažit identifikovat počítač. Stránka bude také použita pro testování rozšíření do prohlížeče JavaScript Restrictor. Stránka bude zkoušet identifikovat počítač při použití různých úrovní zabezpečení a tím testovat jejich efektivnost.

Abstract

This thesis deals with remote computer identification based on its internal clock skew. This clock skew will be determined using JavaScript timestamps. The goal of this thesis is to create web page, that will use clock skew to identify computer. This web page will also be used to test time distortion capabilities of browser extension JavaScript Restrictor. Web page will try to identify computer using different security presents thus testing their efficiency.

Klíčová slova

Identifikace počítače, JavaScript, JavaScript Restrictor, JavaScript Zero, vnitřní hodiny, odchylka hodin, časové značky, posun hodin.

Keywords

Computer identification, JavaScript, JavaScript Restrictor, JavaScript Zero, internal clock, clock offset, timestamps, clock skew.

Citace

JIREŠ, Michal. *Využití časových značek jazyka JavaScript pro identifikaci počítače*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Libor Polčák, Ph.D.

Využití časových značek jazyka JavaScript pro identifikaci počítače

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Libora Polčáka a uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Jireš
4. června 2020

Poděkování

Rád bych poděkoval vedoucímu práce, panu Ing. Liborovi Polčákovi, Ph.D., za cenné rady a poznatky.

Obsah

Kapitola 1

Úvod

V dnešní moderní společnosti je nepředstavitelné nemít zařízení s přístupem k internetu. Používáním těchto zařízení uživatel generuje data, ať už to jsou jeho příspěvky na sociální sítě nebo jenom historie surfování na webu. Stejně, jak se zvyšuje počet těchto dat, tak se zvyšuje počet pokusů tyto data nelegálně získat. Čím dál tím více je patrná stránka soukromí uživatele. Webové stránky si ukládají každé vyhledávání, jaké stránky byly navštíveny, které služby na internetu byly využity.

Začala proto vznikat bezpečnostní rozšíření do prohlížečů, snažící se zachovat bezpečnost a soukromí jejich uživatelů. Jedno takové rozšíření vzniklo jako diplomová práce Ing. Zbyňka Červinky [2]. Rozšíření s názvem JavaScript Restrictor funguje na principu zapouzdřování javascriptových funkcí, kde upravuje jejich funkčnost. Zamezuje webové stránce přímý přístup k těmto funkcím.

Cílem této práce je vybudovat webovou stránku, která bude použita k identifikaci počítače. Tato stránka bude také použita pro testování rozšíření JavaScript Restrictor. Konkrétně k testování manipulace s časem. Testy budou probíhat na základě pokusu identifikovat počítač pomocí posunu jeho vnitřních hodin. Identifikace pomocí posunu vnitřních hodin byla původně představena a popsána v práci „Remote physical device fingerprinting“ [7]. Funguje na principu nedokonalosti vnitřních hodin počítače a možnosti zjištění jedinečného posunu těchto hodin. Pro zjištění posunu je zapotřebí znát přesný čas, proto je tato technika vhodným kandidátem na testování výše zmíněného rozšíření.

Kapitola 2

Identifikace počítače

Každý počítač má vnitřní hodiny. Signál hodin produkuje oscilátor, který kmitá na určité frekvenci. Výrobní proces těchto oscilátorů není dokonalý a proto mají určité vady. Tyto vady způsobí, že oscilátory kmitají na odlišných frekvencích. Každé hodiny mají tento posun frekvence jedinečný, proto jej lze použít k identifikaci počítače.

Abychom tento posun zjistili potřebujeme znát hodiny zkoumaného zařízení. Tyto hodiny se dají získat více způsoby, které můžeme dělit na aktivní a pasivní.

2.1 Pasivní a aktivní zjišťování časového razítka

U pasivního zjišťování časového razítka nekomunikujeme se zkoumaným počítačem. Pouze prohlížíme hlavičky paketů, které zkoumaný počítač posílá. Předpokládáme tedy, že zkoumaný počítač s někým komunikuje. Ne všechny hlavičky v sobě nesou časovou značku počítače. Proto odchytáváme pakety jen určitých protokolů, například TCP. Výhodou pasivního zjišťování je, že zkoumaný počítač nezjistí, že je zkoumán.

U aktivního zjišťování musíme navázat spojení se zkoumaným počítačem. Toto spojení je viditelné ze strany zkoumaného počítače. Pro získávání časového razítka lze použít například ICMP nebo HTTP.

2.2 Způsoby zjišťování časové značky

V našem případě budeme časové značky zkoumaného počítače získávat pomocí JavaScriptové funkce *performance.now*. Časové značky se porovnají s časy referenčních serverů více popsaných v kapitole 4.1.

Obecně existuje více způsobů získávání časových značek. Většina z těchto způsobů, však pracuje na principu, že měřící počítač naslouchá a nebo vyvolává komunikaci, z které získává časové značky. Tyto časové značky měřeného počítače pak porovnává s aktuálním časem měřícího počítače. Na základě tohoto porovnání pak počítá posun vnitřních hodin měřeného počítače vůči měřícímu počítači.

V případě této stránky bude měřený počítač posílat dotazy serveru, který mu bude odpovídat aktuální čas. Pro stránku pak získaný čas ze serveru hraje roli aktuálního času měřícího počítače a čas měřeného počítače bude získávat pomocí funkce *performance.now*. Bude tedy počítat posun vnitřních hodin počítače, na kterém stránka běží, vůči serveru, který posílal aktuální čas jako odpověď.

2.3 Odchylka a posun vnitřních hodin

V následující sekci použijeme terminologii zavedenou v [7]. Mějme hodiny C, které určují čas uběhlý od nějakého původního času $i[C]$. Hodnota $r[C]$ je pak nejmenší možný přírůstek času. Frekvence hodin $Hz[C]$, tedy odpovídá inverzi přírůstku $r[C]$.

$$Hz[C] = \frac{1}{r[C]} \tag{2.1}$$

Nechť $R[C](t)$ reprezentuje čas udaný hodinami C v čase t, kde čas t je reálný čas definovaný normami. Odchylka hodin C, $off[C](t)$, bude rozdílem času podaném hodinami C a reálného času t. Odchylka má jednotku jako čas, udává se typicky v milisekundách.

$$\forall t \geq i[C], \quad off[C](t) = R[C](t) - t \tag{2.2}$$

Posun vnitřních hodin, $s[C]$, je první derivací odchylky hodin podle času, kde předpokládáme, že $R[C]$ je spojitá funkce v čase t. Jednotkou posunu jsou mikrosekundy za sekundu ($\mu s/s$) nebo počet částí v milionu (ppm).

Posun vnitřních hodin k identifikaci počítače je možné, protože se v rámci jednoho počítače tento posun pohybuje v rozmezí ± 1 - 2 ppm, zatímco u různých počítačů je rozdíl až 50 ppm.

2.4 Určení posunu vnitřních hodin

V následující sekci použijeme terminologii zavedenou v [7]. Předpokládejme, že již máme získanou množinu informací, obsahující časová razítka. Tato množina tvoří posloupnost, seřazenou podle času přijetí, kterou nazýváme T. Nechť t_i je čas v sekundách, ve kterém jsme obdrželi i-tou časovou informaci a T_i nechť je ona časová značka obsažena v i-té informaci.

Proměnná x_i, určuje čas v sekundách, uběhnutých od počátku měření po získání i-té informace.

$$x_i = t_i - t_1 \tag{2.3}$$

Rozdíl hodnot časového razítka i-té informace T_i a první informace T_1 určuje proměnná v_i.

$$v_i = T_i - T_1 \tag{2.4}$$

Jelikož, pro další postup potřebujeme, aby proměnné x_i a v_i měly stejný rozměr (x_i je v sekundách a v_i je závislá na frekvenci hodin C). Musíme v_i převést na sekundy.

$$w_i = \frac{v_i}{Hz[C]} \tag{2.5}$$

Potom odchylka $off[C]$ i-té informace je rozdíl mezi časem udaným časovým razítkem a časem reálným.

$$y_i = w_i - x_i \tag{2.6}$$

Výsledkem výše zmíněných výpočtů je posloupnost odchylek O_T, náležící k posloupnosti T. Takovou posloupnost, lze vidět na obrázku 2.1 jako modré body.

$$O_T = \{(x_i, y_i) : i \in \{1, ..., |T|\}\} \tag{2.7}$$

Pro určení posunu vnitřních hodin existuje více metod, například metoda založená na lineárním programování, kterou zde použijeme, nebo metoda nejmenších čtverců. Metoda založená na lineárním programování, vychází z faktu, že je posun dvou porovnávaných hodin s časem konstantní. Zpoždění kvůli síťové komunikaci se může měnit, ale při dostatečně dlouhém měření na výsledek nemá vliv.

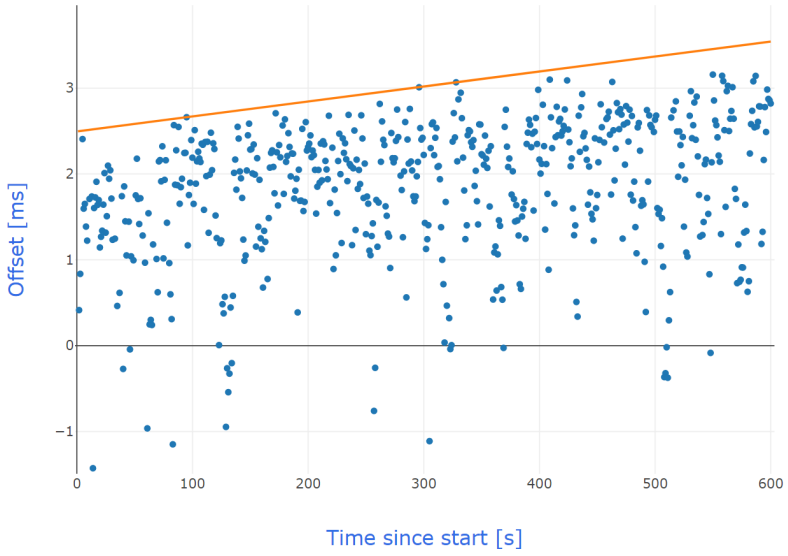
Pro zjištění rychlosti změny odchylek, proložíme body přímkou a změříme úhel, který přímka svírá s osou x (v našem případě je osa x čas měření). Použitá přímka je lineární funkce, shora ohraničující množinu všech odchylek. Příklad takovéto přímky lze vidět na obrázku 2.1 jako oranžovou čáru.

$$\forall i \in \{1, ..., |T|\}, \quad \alpha \cdot x_i + \beta \geq y_i \tag{2.8}$$

Výsledkem lineárního programování je taková funkce, která má nejmenší vertikální vzdálenost od všech bodů posloupnosti O_T.

$$min\Big(\sum_{i=1}^{|T|}(\alpha \cdot x_i + \beta - y_i)\Big) \tag{2.9}$$

Hledaný posun vnitřních hodin je pak hodnota α, určující směrnici přímky.



Obrázek 2.1: Graf zobrazující posloupnost odchylek a přímku reprezentující horní ohraničení této posloupnosti.

Kapitola 3

Modifikace času

V nynější době, kdy každým dnem přibývá počet škodlivých stránek, začínají lidé používat bezpečnostní rozšíření do svých prohlížečů. Tato rozšíření zajišťují bezpečnost a soukromí. Většina těchto rozšíření funguje na principu, omezování funkčnosti JavaScriptu. Mezi tato omezení ovšem patří i omezení na volání časových funkcí.

JavaScript nabízí dvě základní možnosti pro zjišťování času. Objekt Date, který udává čas v milisekundách, nebo metodu *performance.now*, která patří pod objekt Performance a udává čas s přesností na tisíciny milisekund. Kvůli bezpečnosti, bezpečnostní rozšíření tyto funkce obalují a snižují přesnost vráceného času.

3.1 Modifikace času u Chrome zero

Chrome zero je bezpečnostní rozšíření pro prohlížeč Google Chrome. Toto rozšíření bylo vytvořeno Michaelem Schwarzem, Moritzem Lippem a Danielem Grussem jako implementace JavaScriptu Zero [10].

Poskytuje čtyři úrovně zabezpečení. Tabulka 3.1 ukazuje, jak se bude pracovat s funkcí *performance.now* při různých úrovních zabezpečení.

	Off	Low	Medium	High	Paranoid
Accurate Timing	-	Ask	Low-resolution timestamp	Fuzzy time	Disable

Tabulka 3.1: Tabulka práce s časem korespondující s úrovní zabezpečení. Převzato z [10].

- **Ask** - Ptá se uživatele jak chce zareagovat.
- **Low-resolution timestamp** - Zaokrouhluje, popsané v sekci 3.1.1.
- **Fuzzy time** - Zaokrouhluje a přidává šum, popsané v sekci 3.1.2.
- **Disable** - Zakázání volání funkce.

3.1.1 Low-resolution timestamp

Při použití tohoto zkreslení se čas získaný z funkce *performance.now* zaokrouhluje na stovky milisekund. Implementace zobrazena na obrázku 3.1. Stejné zkreslení se používá například v prohlížeči Tor.

```
"window.performance.now": {
  "action": "modify",
  "return": "Math.floor(window.performance.now() / 100.0) * 100.0"
},
```

Obrázek 3.1: Implementace Low-resolution timestamp v Chrome zero.

3.1.2 Fuzzy time

Tato metoda počítá zkreslený čas s pomocí zavedení šumu. Tento šum zabraňuje přesným měřením času, ale garantuje, že čas získaný touto funkcí stále roste. Obrázek 3.2 zobrazuje implementaci. Hodnota šumu se pohybuje v intervalu $\langle 0, 999 \rangle$ (pro čas v mikrosekundách), získává se pomocí matematické funkce random. Tato hodnota se následně používá pro výpočet zkresleného času. Šum způsobí náhodný posun času, tento posun je menší než 1 ms. Funkce vrací čas v milisekundách s desetinnou částí určující mikrosekundy.

```
window.performance.now = function() {
    window._policy = true;
    var now = Math.floor(wpn.call(window.performance) * 1000);
    var fuzz = Math.floor(Math.random() * 1000);
    var t = now - now % fuzz; // now is the current time
    if(t > last) last = t;
    return last / 1000.0;
};
```

Obrázek 3.2: Implementace Fuzzy time v Chrome zero.

3.2 Modifikace času u JavaScript Restrictor

JavaScript Restrictor je rozšíření, které vytvořil Ing. Zbyněk Červinka jako svou diplomovou práci [2] a následně ho vylepšil Ing. Martin Timko jako součást své diplomové práce [11].

JavaScript Restrictor umožňuje různé míry zkreslení času. Míra zkreslení je vybrána na základě přednastavené úrovně zabezpečení. Tyto úrovně jsou očíslovány od 0 do 3. Kde úroveň 0 je žádné zabezpečení a úroveň 3 je maximální zabezpečení. Další možnost je vybrání úrovně Custom, ta umožňuje si uživateli určit, které bezpečnostní opatření chce a nebo nechce aby se projevovaly.

3.2.1 Objekt Performance a objekt Date

Funkce *performance.now*, která normálně vrací čas s přesností na mikrosekundy, se upravuje na každé úrovni jinak. Objekt Date, který normálně vrací čas s přesností na milisekundy, se upravuje stejně jako *performance.now*. Úpravy na základě úrovní vypadají takto:

- **Úroveň 0** - Nezaokrouhluje.
- **Úroveň 1** - Zaokrouhluje na desítky milisekund.
- **Úroveň 2** - Zaokrouhluje na stovky milisekund.
- **Úroveň 3** - Zaokrouhluje na celé sekundy.

Kapitola 4

Návrh

Výstupem této práce by měla být webová stránka, která bude zkoumat čas počítače a porovnávat ho s referenčním časem serveru. Na základě těchto porovnávání bude stránka identifikovat počítač. Tato stránka bude také použita pro testování efektivnosti zkreslování času pro pokus o zmaření identifikace. Pro zkreslování času se využije rozšíření JavaScript Restrictor. Stránka bude rozdělena na dvě části, serverovou a klientskou. Serverová část bude API, které po dotázání odpoví se svým aktuálním časem. Klientská část pro nás bude samotný javascriptový skript, který bude počítat posun hodin a výsledky ukládat. Po dohodě s vedoucím práce, jsme se shodli, že data budou ukládána lokálně. Centrální ukládání není účelem této práce, pro tuto stránku je to zbytečné.

4.1 Serverová část

Od serveru se očekává, že po dotázání klienta, odešle jako odpověď svůj aktuální čas. Jsou zde dvě možnosti, implementovat vlastní řešení na školním serveru Eva a nebo vyhledat již existující API. Po dohodě s vedoucím práce jsme se rozhodli použít obě možnosti.

Na internetu jsem našel tři použitelné API, které po dotázání odpoví se svým aktuálním časem ve formátu JSON. Nevýhodou těchto stránek a zároveň důvodem proč budu implementovat i vlastní řešení je skutečnost, že kterékoliv z těchto API může během psaní této práce přestat fungovat.

4.1.1 Formát JSON

JSON (JavaScript Object Notation) je formát pro výměnu dat, definovaný standardem RFC 8259 [1]. Používá se hlavně kvůli své čitelnosti a nezávislosti na použitém programovacím jazyku.

Je založen na dvou strukturách:

- **Kolekce párů název/hodnota** - V jazycích implementována jako objekt, struktura, hash tabulka, nebo asociativní pole.
- **Seřazený seznam hodnot** - Ve většině jazyků implementován jako pole, vektor, list nebo posloupnost.

Tyto struktury jsou zde realizovány s využitím konstrukcí *objekt*, *pole*, *hodnota*. Jednotlivé struktury mohou být vzájemně vnořovány.

Kde *objekt* je neuspořádaná množina párů název, hodnota, započata znakem „{“ a zakončena znakem „}“. Název je následován znakem „:“ a hodnotou. Páry název, hodnota jsou odděleny znakem „,“.

Pole je seřazenou kolekcí hodnot. Začíná znakem „[“ a končí znakem „]“. Obsahuje *hodnoty* vzájemně oddělené znakem „,“.

Hodnotou rozumíme *objekt*, *pole*, číslo, true, false, null a nebo řetězec uzavřený dvojitými závorkami.

Příklady je možné vidět na obrázcích 4.1, 4.2 a 4.3 níže. Je na nich vidět použití objektu obsahujícího páry název, hodnota.

4.1.2 První API - worldtimeapi.org

World Time API[1] je API, které po dotázání, vrátí JSON obsahující informace o zadané lokaci. Na jakou lokaci se tážeme je pro fungování stránky nevýznamné. Odpověď je zobrazena na obrázku 4.1


```
{
  "week_number": 2,
  "utc_offset": "+01:00",
  "utc_datetime": "2020-01-11T10:33:44.099082+00:00",
  "unixtime": 1578738824,
  "timezone": "Europe/Prague",
  "raw_offset": 3600,
  "dst_until": null,
  "dst_offset": 0,
  "dst_from": null,
  "dst": false,
  "day_of_year": 11,
  "day_of_week": 6,
  "datetime": "2020-01-11T11:33:44.099082+00:00",
  "abbreviation": "CET"
}
```


Obrázek 4.1: Odpověď World Time API na dotaz s městem Prague.

Obsahuje spoustu pro tuto práci nepotřebných informací, pro nás jsou důležité pouze položky týkající se času. A to *utc_datetime*, *unixtime* a *datetime*. *Unixtime* pro nás není zajímavý z důvodu jeho nízké rezoluce jedné sekundy. Položka *utc_datetime* obsahuje koordinovaný světový čas, pro nás o hodinu posunutý. Použijeme tedy *datetime* obsahující aktuální čas našeho časového pásma ve formátu ISO 8601. S tím, že čas udává s přesností na mikrosekundy.

4.1.3 Druhé API - jsontest.com

JSON Test[2], jak již jméno napovídá, je testovací platforma používající JSON. Pro nás je důležitá pouze jejich služba Date & Time, která vrací JSON zobrazený na obrázku 4.2.

Pro nás je nejdůležitější položka odpovědi *milliseconds_since_epoch*, která udává počet milisekund od Unixové epochy, neboli od 1. ledna 1970 00:00:00 UTC.

[1] http://www.worldtimeapi.org/
[2] https://www.jsontest.com/

```
{
  "date": "01-11-2020",
  "milliseconds_since_epoch": 1578738133438,
  "time": "10:22:13 AM"
}
```


Obrázek 4.2: Odpověď JSON Test po volání služby time.

4.1.4 Třetí API - worldclockapi.com

World Clock API[3] je API, které po dotázání na určitou časovou zónu vrátí čas této časové zóny. JSON odpovědi je zobrazen na obrázku 4.3.


```
{
  "$id": "1",
  "currentDateTime": "2020-01-11T11:31+01:00",
  "utcoffset": "01:00:00",
  "isDayLightSavingsTime": false,
  "dayOfTheWeek": "Saturday",
  "timeZoneName": "Central Europe Standard Time",
  "currentFileTime": 132232158859809650,
  "ordinalDate": "2020-11",
  "serviceResponse": null
}
```


Obrázek 4.3: Odpověď World Clock API na dotaz na časovou zónu CET.

Odpověď obsahuje více časových údajů. Kde *currentDateTime* je čas ve formátu ISO 8601, stejně jako u **World Time API** popsané v kapitole 4.1.2. Na rozdíl od **World Time API**, které do odpovědi dává čas s přesností na mikrosekundy, zde je čas s přesností pouze na sekundy. V odpovědi se, ale také nachází údaj *currentFileTime*, obsahující čas ve formátu Windows NT time format. Tento formát udává počet 100 nanosekundových intervalů od 1. ledna 1601 00:00:00 UTC. Časový údaj je tedy s přesností na desetinu mikrosekundy.

4.1.5 Vlastní řešení

Vlastním řešením se bere napsání PHP skriptu, který bude umístěn na školní server. Jediným úkolem tohoto skriptu, bude zpracování požadavků od klienta, vložení aktuálního serverového času do JSONu a následně odeslání odpovědi klientovi.

Pro získání času v PHP lze použít funkci *microtime*, funkce podporuje i starší verze PHP a vrací čas v mikrosekundách.

Další možností je nově přidaná funkce do PHP verze 7.3.0 *hrtime*, vracející čas z monotónních hodin určující počet nanosekund od neurčitého času. Tyto hodiny nejsou závislé na systémových hodinách.

[3] http://www.worldtimeapi.org/

4.2 Klientská část

Klientská část bude mít za úkol získávat časové značky z vybraných serverů, stejně jako časovou značku z počítače, na kterém JavaScript poběží. Následně podle těchto časů spočítá posun hodin, postupem popsaným v kapitole 2.4.

Skript bude v pravidelných intervalech posílat žádosti serverům a čekat na odpověď. Po obdržení odpovědi si určí svůj lokální čas pomocí funkce *performance.now*. Z těchto dvou časů vypočítá posun hodin. V našem případě se bude x_i z rovnice 2.3 počítat pomocí hodin získaných ze serverů a v_i z rovnice 2.4 pomocí lokálních hodin počítače. Stránka bude počítat pro každý server svůj posun hodin zvlášť. Po dostatečném počtu provedení výpočtů, až se ustálí výsledný posun, zobrazí vypočítaný posun uživateli a následně ho lokálně uloží.

Pro ukládání výsledků je možné použít *cookies*, ty však nejsou vhodné z důvodů nutnosti přenášet je na server. Proto bude výhodnější použití *localStorage*. Jak již lze odvodit z názvu, *localStorage* ukládá data pouze lokálně, není tedy nutné je přenášet na server. Tyto data mají také neomezenou dobu platnosti.

Kapitola 5

Implementace

Praktickým výstupem této práce bude stránka, která bude měřit posun vnitřních hodin počítače. Stránka bude získávat časové informace ze serverů a porovnávat je s časem počítače, na kterém je stránka spuštěna. Časové informace ze serverů získává jednou za sekundu pomocí objektu XMLHttpRequest. Tyto časové informace jsou následně použity pro výpočet posunu hodin a tvorbu grafu. Je počítáno více posunů, každý vůči jednomu ze serverů. Osa y grafu reprezentuje offset v milisekundách a osa x reprezentuje čas v sekundách od počátku měření. Do grafu se vykreslí všechny body všech serverů a jejich přímky určující ohraničení bodů. Bod reprezentuje jednu získanou časovou informaci ze serveru, souřadnice x bodu určuje čas získaný ze serveru určující čas od začátku měření. Souřadnice y bodu je vypočítaný offset časových informací serveru a klienta.

5.1 Získávání časových informací

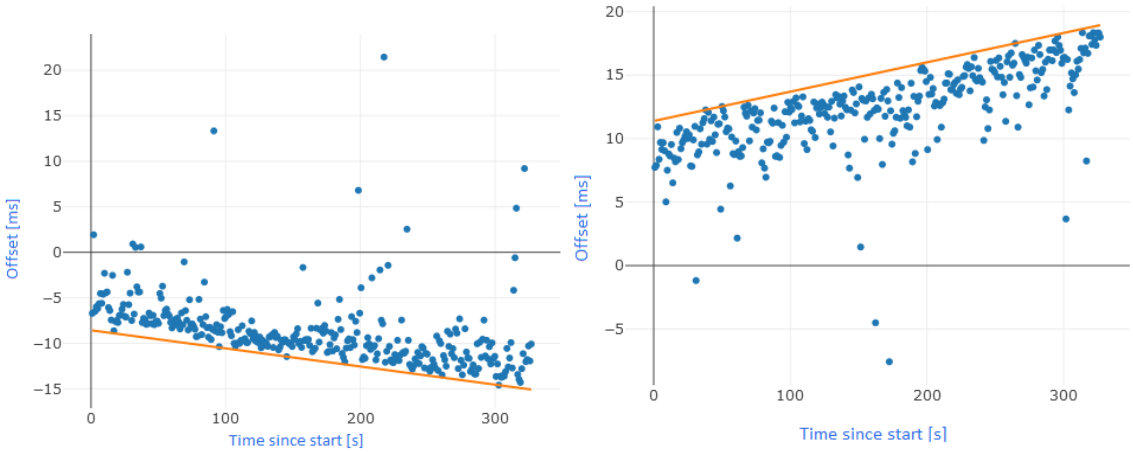
Pro zajištění pravidelnosti je použit javascriptový interval. Tento interval pravidelně volá hlavní funkci CallThemAll. Ta poté volá funkce, které získají časové údaje ze svých serverů. Každý server má svou vlastní funkci.

Tyto funkce pro získávání časových údajů ze serverů, vytváří žádost, kterou odešlou na příslušný server a následně čekají na odpověď. Server může odpovědět očekávanou odpovědí a nebo chybou. Pokud server odpoví chybou, nastaví se chybový příznak daného serveru což zapříčiní, že se jeho funkce pro získávání časových údajů nebude znovu volat. Aby se tato funkce znovu volala, je zapotřebí akce uživatele. Pokud server odpoví očekávanou odpovědí, v našem případě je očekávaná odpověď JSON obsahující časovou informaci, analyzuje se tato odpověď a vybere se z ní potřebný časový údaj. Tato časová informace se převede na sekundy a odečte se od ní čas získaný na začátku měření. Tímto docílíme, že časová informace bude desetinné číslo, které uvádí počet sekund od počátku měření jako svou celou část a k tomu desetinnou část reprezentující zlomky sekundy. Desetinná část se u každého serveru liší, například JSONTest, který posílá čas s přesností na milisekundy, má desetinnou část na 3 desetinná místa. Časová informace získaná z WorldClock má desetinnou část na 7 desetinných míst. Server Eva pak posílá časovou informaci, která po převodu má desetinnou část na 9 desetinných míst.

Časové údaje klienta se získají pomocí javascriptové funkce *performance.now*. Tato funkce vrací čas s přesností na mikrosekundy. Po převodu na desetinné číslo sekund bude mít tedy 6 desetinných míst. Časový údaj z klienta se získá až potom co klient obdrží odpověď ze serveru obsahující časovou informaci serveru.

5.2 Výpočet offsetu

Pro výpočet posunu potřebujeme vypočítat offset klienta vůči serveru. Offset získáme odečtením časových hodnot od sebe. Při tomto odčítání záleží na pořadí. Pokud budeme odečítat časový údaj klienta od časového údaje serveru, počítáme posun hodin serveru vůči klientu. Cílem této práce je počítat posun klienta vůči serveru. Proto musíme odečíst časový údaj serveru od klienta. To ale způsobí to, že navrhovaný postup v kapitole 2.4 musíme pozměnit. Jelikož v našem případě bereme jako reálný čas, čas získaný ze serveru, tedy čas získaný se zpožděním. Pokud narazíme na časovou informaci s výrazně jiným zpožděním, než ostatní, bude ve výsledném grafu zobrazen jako číslo vyšší než průměr. Navrhovaný postup předpokládá, že časová informace s výrazně jiným zpožděním je v grafu číslo nižší než průměr. Proto nedává smysl zjišťovat posun hodin pomocí přímky shora ohraničující množinu všech bodů. Přímka musí ohraničovat body zdola. Obrázek 5.1 ukazuje rozdíl těchto dvou způsobů počítání. Pro oba zobrazené grafy platí, že osa y reprezentuje offset v milisekundách a osa x reprezentuje reálný čas v sekundách uběhlý od počátku měření. Modré body reprezentují časové informace přijaté v čase a jejich vypočítaný offset. Oranžové přímky reprezentují dolní nebo horní ohraničení množiny všech bodů. Pokud by výpočet probíhal stejně jako je vyobrazen v kapitole 2.4, vypadal by výsledný graf jako graf vpravo. Graf vygenerovaný výslednou stránkou bude vypadat jako graf vlevo.



Obrázek 5.1: Rozdíl počítání posunu klienta vůči serveru (vlevo) a serveru vůči klientu (vpravo).

V kódu stránky je to řešeno využitím faktu, že posun hodin serveru vůči klientu a posun hodin klienta vůči serveru jsou navzájem inverzní hodnoty. Skript počítá posun hodin serveru vůči klientu. Offset počítá odečtením klienta od serveru. Pouze pak při prezentaci, výsledky vynásobí číslem -1.

5.3 Výpočet posunu

Vnitřní posun hodin se počítá z množiny vypočítaných offsetů a jejich času přijetí od začátku měření. Pro samotný výpočet posunu jsem použil již existující funkce z programu pcf [6]. Tento program vznikl jako výsledek diplomové práce Jakuba Jiráska. Program pcf byl původně napsán v jazyce C, následně byl však přepsán do jazyka C++. Z tohoto programu jsem použil funkce pro počítání konvexního obalu bodů a funkci pro hledání přímky

s nejmenší vzdáleností od bodů a následné vypočítání samotného posunu. Tyto funkce jsem s mírnými úpravami přepsal do jazyka JavaScript.

Funkce pro počítání konvexního obalu bodů používá algoritmus popsaný Ronaldem L. Grahamem [4]. Tato funkce vrátí body, které patří do konvexního obalu. Těmito body jsou následně vedené přímky. Tyto přímky jsou ve tvaru.

$$y = alpha * x + beta$$

Porovnává se vzdálenost těchto přímek od všech bodů. Výsledná přímka je taková, která má tuto vzdálenost nejmenší. Hodnota alpha této přímky pak udává hledaný posun. Tento získaný posun má jednotku *ms/s*, jelikož se počítá s offsetem v milisekundách a časem měření v sekundách, proto ho musíme vynásobit jedním tisícem, abychom dostali posun v požadované jednotce *ppm* tedy $\mu s/s$.

5.3.1 Frekvence volání výpočtu posunu

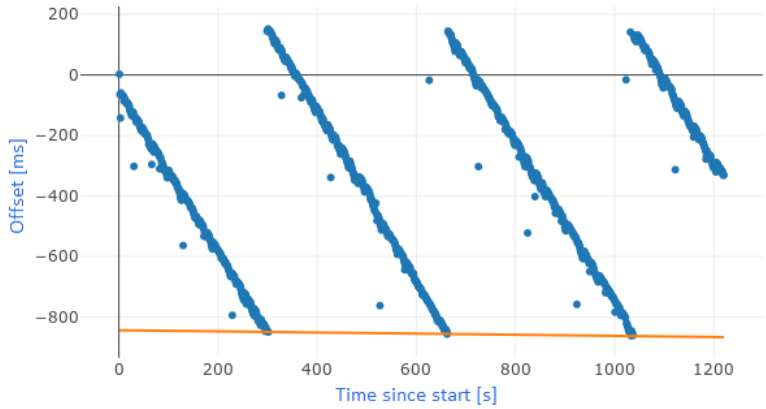
Jelikož výpočet pro vysoký počet bodů může být časově náročný, nevolá se pro každou nově přijatou časovou informaci. Také by to ani nebylo užitečné, jelikož je malá šance, že tato nově přijatá časová informace změní podobu vypočítaného posunu. Funkce pro výpočet se volá každých **x** sekund, resp. každých **x** nově přijatých časových informací.

Skript hodnotu **x** mění na základě přesnosti získaného času z funkce *performance.now*. Počítá s mírami zaokrouhlení zobrazenými v kapitole 3.2.1. Pokud je vyšší míra zaokrouhlení, čas mezi výpočty se musí prodloužit, protože předpokládáme, že dostaneme méně přesné časové informace. Tedy musíme nepřesné informace vykompenzovat delší dobou měření.

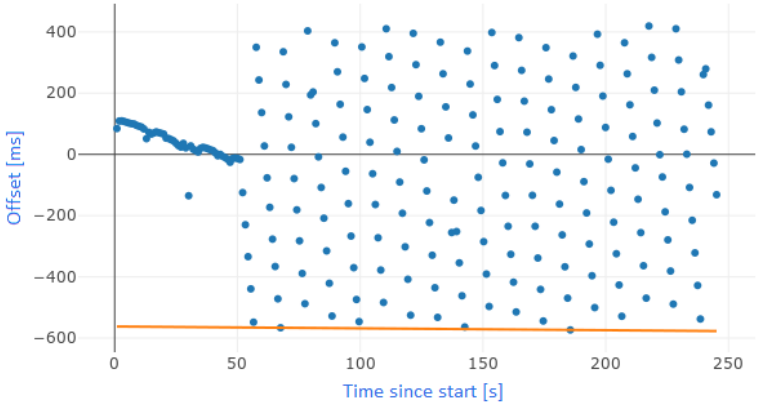
Tyto hodnoty byly zjištěny měřením. Základní hodnotu, když čas získaný z *performance.now* není změněn, nastaví na 300. To znamená, že se každých 300 sekund, resp. 300 přijatých časových informací zavolá výpočet posunu. Hodnotu mění, když je čas získaný z *performance.now* zaokrouhlován na desítky milisekund a to na 400. Pokud je čas zaokrouhlován na stovky milisekund použije hodnotu 500. Při zaokrouhlení na sekundy nebo vyšší použije hodnotu 900. Při zaokrouhlení na sekundy se v grafu body sdružují do shluků připomínající úsečku viz. obrázek 5.2. V tomto případě to způsobí, že body, které by mohly patřit do dolního ohraničení dostaneme pouze jednou za cca 350 sekund. Tato doba se může lišit u jiných počítačů i měření. Kdyby byla použita detekce tzv. „jump pointů“ [12], byla by v tomto měření odhalena umělá nepřesnost. To by znamenalo, že by program počítal posun jako rychlost změny jednotlivých shluků. Skutečný posun je ale na obrázku reprezentován oranžovou přímkou.

Proto skript změní nejen hodnotu frekvence volání výpočtu posunu, ale změní i frekvenci získávání časových informací ze serverů. Původní frekvence získávání časových informací ze serverů je jednou za 1000 ms, při zaokrouhlování *performance.now* na celé sekundy se tato frekvence změní na jednou za 1100 ms. Znamená to, že získáme méně časových informací, ale získané informace jsou pro nás užitečnější. Při změně frekvence se nabízelo použít Nyquistův-Shannonův vzorkovací teorém a získávat informace například jednou za 450 ms. Při použití periody získávání informací 450 ms, však můžeme narazit na problém, kde zmenšování této periody může být kontraproduktivní a vést ke zhoršování přesnosti výsledných posunů. Použití periody 450 ms a 1100 ms má na graf víceméně stejný vliv a to, že z grafu zmizí shluky bodů připomínající úsečky viz. obrázek 5.3.

Prvních 50 bodů v grafu 5.3 vypadá stejně a to z toho důvodu, že se rozpoznávání zaokrouhlení času funkce *performance.now* volá až po 50 získaných časových informacích.



Obrázek 5.2: Graf při zaokrouhlování na sekundy s původní frekvencí.



Obrázek 5.3: Graf při zaokrouhlování na sekundy se změnou frekvence.

Vzhledem k tomu, že při malém počtu časových informací, je velká šance, že zaokrouhlení na stovky milisekund se může zdát jako zaokrouhlení na celé sekundy. Při testování jsem narazil na sekvence až 30 časových informací, které vypadaly jako by byly zaokrouhlovány na celé sekundy, ale doopravdy však byly zaokrouhlovány na stovky milisekund.

5.3.2 Ukončení měření

Měření se ukončí, pokud se poslední vypočítaný výsledek uzná za konečný. Za konečný se uzná v případě, že rozdíl dvou po sobě vypočítaných výsledků je menší než 1 ppm. Jinými slovy, pokud se vypočítaný posun za posledních **x** nově přijatých časových informací výrazně nezměnil, bere se výsledek jako konečný. Pro každý server se počítá vlastní posun, znamená to, že doby měření jednotlivých serverů se mohou lišit. Pokud se úspěšně vypočítá posun jednoho serveru, zastaví se získávání nových časových informací tohoto serveru, ostatní servery s ještě neukončeným měřením získávají časové informace nadále.

5.4 Uložení výsledného posunu

Pokud se určí výsledek měření za konečný, uloží se do lokálního úložiště klienta. Pro ukládání se používá funkcionalita JavaScriptu zvaná *localStorage*.

LocalStorage umožňuje přístup k objektu *Storage*. Objekt *Storage* je unikátní pro každou stránku s jiným původem. Uložená data přetrvávají i po ukončení sezení. Data jsou vždy uložena jako textový řetězec. Přistupuje se k nim pomocí klíčů, které se nastavují při vkládání dat do *localStorage*.

Pro uložení výsledku, vygeneruje stránka klíč, který se skládá z názvu serveru následovaným pomlčkou a časem získaným při ukládání pomocí funkce Date.now, která vrací čas v milisekundách od unixové epochy. Přidání časové informace do klíče, zajistí za normálních okolností jedinečnost těchto klíčů. Samotná data určená pro uložení, obsahují čas doby měření, získaný z poslední přijaté časové informace daného serveru, jako čas klienta, číslo alpha přímky daného serveru a číslo beta přímky daného serveru.

Pokud se při načtení stránky zjistí, že *localStorage* obsahuje informace o předešlých měřeních, zobrazí se uživateli do tabulky.

5.5 Vlastní řešení serveru

Jako vlastní řešení serveru jsem vložil PHP skript na školní server Eva. Jelikož server Eva podporuje PHP verzi novější než je PHP verze 7.3.0 mohl jsem pro získání času použít funkci *hrtime*.

Funkce *hrtime* vrací časový údaj z monotónních hodin v nanosekundách, určující čas od bodu v čase. Monotónní čas, je takový, který se nemění při změně hodin počítače. To znamená, že na získaném časovém údaji, nebude poznat, kdy se synchronizoval čas serveru Eva, který se synchronizuje každou hodinu.

Obrázek 5.4: Odpověď serveru Eva na dotaz.

Získaný časový údaj v nanosekundách skript zabalí do JSONu a odešle jako odpověď na dotaz klienta. Tato odpověď je zobrazena na obrázku 5.4. Vlastní řešení je dostupné na školních stránkách[1].

[1] http://www.stud.fit.vutbr.cz/~xjires02/

Kapitola 6

Grafické rozhraní

Při tvorbě grafického rozhraní byl kladen důraz hlavně na jednoduchost a funkčnost. Celá stránka je rozdělena na pět částí. A to hlavička, část pro pole s časovými informacemi, graf, tabulka pro předešlá měření a legenda časů. Některé z těchto částí obsahují v levém horním rohu ikonku informací, která po kliknutí na ni, zobrazí podrobnější informace o dané části. Hlavička obsahuje název stránek *PC Fingerprinter JS* a v pravém horním rohu při probíhajícím měření červené tlačítko, sloužící k zastavení měření. Legenda časů obsahuje základní informace o relevantních časech, například, která funkce vrací čas s jakou přesností atd...

6.1 Zobrazení časových informací serverů

Každý server má svoje vlastní políčko, kam se zapisují časové údaje získané ze serverů. Tato políčka jsou tři, pro servery JSONTest, WorldClock a Eva. Každý nově přijatý časový údaj se zde zobrazí ve formátu:

```
čas získaný ze serveru -- čas získaný klientem
```

Tyto časové údaje jsou zobrazeny z více důvodů. Uživatele ujistí, že výpočet stále probíhá, při zobrazování nových časových informací. Uživatele informují o době měření a také na těchto časových informacích, lze sledovat fluktuace offsetů. Čas získaný klientem také informuje, jestli je zaokrouhlován a pokud ano, tak na kolik desetinných míst.

Nad každým polem je zobrazen název serveru, kterému tyto časové informace náleží. Pod polem se po uběhnutí určité doby měření zobrazí mezivýsledky posunu. Tyto mezivýsledky mohou naznačovat přibližnou podobu výsledného posunu. Pokud se mezivýsledky mění pouze minimálně, může to znamenat, že se měření chýlí ke konci. Pole pro časové informace serverů jsou vyobrazeny na obrázku 6.1. Na obrázku je vidět, že server JSONTest a server WorldClock již mají ukončené měření, server JSONTest neúspěšně a server WorldClock úspěšně. Pro server Eva stále měření probíhá. Na obrázku je také vidět, že je čas klienta zaokrouhlován na milisekundy a to z toho důvodu, že obrázek vyobrazuje měření stránky spuštěné webovým prohlížečem Firefox, který čas získaný z funkce *performance.now* zaokrouhluje na milisekundy.

Měření daného serveru může skončit třemi způsoby. První způsob, kdy dojde k úspěšnému vypočtení konečného posunu. V tomto případě se nastaví ohraničení pole daného serveru jako zelená čárkovaná čára a pod pole se velkým, tučným písmem napíše výsledný

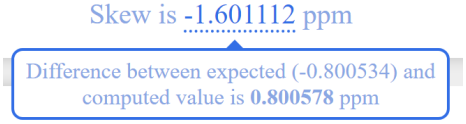
Obrázek 6.1: Ukázka polí pro časové informace serverů.

konečný posun. Do pole se vloží text potvrzující úspěšnost výpočtu. Ukázku tohoto způsobu ukončení je možné vidět na obrázku 6.1, konkrétně prostřední pole serveru WorldClock.

Druhý způsob ukončení měření je zmáčknutí červeného tlačítka umístěného v pravém horním rohu. Po jeho zmáčknutí se ukončí měření všech serverů, a nastaví se ohraničení jejich pole na červenou čárkovanou čáru. Do jejich polí se vloží text potvrzující, že měření byla zastavena. Pokud ještě před zmáčknutím červeného tlačítka, některý ze serverů úspěšně skončil své měření, zmáčknutí tlačítka tento server neovlivní. Tedy zůstane se zeleným ohraničením. Co se napíše pod pole zastavených serverů, záleží na více faktorech, zejména jestli je pro daný server k dispozici mezivýsledek. Mezivýsledek nemusí být k dispozici, když má server malé množství získaných časových informací.

Třetí možnost ukončení měření, je při přijmutí chybné zprávy od serveru. V tomto případě se zastaví zasílání dalších dotazů danému serveru, do pole se vloží text s informací, že nastala chyba a ohraničení pole se nastaví na červenou čárkovanou čáru. Toto je možné vidět na obrázku 6.1, konkrétně první pole serveru JSONTest. Pod polem se zobrazí tlačítko, které po zmáčknutí znovu aktivuje zasílání dotazů danému serveru. Tato aktivace může vést k pokračování měření, kdy přijatá chyba byla součástí pouze krátkého výpadku. Pokud však i při další odpovědi přijde chyba, znovu se zastaví zasílání dalších dotazů. Uživatel může takto zkoušet dostupnost serveru neomezeně. Toto tlačítko zmizí pouze v případě, že je server opět funkční a nebo, když všechny servery skončily svá měření.

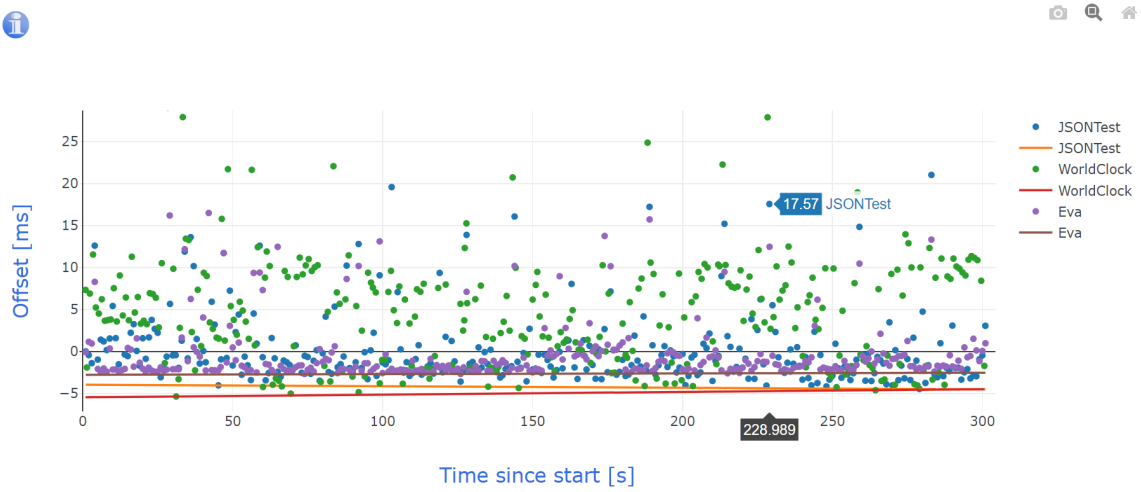
Pokud se pod polem nachází již finální výsledek posunu, může obsahovat tečkované podtržení a to v případě, že se na zařízení nacházejí uložené informace o předešlých měřeních. Tento jev je zobrazen na obrázku 6.1. Konkrétně pod středním polem serveru WorldClock. Z předešlých měření se vybere měření, které trvalo nejdéle a jeho hodnota se porovná s výsledkem právě skončeného měření. Informace o tomto porovnání se zobrazí po najetí na tečkovaně podtržený výsledek. Vzhled těchto informací je vyobrazen na obrázku 6.2.



Obrázek 6.2: Ukázka zobrazení porovnání výsledku s předešlým.

6.2 Graf

Graf se zobrazí poprvé po určitém čase, kdy je nasbíráno dostatečné množství časových informací. Tento čas je nastaven na 40 sekund. Graf se vykreslí pomocí javascriptové knihovny **plotly.js**[1]. Tento graf obsahuje veškeré zatím přijaté body a může obsahovat i přímku dolního ohraničení všech bodů serveru. Pro každý server se zvolí jiná barva bodů a přímky, aby je bylo možné rozlišit. Na pravé straně grafu je zobrazena legenda, která obsahuje názvy serverů a jejich přiřazené barvy. Z důvodu principu fungování skriptu pro vykreslování bodů, se přiřazené barvy mohou při aktualizaci bodů a přímek měnit. Společně s grafem se zobrazí text, obsahující informaci o tom, za jak dlouho se graf aktualizuje. Doba mezi aktualizacemi je nastavena na 100 sekund. Tato doba je vysoká z důvodu náročnosti vykreslení grafu. Ukázka grafu probíhajícího měření je vyobrazena na obrázku 6.3.



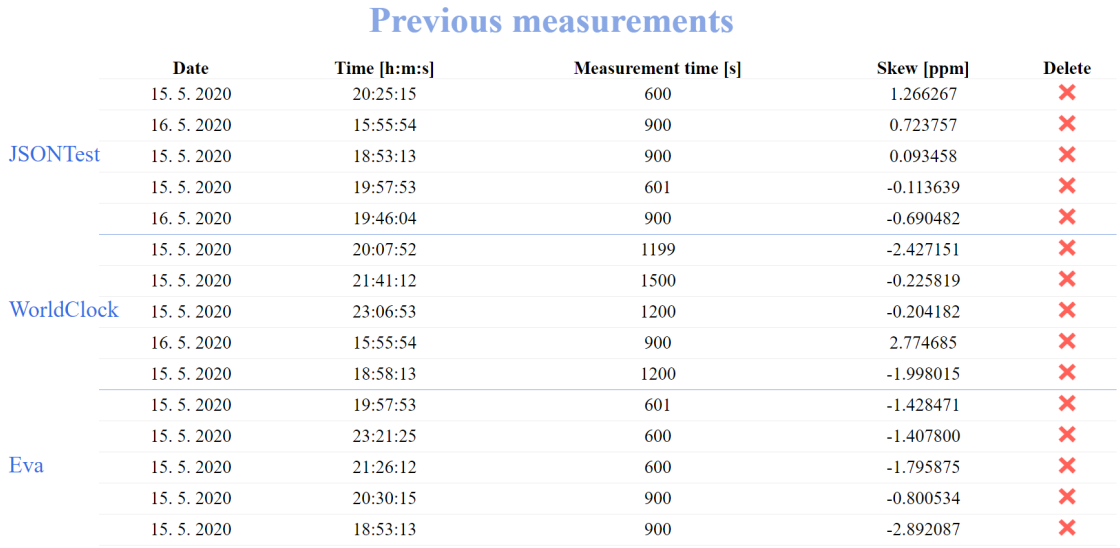
Obrázek 6.3: Ukázka vzhledu grafu.

Graf je interaktivní, je možné ho přibližovat pomocí kliknutí a táhnutí, čímž se vymezí prostor, který se přiblíží. Oddálit zpět, lze pomocí dvojitého kliknutí do grafu a nebo kliknutí na domeček, který se zobrazí v pravém horním rohu grafu. Při najetí myší na jednotlivé body grafu, se zobrazí jejich souřadnice. Kliknutím na název serveru v legendě grafu, je možné tyto body/přímku zobrazit nebo schovat. Pomocí dvojitého kliknutí na název serveru v legendě grafu, je možné tyto body/přímku izolovat, tedy schovat všechny ostatní body/přímky. Kliknutí na tlačítko fotoaparátu v pravém horním rohu grafu, je možné uložit si zobrazený graf jako obrázek s příponou *png*.

6.3 Tabulka předešlých měření

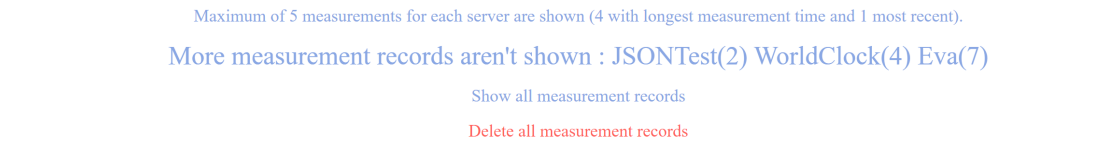
Pokud skript zjistí, že *localStorage* obsahuje informace o předešlých měřeních, vypíše tyto informace do tabulky. Ukázka takové tabulky je vyobrazena na obrázku 6.4. Každý řádek tabulky představuje, jedno úspěšné měření konkrétního serveru. Názvy serverů jsou napsány na levé straně tabulky. Aby bylo možné rozlišit konec řádků jednoho a začátek řádků druhého serveru, jsou tyto řádky odděleny modrou čárou.

[1] https://plotly.com/javascript/



Obrázek 6.4: Ukázka vzhledu tabulky předešlých měření.

Pokud skript zjistí, že informací o předešlých měření je velké množství, zobrazí maximálně pět záznamů pro každý server. Které záznamy se zobrazí, při velkém množství informací, není náhodné. Vybírá se na základě informace o době trvání měření a času, kdy byl záznam uložen. Pět záznamů se vybere tak, že se vyberou čtyři záznamy s nejdelší dobou trvání měření. Přidá se k nim záznam, který je nejnovější, tedy má datum uložení nejblíže k současnosti. Pokud již nejnovější záznam je v seznamu čtyř záznamů s nejdelší dobou trvání měření, vybere se další záznam s nejdelší dobou trvání měření ze zbylých. Pokud chce uživatel ukázat všechny záznamy pohromadě, může kliknout na tlačítko „Show all measurement records“, které do tabulky přidá i nezobrazené záznamy. Pod tabulkou se také nachází informace o tom, kolik záznamů jakého serveru je zrovna nezobrazeno. Tyto informace je možné vidět na obrázku 6.5.



Obrázek 6.5: Ukázka informací pod tabulkou předešlých měření.

Záznam je možné smazat kliknutím na červený křížek na příslušném řádku. Po kliknutí se stránka zeptá uživatele, zda si je jist, že chce záznam smazat. Pokud uživatel smazání potvrdí, záznam se nevratně smaže. Je možné smazat veškeré uložené záznamy a to pomocí tlačítka „Delete all measurement records“, nacházejícího se pod tabulkou. Po kliknutí na toto tlačítko, je uživatel také tázán, zda si je jist. Po potvrzení se nenávratně smažou veškeré informace uložené v *localStorage*.

Kapitola 7

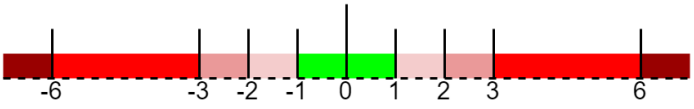
Měření

Měření bylo prováděno na dvou počítačích, nadále jako *počítač 1* a *počítač 2*. Cílem testů je zkoumání, účinnosti zaokrouhlování času získaného javascriptovou funkcí *performance.now*, programu JavaScript Restrictor. Tato zaokrouhlování jsou popsána v kapitole 3.2. Testování bylo provedeno pomocí dvou typů testů. První typ testů byl prováděn, testováním identifikace počítače při pevně dané délce měření. Druhý typ testů se zaměřil na zjištění doby trvání měření, při různých úrovních zaokrouhlení. Jeden test se bere jako určení tří posunů, každý posun vůči jinému ze serverů JSONTest, WorldClock a Eva. Pokud při testu poslal některý ze serverů chybovou hlášku, test pokračuje, pouze určí méně posunů (podle toho kolik serverů poslalo chybovou hlášku).

Testy také nabízí kontrast různých přesností časů získaných ze serverů a různých vzdáleností klienta od serveru. Testy na obou testovaných počítačích byly prováděny na území České republiky. Umístění serverů, bylo zjištěno pomocí webové stránky[1], která po zadání IP adresy serveru, ukáže přibližnou lokaci serveru. Stránka umístila lokaci serveru WorldClock do státu Iowa ve Spojených státech amerických. Pro server JSONTest pak umístila lokaci do Irska a server Eva se nachází v Brně v České Republice. Co se týče přesnosti časů získaných ze serverů, více informací v kapitolách 4.1.3 a 4.1.4, JSONTest dává čas s přesností na jednotky milisekund, WorldClock dává čas s přesností na stovky nanosekund a server Eva dává čas s přesností na jednotky nanosekund.

Pro měření bylo důležité zjistit očekávané hodnoty posunů vůči jednotlivým serverům. Bylo proto vyhrazeno několik dlouhých měření. Tato měření trvala déle než 1 hodinu. Pro zjištění optimální hodnoty se ukázalo nejlepší měření trvající 3600 sekund, tedy 1 hodinu, které získávalo časové údaje ze serveru jednou za 500 ms. Při zjištění optimálních hodnot se ukázalo, že hodnoty posunů jednotlivých serverů pro jeden počítač jsou si podobné. V tomto měření posuny pro *počítač 1* vyšly v rozmezí 0,16 ppm a pro *počítač 2* v rozmezí 0,25 ppm. Tyto očekávané výsledky jsou použity pro porovnávání správnosti výsledných posunů testů.

Pro názornost jsem výsledné posuny, vždy rozdělil do určitých rozmezí, viz. obrázek 7.1.



Obrázek 7.1: Rozmezí podle rozdílů výsledných posunů vůči očekávanému posunu.

[1] https://www.maxmind.com/en/geoip2-precision-demo

7.1 Problémy se servery

Hned na začátku testování, bylo zjištěno, že server WorldTime, popsaný v kapitole 4.1.2, dává velice nekonzistentní výsledky. Při poslání dvou požadavků za sekundu po dobu jedné minuty, bylo změřeno, že na přibližně polovinu požadavků odpoví chybovou hláškou. Proto bylo učiněno rozhodnutí, že server WorldTime bude vyřazen ze seznamu serverů pro testování.

Při delších testovacích sezeních, bylo také zjištěno, že server JSONTest má nastavené kvóty, kde při jejich překročení, dojde k deaktivaci posílání časových údajů. Při celodenních testováních byly tyto kvóty překročeny, vždy v přibližně stejnou dobu a to v 21:00 CEST. Není jisté, jak značně se na překročení kvót podílelo testování, zda je tento výpadek pravidelným jevem, nebo to bylo zapříčiněné hlavně dlouhým testováním.

U serveru WorldClock nebyly zjištěny žádné pravidelné výpadky. Výpadky nastaly pouze občasně po dobu několika minut. Server Eva fungoval po dobu celého testování bez problémů.

7.2 Testování při pevné délce doby měření

Cílem tohoto typu testování, bylo zjistit rozdíl mezi výslednými posuny vnitřních hodin počítače, kdy jediná měnící se hodnota byla úroveň zaokrouhlení času získaného javascriptovou funkcí *performance.now*. Pro zaokrouhlování časových informací bylo použito bezpečnostní rozšíření JavaScript Restrictor.

7.2.1 Zjištění optimální doby měření

Prvním úkolem, bylo zjistit jaká hodnota doby měření je pro toto testování optimální. Při zjišťování této hodnoty jsem použil literaturu primárně „Clock-Skew-Based Computer Identification: Traps and Pitfalls“ [9] a bakalářskou práci Barbory Frankové [3]. Tyto práce se zabývají dobou trvání měření testů a blízkost výsledného posunu k očekávanému. V práci [9] byly testovány tři způsoby získávání časových značek a to Python, TCP a JavaScript. U získávání časových značek pomocí JavaScriptu bylo naměřeno, že 53,4 % konverguje v prvních 20 sekundách, 96,9 % konverguje do 3,5 minut a 99,7 % konverguje do 6 minut. Tyto časové značky byly odesílány periodicky měřeným počítačem, tyto periody se pohybovaly v rozmezí 100 ms a 1000 ms.

Stránka zjišťuje informace jednou za 1000 ms. Optimální hodnotu doby měření jsem nadále hledal stylem pokus-omyl. Zkoumání jsem začal na hodnotě 4 minuty. Při této době měření, bylo po pěti pokusech identifikace počítače jasné, že je hodnota moc malá. Například ani jeden z posunů vůči serveru Eva se nepřiblížil do rozsahu ± 2 ppm očekávané hodnoty. Při pokusu ovlivnit správnost výsledných posunů získáváním více časových informací ze serverů, tedy zmenšením periody získávání časových informací z 1000 ms na 500 ms, bylo zjištěno, že tyto informace navíc mají kladný vliv pouze pro posun vůči serveru Eva. U serverů JSONTest a WorldClock výsledky naznačují, že při zmenšení periody získávání časových informací nedošlo ke zlepšení výsledných posunů. Dokonce při použití period menších jak 500 ms docházelo u všech serverů ke zhoršování přesnosti výsledných posunů. Z těchto důvodů, jsem při dalších testech zanechal periodu zjišťování časových informací na původních 1000 ms.

Při prodloužení doby měření na 5 minut se zlepšily výsledky serveru Eva. Pro server Eva bylo naměřeno 57,1 % výsledků v rozmezí ± 1 ppm, 85,7 % pak v rozmezí ± 2 ppm.

U serverů JSONTest a WorldClock, byly výsledky opačné, kde u serveru JSONTest bylo 83,3 % výsledků mimo rozmezí ± 3 ppm. U serveru WorldClock bylo mimo rozmezí ± 3 ppm 71,4 % výsledků. Z těchto výsledků bylo jasné, že se doba měření bude lišit od doby měření naměřené v [9]. To je způsobeno více faktory, hlavní z nich může být, že v [9] bylo měření prováděno v laboratorním prostředí a tato stránka měření provádí se získáváním času ze serveru vzdáleného tisíce kilometrů. Další příčina může být, že v [9] byl použit tzv. „synchronised sampling“, který stránka nepoužívá. K prodloužení doby měření také přispívá získávání časových informací přímo z prohlížeče a zpoždění s tím spojené. Další pokus, byl prodloužit dobu měření na 10 minut. I při takto dlouhém měření se u serveru WorldClock pohybovalo pouze 20 % výsledných posunů v rozmezí ± 3 ppm od očekávaného.

Jako optimální doba měření se ukázala až hodnota 20 minut.

7.2.2 Bez zaokrouhlování času z klienta (úroveň 0)

Byla použita doba měření 20 minut s periodou získávání časových informací 1000 ms. Celkem na obou počítačích bylo provedeno 116 těchto testů.

Pro *počítač 1* jsou výsledky měření zobrazeny v tabulce 7.1. Vůči serveru JSONTest bylo 87,2 % výsledných posunů v rozmezí ± 2 ppm od očekávaného posunu. Veškeré výsledné posuny se pak vešly do rozmezí ± 3 ppm. Největší rozdíl výsledného posunu a očekávaného posunu pak byl 3,2 ppm. Vůči serveru WorldClock bylo 93,2 % výsledných posunů v rozmezí ± 2 ppm od očekávaného posunu. Na rozdíl od měření vůči serveru JSONTest, měření vůči serveru WorldClock obsahovalo i 5 % výsledných posunů mimo rozmezí ± 3 ppm. Kde největší rozdíl výsledného posunu a očekávaného posunu byl 6,4 ppm. Vůči serveru Eva bylo 98,3 % výsledných posunů v rozmezí ± 2 ppm od očekávaného posunu. Největší rozdíl výsledného posunu a očekávaného posunu byl pak 3,2 ppm. Zajímavé zjištění je, že měření s největšími rozdíly posunů serverů WorldClock a Eva nastaly v jednom měření, ovšem rozdíl posunu pro server JSONTest toto měření bylo v rozmezí ± 1 ppm od očekávaného.

Počítač 1	Celkem	0 - 1 ppm	1 - 2 ppm	2 - 3 ppm	3 - 6 ppm	6+ ppm
JSONTest	55	30	18	7	0	0
WorldClock	59	36	19	1	2	1
Eva	59	51	7	0	1	0

Tabulka 7.1: Tabulka počtů měření a jejich rozdělení do rozmezí pro *počítač 1* a úroveň 0.

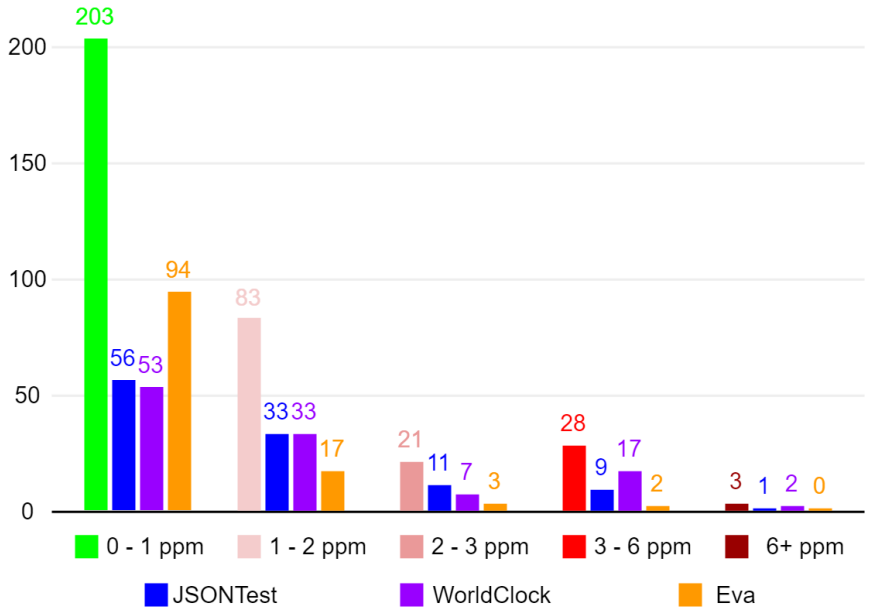
Počítač 2 dopadl s měřeními bez zaokrouhlování celkově hůře než *počítač 1*. Čím to bylo si nejsem jistý, pravděpodobně byla vyšší aktivita na síti a proto mohli offsety více fluktuovat. Výsledky měření je možné vidět v tabulce 7.2. Vůči serveru JSONTest bylo naměřeno 74,5 % výsledných posunů v rozmezí ± 2 ppm od očekávaného posunu. Největší rozdíl výsledného posunu a očekávaného posunu byl 6,6 ppm. Pro server Eva bylo 92,9 % výsledných posunů v rozmezí ± 2 ppm od očekávaného posunu. Největší rozdíl výsledného a očekávaného posunu byl 4,4 ppm. Server WorldClock dopadl nejhůře, pouze 58,4 % výsledných posunů bylo v rozmezí ± 2 ppm od očekávaného posunu. Vysoké zastoupení výsledků bylo v rozmezí od ± 3 ppm do ± 6 ppm od očekávaného posunu a to 28,3 %. Největší rozdíl výsledného a očekávaného posunu byl 5,4 ppm.

Na obrázku 7.2 jsou výsledky měření obou počítačů dané dohromady. Výsledné posuny jsou rozděleny do rozmezí vymezených v obrázku 7.1. U každého rozmezí je ukázán celkový počet výsledných posunů spadajících do tohoto rozmezí a zastoupení jednotlivých serverů

Počítač 2	Celkem	0 - 1 ppm	1 - 2 ppm	2 - 3 ppm	3 - 6 ppm	6+ ppm
JSONTest	55	26	15	4	9	1
WorldClock	53	17	14	6	15	1
Eva	57	43	10	3	1	0

Tabulka 7.2: Tabulka počtů měření a jejich rozdělení do rozmezí pro *počítač 2* a úroveň 0.

v tomto rozmezí. Je zde vidět že celkově 85 % všech výsledných posunů spadá do rozmezí ± 2 ppm od očekávaných posunů. Nejvíce posunů náleží do rozmezí s rozdílem 1 ppm od očekávaných posunů a to 60 %. Rozmezí s rozdílem 1 - 2 ppm od očekávaných posunů obsahuje 25 % výsledných posunů. Rozmezí 2 - 3 ppm pak 6 % posunů, rozmezí 3 - 6 ppm obsahuje 8 % posunů a do posledního rozmezí s rozdílem více jak 6 ppm náleží 1 % výsledných posunů.



Obrázek 7.2: Výsledné posuny měření pro úroveň 0, rozřazené do rozmezí.

Výsledky těchto testů jsou použity jako základní hodnoty, ke kterým se bude vztahovat míra účinnosti zaokrouhlování na základě testů při zapnutém zaokrouhlování.

7.2.3 Zaokrouhlení času z klienta na stovky milisekund (úroveň 1)

Doba měření stále zůstala 20 minut s periodou získávání časových informací 1000 ms. Těchto testů, se zaokrouhlením na stovky milisekund bylo celkem na obou počítačích provedeno 96. Při měření se zapnutým zaokrouhlováním na stovky milisekund bylo zjištěno, že procenta přijatelných výsledků jsou nižší než u vypnutého zaokrouhlování, jejich rozdíl však není drastický.

Pro *počítače 1* jsou výsledky možné vidět v tabulce 7.3. Vůči serveru JSONTest patří z 72.9 % do rozmezí ± 2 ppm od očekávaného posunu. Pouze 8,3 % výsledků pak má vyšší rozdíl výsledného a očekávaného posunu větší než 3 ppm, největší z těchto rozdílů je 7,1 ppm. Pro výsledky vůči serveru WorldClock platí, že 65,9 % patří do rozmezí ± 2 ppm od očekávaného posunu a 15,9 % výsledných posunů má větší rozdíl než 3 ppm od očekávaného

posunu. Největší rozdíl pro výsledný posun a očekávaný posun je 5,7 ppm. Výsledky vůči serveru Eva jsou nejlepší, 89,6 % náleží do rozmezí ± 2 ppm od očekávaného posunu. Pouze 6,2 % výsledných posunů pak má rozdíl vůči očekávanému posunu větší než 3 ppm. Největší rozdíl výsledného a očekávaného posunu je 6,2 ppm.

Počítač 1	Celkem	0 - 1 ppm	1 - 2 ppm	2 - 3 ppm	3 - 6 ppm	6+ ppm
JSONTest	48	23	12	9	3	1
WorldClock	44	16	13	8	7	0
Eva	48	34	9	2	1	2

Tabulka 7.3: Tabulka počtů měření a jejich rozdělení do rozmezí pro *počítač 1* a úroveň 1.

U *počítače 2* jsou výsledky zaznamenány v tabulce 7.4. Zaznamenaly zde větší rozdíl oproti měření bez zaokrouhlování pouze výsledky posunů vůči serveru JSONTest. V rozmezí ± 2 ppm od očekávaného posunu, klesl počet výsledných posunů o 74,5 % na 54,3 %. Procento výsledných posunů s větším rozdílem od očekávaného posunu, než 3 ppm vzrostl na 28,2 %. Výsledky vůči serveru WorldClock v rozmezí ± 2 ppm od očekávaného posunu, zaznamenaly pouze malé zmenšení počtu a to na 55,3 %. Vysoký počet byl výsledných posunů s rozdílem větším než 3ppm, vůči očekávaném posunu a to 29,8 %. To znamená, že byl stejný počet posunů v rozmezí ± 1 ppm od očekávaného posunu a posunů s rozdílem větším než 3ppm. Největší rozdíl výsledného a očekávaného posunu byl 8,8 ppm. Výsledky posunů vůči serveru Eva zapnuté zaokrouhlování významně neovlivnilo. Všechny tyto výsledky náleží do rozmezí ± 3 ppm od očekávaného posunu vůči serveru Eva. V rozmezí ± 2 ppm od očekávaného bylo 91,7 % výsledných posunů. Největší rozdíl posunu očekávaného a výsledného byl 2,6 ppm.

Počítač 2	Celkem	0 - 1 ppm	1 - 2 ppm	2 - 3 ppm	3 - 6 ppm	6+ ppm
JSONTest	46	13	12	8	8	5
WorldClock	47	14	12	7	12	2
Eva	48	32	12	4	0	0

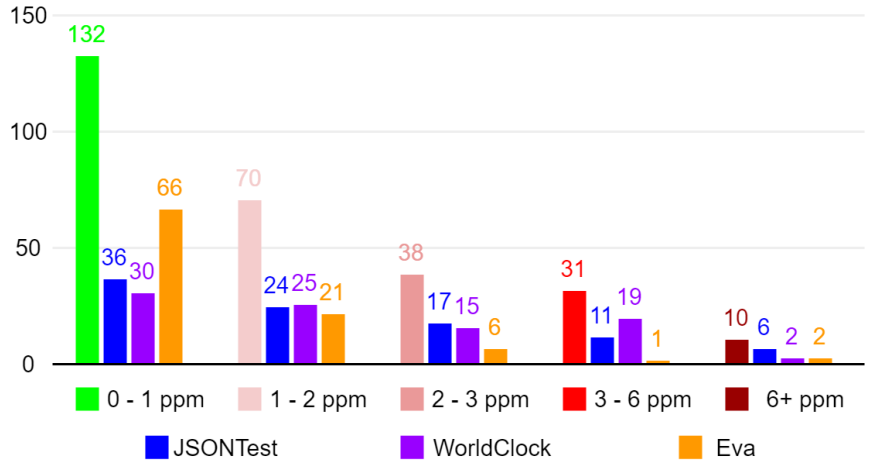
Tabulka 7.4: Tabulka počtů měření a jejich rozdělení do rozmezí pro *počítač 2* a úroveň 1.

Na obrázku 7.3 jsou zobrazeny výsledky obou počítačů v jednom grafu. Oproti předešlé úrovni zaokrouhlování došlo k posunu výsledných posunů do rozmezí s vyššími rozdíly. Rozmezí 0 - 1 ppm již neobsahuje většinu výsledků, obsahuje jich pouze 47 %. Zastoupení rozmezí 1 - 2 ppm se od předešlé úrovně nezměnilo, zůstalo 25 %. Do rozmezí 2 - 3 ppm náleží stále 13 % všech výsledných posunů, do rozmezí 3 - 6 ppm pak 11 % výsledných posunů a do rozmezí s rozdíly posunů větší jak 6 ppm náleží 4 % všech výsledných posunů. Největší zvýšení zastoupení zaznamenalo rozmezí 2 - 3 ppm.

7.2.4 Zaokrouhlení času z klienta na desítky milisekund (úroveň 2)

Doba měření a perioda získávání časových informací se při testování nezměnila. Těchto testů bylo provedeno celkem na obou počítačích 91.

Výsledky *počítače 1* jsou vypsány v tabulce 7.5. Z výsledných posunů *počítače 1* vůči serveru JSONTest náleží do rozmezí ± 2 ppm od očekávaného posunu, 53,5 %. Výsledných posunů s rozdílem větším jak 3 ppm od očekávaného posunu, bylo 30,2 %. Což je zvýšení



Obrázek 7.3: Výsledné posuny měření pro úroveň 1, rozřazené do rozmezí.

skoro o 22 % oproti výsledkům nižší úrovně zaokrouhlení. Největší rozdíl výsledného a očekávaného posunu byl 8,4 ppm. Pro výsledné posuny vůči serveru WorldClock pak do rozmezí ± 2 ppm od očekávaného posunu spadá pouze 34,9 %. Výrazně se zvětšilo procento posunů s rozdílem větším jak 3 ppm od očekávaného posunu, na 16,3 %. Největší rozdíl z těchto posunů byl 8,7 ppm. Výsledné posuny vůči serveru Eva měly opět největší zastoupení v rozmezí 0 - 1 ppm od očekávaného posunu, a to 58,1 %, pro rozšířené rozmezí ± 2 ppm od očekávaného posunu toto procento vzrostlo na 79 %. Pouze 9,3 % výsledných posunů mělo větší rozdíl posunu jak 3 ppm oproti očekávanému posunu. Největší z těchto rozdílů byl rozdíl 8 ppm.

Počítač 1	Celkem	0 - 1 ppm	1 - 2 ppm	2 - 3 ppm	3 - 6 ppm	6+ ppm
JSONTest	43	11	12	7	11	2
WorldClock	43	10	5	12	9	7
Eva	43	25	9	5	3	1

Tabulka 7.5: Tabulka počtů měření a jejich rozdělení do rozmezí pro *počítač 1* a úroveň 2.

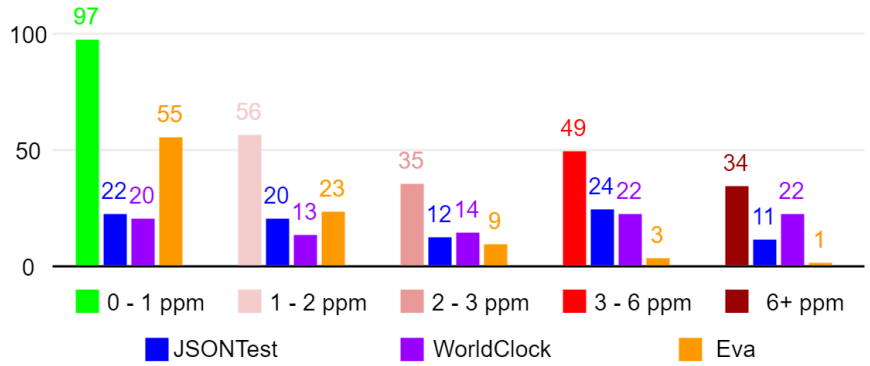
Výsledky *počítače 2* jsou na této úrovni zaokrouhlení více podobné *počítači 1*, než u předchozích úrovních. Kde u předchozích *počítač 2* dopadl hůře, u této úrovně zaokrouhlení má u některých serverů i lepší výsledky. Tyto výsledky jsou zobrazeny v tabulce 7.6. Pro výsledné posuny vůči serveru JSONTest platí, že 41,3 % patří do rozmezí ± 2 ppm od očekávaného posunu a 47,8 % má rozdíl výsledného a očekávaného posunu větší jak 3 ppm. Největší rozdíl byl 21 ppm. Nejhůře dopadl server WorldClock, 58,3 % výsledných posunů mělo rozdíl výsledného a očekávaného posunu větší než 3 ppm, 37,5 % patřilo do rozmezí ± 2 ppm od očekávaného posunu. Největší rozdíl výsledného a očekávaného posunu byl 14,5 ppm. Server Eva dopadl, lépe než u *počítače 1*. Žádný z výsledných posunů, neměl rozdíl výsledného a očekávaného posunu větší než 3 ppm. Největší rozdíl byl 2,9 ppm. Do rozmezí ± 2 ppm od očekávaného posunu náleželo 91,7 %. Je zajímavé, že oproti úrovni 1 se servery JSONTest a WorldClock zhoršily přibližně o 10 - 20 %, server Eva má výsledky stejně dobré.

Na obrázku 7.4 jsou ukázány výsledky obou počítačů dané dohromady. Stále převažuje rozmezí 0 - 1 ppm s 36 % celkového zastoupení. Rozmezí 1 - 2 ppm má zastoupení 21 %, roz-

Počítač 2	Celkem	0 - 1 ppm	1 - 2 ppm	2 - 3 ppm	3 - 6 ppm	6+ ppm
JSONTest	46	11	8	5	13	9
WorldClock	48	10	8	2	13	15
Eva	48	30	14	4	0	0

Tabulka 7.6: Tabulka počtů měření a jejich rozdělení do rozmezí pro *počítač 2* a úroveň 2.

mezí 2 - 3 ppm je zastoupeno 13 % všech výsledných posunů, stejně jako u předešlé úrovně zaokrouhlení, rozmezí 3 - 6 ppm náleží 18 % výsledných posunů a v posledním rozmezí s rozdíly výsledného a očekávaného posunu většími, než 6 ppm se nachází 12 % všech výsledných posunů. Oproti minulým úrovním zaokrouhlení, tato úroveň zaznamenala nejvyšší nárůst výsledných posunů v rozmezí 6+ ppm. Třetím nejvíce zastoupeným rozmezím se stalo rozmezí 3 - 6 ppm.



Obrázek 7.4: Výsledné posuny měření pro úroveň 2, rozřazené do rozmezí.

7.2.5 Zaokrouhlení času z klienta na sekundy (úroveň 3)

Doba měření a perioda získávání časových informací se při tomto testování nezměnila. Těchto testů bylo provedeno celkem na obou počítačích 95. U obou počítačů došlo při této úrovni zaokrouhlení k výraznému zhoršení výsledných posunů.

Počítač 1 zaznamenal největší změny v rozmezí s rozdíly výsledného a očekávaného posunu většími jak 6 ppm. Výsledky měření jsou ukázány v tabulce 7.7. Výsledných posunů vůči serveru JSON, je v rozmezí ± 2 ppm od očekávaného posunu pouze 25,6 %. Největší nárůst zaznamenalo rozmezí s rozdíly 6 ppm a více a to na 37,2 %. Největší rozdíl byl 64,7 ppm. Nejhůře dopadl znovu server WorldClock, kterému výsledné posuny spadaly nejvíce do rozmezí s rozdíly výsledných a očekávaných posunů více jak 6 ppm, konkrétně do tohoto rozmezí spadalo 59,6 % všech výsledných posunů. V rozmezí ± 2 ppm od očekávaného posunu je pouze 17 %. Největší rozdíl výsledného a očekávaného posunu byl 26,3 ppm. Velké zhoršení zaznamenaly i výsledné posuny vůči serveru Eva. V rozmezí ± 2 ppm od očekávaného posunu se nachází 36,2 %. Největší rozdíl výsledného a očekávaného posunu byl 361,6 ppm. Největší rozdíly serverů Eva a JSONTest nastaly ve stejném měření a jsou mnohonásobně větší než ostatní rozdíly. Zajímavé je, že při tomto měření byl naměřen vcelku průměrný posun vůči serveru WorldClock. Druhé největší rozdíly výsledných a očekávaných posunů jsou, pro server JSONTest 22 ppm a pro server Eva 20,9 ppm.

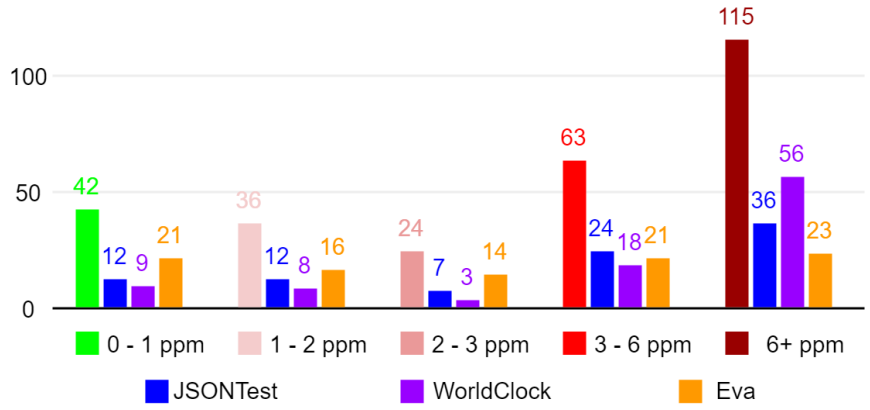
Počítač 1	Celkem	0 - 1 ppm	1 - 2 ppm	2 - 3 ppm	3 - 6 ppm	6+ ppm
JSONTest	43	5	6	3	13	16
WorldClock	47	6	2	2	9	28
Eva	47	9	8	8	9	12

Tabulka 7.7: Tabulka počtů měření a jejich rozdělení do rozmezí pro *počítač 1* a úroveň 3.

Počítač 2 dopadl podobně jako *počítač 1*. Bylo změřeno výrazné zvýšení výsledných posunů s rozdílem větším jak 6 ppm od očekávaných posunů, tato měření jsou vypsána v tabulce 7.8. Pro výsledné posuny vůči serveru JSONTest, bylo naměřeno 27,1 % posunů v rozmezí ± 2 ppm od očekávaného posunu. Největší zastoupení má rozmezí s rozdíly většími než 6 ppm a to 41,7 %. Největší rozdíl byl 26,9 ppm. Server WorldClock dopadl opět nejhůře s pouze 19,1 % posunů v rozmezí ± 2 ppm od očekávaného posunu. Zastoupení v rozmezí s rozdíly většími než 6 ppm je dokonce většinové, konkrétně 59,6 %. Největší rozdíl byl 32,8 ppm. Server Eva dopadl v této úrovni nejlépe. Do rozmezí ± 2 ppm od očekávaného posunu náleží 41,7 % výsledných posunů. Zastoupení v rozmezí s rozdíly většími než 6 ppm má server Eva menší oproti ostatním serverů a to 20,8 %. Největší rozdíl byl 24,9 ppm.

Počítač 2	Celkem	0 - 1 ppm	1 - 2 ppm	2 - 3 ppm	3 - 6 ppm	6+ ppm
JSONTest	48	7	6	4	11	20
WorldClock	47	3	6	1	9	28
Eva	48	12	8	6	12	10

Tabulka 7.8: Tabulka počtů měření a jejich rozdělení do rozmezí pro *počítač 2* a úroveň 3.



Obrázek 7.5: Výsledné posuny měření pro úroveň 3, rozřazené do rozmezí.

Na obrázku 7.5 jsou ukázány výsledky obou počítačů dané dohromady. Je názorně vidět masivní zvětšení posledního rozmezí 6+ ppm. První rozmezí 0 - 1 ppm již není nejčetnější, obsahuje pouze 15 % výsledných posunů. Do rozmezí 1 - 2 ppm náleží 13 % výsledných posunů, rozmezí 2 - 3 ppm obsahuje 9 % posunů. Rozmezí 3 - 6 ppm se stalo druhým nejčetnějším se zastoupením 22 % posunů. Nejčetnějším rozmezím se stalo rozmezí 6+ ppm, konkrétně s 41 % zastoupením. Překvapivé zjištění je u serveru Eva, kde ve všech předchozích úrovních obsahoval pouze 3 případy výsledných posunů s rozdílem větším jak 6 ppm.

V této úrovni zaokrouhlení se, ale rozmezí s rozdíly většími jak 6 ppm stalo pro server Eva nejpočetnější.

7.2.6 Zhodnocení testu

Test zkoumal vliv zaokrouhlování času získaného z klienta při pevně dané době měření na schopnost identifikace počítače pomocí vypočítání posunu vnitřních hodin.

Pro server JSONTest bylo možné pozorovat změny mezi všemi úrovněmi zaokrouhlování. Největší změny jsou však viditelné mezi úrovní 2 a úrovní 3. Změny mezi úrovní 1 a úrovní 2 jsou také patrné. Mezi úrovní 1 a úrovní 2 došlo pouze k malým změnám. Pokud budeme brát hranici úspěšné identifikace jako maximální rozdíl 2 ppm. Pak zjistíme, že při úrovni 0 byla úspěšná identifikace v 80,9 % případů. Při úrovni 1 bylo možné identifikovat počítač při 63,8 % měření. Na úrovni 2 již více jak polovinu měření identifikujeme neúspěšně, úspěšně identifikuje v 47,2 % případů. Nejméně případů pak úspěšně identifikuje při použití úrovně 3, konkrétně 26,4 %.

Server WorldClock dopadl obecně nejhůře ze všech testovaných serverů. Rozdíly úrovní jsou patrné mezi všemi úrovněmi. Při úrovni 0 bylo úspěšně identifikováno 76,7 % případů. Pro úroveň 1 toto číslo kleslo na 60,5 %. Při použití úrovně 2 toto číslo kleslo výrazně na 36,3 % a při použití poslední úrovně, úrovně 3, byla identifikace úspěšná v minimu případů, konkrétně v 18,1 %.

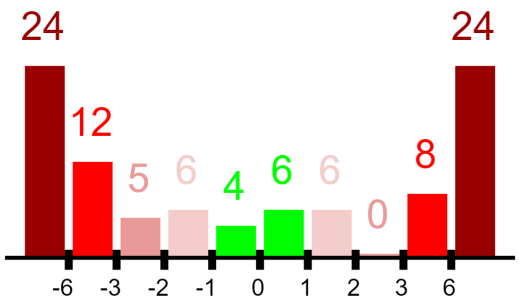
Server Eva dopadl ze všech testovaných serverů nejlépe. Změny zaokrouhlování se výrazně neprojevily, až do úrovně 2. Při použití úrovně 3 pak byl výrazný pokles v úspěšnosti identifikace. Pro úroveň 0 bylo úspěšně identifikováno 95,7 % případů, pro úroveň 1 pak 90,7 % případů. Úroveň 2 úspěšnost ovlivnila výrazně méně než u ostatních serverů a to poklesem úspěšnosti na 85,3 % případů. Při použití úrovně 3, nastal velký pokles úspěšnosti, konkrétně na 38,9 %. Úspěšnost serveru Eva při použití úrovně 3 se však dá srovnávat s ostatními servery a jejich úspěšností při použití úrovně 2.

Vlastnosti výsledných posunů

Zajímavé zjištění nastane, když uděláme průměr všech výsledných posunů vypočítaných pro jednu úroveň. Zjistíme, že i při použití úrovně 3, kde se výsledné posuny pohybují ve vzdálenostech od očekávaného posunu větších než ±6 ppm, se tyto kladné a záporné vzdálenosti při výpočtu průměru vykompenzují.

Když vypočítáme průměr všech výsledných posunů vůči serveru WorldClock, při měření pro *počítač 2*, při použití úrovně 3, kde bylo naměřeno 59,6 % posunů s rozdílem od očekávaného posunu větším než 6 ppm. Zjistíme, že tento průměr nám dá nově vypočítaný posun, který se v tomto případě přiblížil k očekávanému výsledku na 0,4 ppm. Stejně tak se na stejném počítači při stejné úrovni přiblíží server JSONTest na 0,6 ppm k očekávanému výsledku. Pro názornost jsou rozdíly serverů JSONTest a WorldClock zobrazeny na obrázku 7.6. Průměr pro server Eva vyjde hůře než u ostatních serverů. Posun vypočítaný průměrem se přiblíží na 2,2 ppm k očekávanému posunu. Při zmenšování úrovně zaokrouhlení se vypočítané průměry posunů pro jednotlivé servery přiblíží k očekávaným posunům na ±2 ppm. Největší z rozdílů mezi posunem vypočítaným průměrem a očekávaným posunem je pak 1,4 ppm.

Při pokusu o počítání průměrů výsledných posunů na *počítači 1* zjistíme, že při použití úrovně 3 dopadl výrazně hůře. Ani jeden ze serverů se nepřiblížil k očekávanému výsledku na ±1 ppm. Server JSONTest a WorldClock se přiblížili na ±2 ppm. Server Eva dopadl výrazně hůře, kde rozdíl posunu vypočítaného průměrem a očekávaného posunu byl 5,9 ppm. Při



Obrázek 7.6: Rozdíly výsledných posunů serverů JSONTest a WorldClock rozdělené do rozmezí se znaménkem.

zkoumání nižších úrovní zaokrouhlování se však tento trend neopakuje a všechny výsledné posuny vypočítané průměrem se přiblíží k očekávaným posunů na ±1 ppm.

7.3 Testování při proměnné délce doby měření

Toto testování bude probíhat tak, jak je popsáno v kapitole 5.3.2, tedy každých **x** nově příchozích časových informací se bude volat funkce pro počítání posunu. Následně se bude porovnávat posun právě vypočítaný a posun vypočítaný před **x** časovými informacemi. Pokud bude jejich rozdíl menší než ±1 ppm, uzná se poslední vypočítaný posun jako konečný. Měření tedy nemá pevně danou délku doby trvání. Doba trvání měření se ovlivňuje pouze zvyšováním nebo snižováním hodnoty **x**.

7.3.1 Zjišťování hodnot x

Vybírání hodnot **x**, probíhalo stylem pokus-omyl. Cílem bylo zajistit dobrá procenta úspěšnosti identifikace při použití co nejmenší doby měření. Úspěšná identifikace se bere jako identifikace, která má výsledný posun v rozmezí ±2 ppm od očekávaného posunu.

Pro úroveň 1 byla optimální hodnota zjištěna jako 300. Tedy po každých 300 nově přijatých časových informacích se zavolá funkce. Každých 300 nově přijatých časových informacích by mělo být stejně jako každých 300 sekund, když získáváme časové informace jednou za sekundu. Při delších měřeních je počet časových informací získaných za sekundu menší než 1. Rozdíl mezi počtem časových informací a uběhlým časem v sekundách bývá rozdíl řádově 1 %. Pro zjednodušení tedy beru počet časových informací a počet sekund od počátku měření jako stejnou hodnotu. Pro úroveň 1 je hodnota **x** 400, pro úroveň 2 pak 500. Při pokusu zvýšit hodnotu **x** pro úroveň 2 na hodnotu 600, nebyl zaznamenán žádný rozdíl v úspěšnosti identifikace, pouze doba měření trvala déle.

Úroveň 3 pak přinesla nový problém a to ten, že při jejím užití se dostává málo hodnot, které by mohli patřit do dolního ohraničení bodů v grafu. Toto bylo vyřešeno změnou frekvence získávání časových informací. To, že při použití úrovně 3, bylo přijímáno menší množství užitečných bodů, nutně neznamená, že by počítač nešlo identifikovat. Při testování identifikace, se však brzy narazilo na problém, kdy jeden test počítač identifikuje s rozdílem výsledného a očekávaného posunu na ±2 ppm. Další měření však výsledný posun mohl být úplně mimo, až 56,1 ppm rozdíl. Tato měření s takto velkými rozdíly, dokonce i vypočítaly

posun s opačným znaménkem. Kde očekávaný posun *počítače 1* je přibližně -21,2 ppm, tato měření s velkými rozdíly naměřily, posuny s hodnotami až 34,9 ppm.

7.3.2 Bez zaokrouhlení času z klienta (úroveň 0)

Pro hodnotu **x** zde byla použita hodnota 300.

Pro *počítač 1* při nezaokrouhlování času, byla zjištěna pro server JSONTest úspěšnost 81,8 %. Průměrná doba trvání měření posunu vůči serveru JSONTest byla změřena na 1090 s. Nejčastější doba trvání byla 1200 s. Doby trvání se celkově pohybovaly v rozmezí 600 - 1500 s. Úspěšnost identifikace pomocí posunu počítaného vůči serveru WorldClock byla 72,7 %, při průměrné době trvání měření 1063 s. Nejčastější doba trvání měření vůči serveru WorldClock byla 1200 s. Doby trvání se pohybovaly v rozmezí 600 - 1500 s. Úspěšnost identifikace pro server Eva byla naměřena jako 81,8 %. Průměrná doba trvání měření pro server Eva, však byla znatelně menší, než u ostatních serverů a to 845 s. S nejčastější dobou trvání měření 900 s. Tyto doby se pohybovaly v rozmezí 600 - 1200 s.

U *počítače 2* došlo ke zlepšení výsledků. Pro server JSONTest byla zjištěna úspěšnost 86,7 % při průměrné době trvání měření 1020 s. Nejčastější doba trvání měření byla 900 s a doby trvání měření se pohybovaly v rozmezí 600 - 1500 s. Serveru WorldClock byla naměřena úspěšnost identifikace 73,3 % při průměrné době trvání měření 1020 s. S nejčastější dobou trvání měření 900 s. Tyto doby trvání měření se pohybovaly v rozmezí 600 - 1800 s. Server Eva dopadl lépe než u *počítače 1*, úspěšnost byla 100 % při průměrné době trvání měření 780 s. Nejčastější doba měření byla 600 s. Doby trvání měření spadaly do rozmezí 600 - 1200 s.

Některé úspěšnosti mohou být menší, než u měření s pevně danou délkou doby měření a to z toho důvodu, že měření většinou probíhaly kratší dobu.

7.3.3 Zaokrouhlení času z klienta na stovky milisekund (úroveň 1)

Pro hodnotu **x** zde byla použita hodnota 400.

Pro *počítač 1*, byla zjištěna úspěšnost serveru JSONTest jako 75 %, při průměrné době trvání měření 1200 s. Nejčastější doba trvání měření byla 1200 s a doby trvání měření se vyskytovaly v rozmezí 800 - 2000 s. Pro server WorldClock byla zjištěna úspěšnost 100 %, při průměrné době trvání měření 1533 s. Nejčastější doba trvání měření byla 1600 s. Doby trvání měření spadaly do rozmezí 800 - 2000 s. Pro server Eva byla naměřena úspěšnost 91,7 % při průměrné době trvání měření 1067 s. Nejčastější doba trvání měření byla 1200 s. Doby trvání měření byly v rozmezí 800 - 1600 s. Zajímavé je, že server WorldClock má lepší úspěšnost než server Eva, díky tomu má však průměrnou dobu trvání o 466 s delší neboli o 43,7 % delší.

Na *počítači 2*, byla změřena úspěšnost serveru JSONTest jako 66,7 %, při průměrné době trvání měření 1440 s. Nejčastější doba trvání měření byla 1600 s a doby trvání měření spadaly do rozmezí 800 - 2400 s. Server WorldClock měl úspěšnost identifikace 73,3 %, při průměrné době trvání měření 1413 s. Nejčastější dobu trvání měření měl server WorldClock 1200 s. Tyto doby byly v rozmezích 800 - 2000 s. Server Eva měl při průměrné době trvání měření 1200 s, úspěšnost identifikace 100 %. Nejčastější doba měření byla 800 s. Doby měření serveru Eva patřily do rozmezí 800 - 1600 s.

Zajímavý je údaj serveru JSONTest na *počítači 2*, kde při zvýšené době trvání měření oproti *počítači 1*, kdy by člověk očekával, že se zvýší i úspěšnost, se úspěšnost nezvýšila, naopak se snížila o přibližně 8 %.

7.3.4 Zaokrouhlení času z klienta na desítky milisekund (úroveň 2)

Pro hodnotu **x** zde byla použita hodnota 500.

Počítač 1 naměřil, úspěšnost serveru JSONTest jako 75 %, při průměrné době trvání měření 2000 s. Oproti úrovni 1 se tedy úspěšnost nezměnila, průměrná doba trvání měření, však vzrostla o 800 s. Nejčastější doba měření byla 1500 s. Doby trvání měření spadaly do rozmezí 1000 - 3000 s. Pro server WorldClock se snížila úspěšnost na 66,7 %. Průměrná doba trvání měření se však zvýšila pouze o 134 s na 1667 s. Nejčastější doba měření byla menší než u předchozí úrovně a to 1500 s. Doby měření byly rozmístěny v rozsahu 1000 - 2000 s. Server Eva má opět úspěšnost dobrou, konkrétně 91,7 %, při průměrné době trvání 1583 s. Doba měření byla nejčastěji 1500 s dlouhá. Všechny doby měření spadaly do rozmezí 1000 - 2000 s.

Pro *počítač 2*, byla zjištěna úspěšnost serveru JSONTest jako 73,3 %, při průměrné době trvání měření 1964 s. Nejčastější doba trvání měření se oproti úrovni 1 snížila na 1500 s. Doby trvání měření spadaly do rozmezí 1500 - 2500 s. Pro serveru WorldClock se oproti úrovni 1 úspěšnost nezměnila, zůstala 73,3 %. Průměrná doba trvání měření se však výrazně zvýšila na 2100 s. Nejčastější doba měření byla 2000 s a doby měření byly v rozsahu 1000 - 3000 s. Server Eva dopadl opět nejlépe s úspěšností 80 %, při průměrné době trvání měření 1667 s. Nejčastější doba měření byla 1500 s a doby měření spadaly do rozmezí 1000 - 2500 s.

7.3.5 Zaokrouhlení času z klienta na sekundy (úroveň 3)

Pro hodnotu **x** zde byla použita hodnota 900. Tato hodnota ukazuje na počet přijatých informací. Jelikož se u této úrovně měnila i frekvence získávání nových časových informací, bude hodnota 900 znamenat volání funkce pro počítání posunu přibližně jednou za 1000 s.

Na *počítači 1* byly naměřeny zatím nejmenší úspěšnosti těchto testů. Pro server JSONTest byla tato úspěšnost 57,1 %, při průměrné době trvání 4857 s. Nejčastější doba trvání byla 5000 s. Doby trvání měření se pohybovaly v rozsahu 4000 - 5000 s. Server WorldClock naměřil úspěšnost 62,5 %, při průměrné době trvání 4631. Nejčastější doba trvání byla 5000 s. Doby trvání spadaly do rozmístění 3000 - 6000 s. Úspěšnost serveru Eva také klesla oproti předchozím úrovním, na 75 %. Průměrnou dobu trvání, však má server Eva znatelně menší, než ostatní servery a to 3276 s. S nejčastější dobou trvání 3000 s a rozsahem dob trvání 3000 - 5000 s.

Počítač 2 naměřil, úspěšnost serveru JSONTest jako 55,6 %, při průměrné době trvání 3778. Nejčastější doba trvání byla 3000 s. Doby trvání spadaly do rozsahu 3000 - 6000 s. Server WorldClock naměřil úspěšnost 50 %, při průměrné době trvání měření 3700 s a nejčastější době trvání 3000 s. Tyto doby trvání byly v rozsahu 2000 - 5000 s. Server Eva měl úspěšnost identifikace 70 %, při průměrné době trvání měření 4000 s a nejčastější doby trvání byly 3000 s a 5000 s. Rozsah těchto dob měření byl 3000 - 5000 s.

Je zajímavý kontrast skoro stejné úspěšnosti serveru JSONTest na obou počítačích, ale na *počítači 2* má server výrazně kratší průměrnou dobu měření. Podobný fenomén, lze sledovat na výsledcích serveru WorldClock, kde došlo ke zhoršení úspěšnosti o 12,5 %, ale také ke snížení doby trvání měření o přibližně 900 s. Server Eva zaznamenal opačné chování, kde *počítač 1* má lepší úspěšnost o 5 % i kratší dobu měření o přibližně 700 s. Je možné, že tyto fenomény jsou náhodné a zmizely by při vyšším počtu měření.

7.3.6 Zhodnocení testu

Test zkoumal dobu trvání měření, při zachování úspěšnosti nad 50 % a zvyšování hodnoty **x**, dle potřeby. Hodnota **x** byla zvyšována na každé úrovni zaokrouhlování jinak. Kon-

krétně pro úroveň 0 byla 300, pro úroveň 1, byla 400, pro úroveň 2 byla 500, pro úroveň 3 byla 900.

Při testech bylo zjištěno, že průměrné měření zavolá funkci pro výpočet posunu 3 - 4 krát, dokud není vypočítaný posun konečný. To potvrzuje zjištěný fakt, že se při zvýšení úrovní o jednu se zvýšila doba trvání měření přibližně o 400 s. Pro úroveň 3 bylo toto zvýšení mnohem větší.

Zjišťováno bylo také, zda všechna měření, která mají rozdíl výsledného a očekávaného posunu větší, než 2 ppm, patří mezi krátká měření. Krátké měření bylo bráno, jako takové, které bralo posun jako výsledný po třech nebo méně volání funkce počítající posun. Pro měření, bez zapnutého zaokrouhlování času klienta, byla naměřeno, že 69,2 % neúspěšných identifikací patřilo do krátkých měření. Krátké měření bylo zde bráno jako měření s dobou trvání 900 s a kratší. Pro úroveň 1, bylo krátké měření bráno, za měření s dobou trvání 1200 s a kratší. Bylo zde zjištěno, že 76,9 % neúspěšných identifikací patřilo mezi tyto krátké měření. Pro úroveň 2 bylo krátké měření bráno jako měření s dobou trvání 1500 s a méně. Tato úroveň obsahovala nejmenší procentuální zastoupení krátkých měření v neúspěšných identifikacích, konkrétně 50 %. Pro úroveň 3 bylo krátké měření bráno, jako měření s dobou trvání 3000 s nebo kratší. Výsledky pro úroveň 3 obsahují nejvíce neúspěšných měření, pouze 60 % jich lze však brát jako krátká měření.

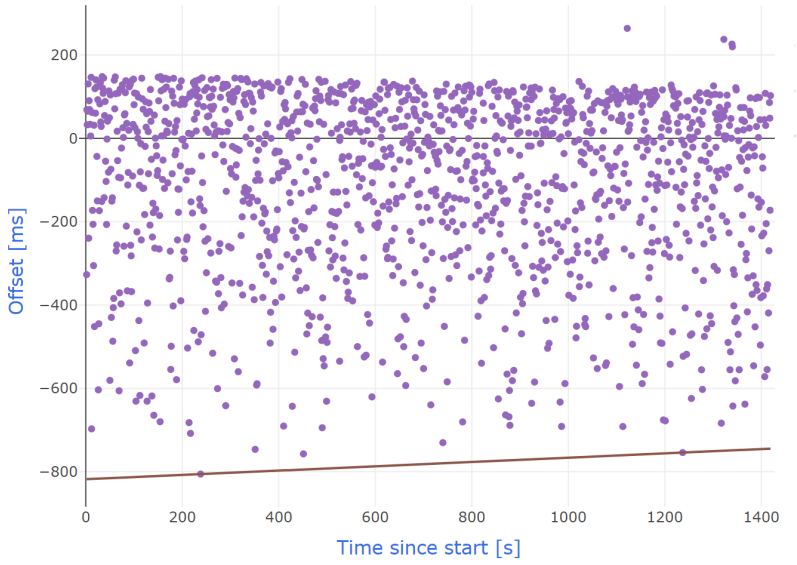
7.4 Testy měření s přidáním „fuzzy time“ k času klienta

Petr Horňák pracoval na své bakalářské práci [5] současně se mnou. Měl za cíl své práce implementovat bezpečnostní opatření Chrome Zero do rozšíření JavaScript Restrictor. Součástí této implementace bylo přidání, tzv. „fuzzy time“ do JavaScript Restrictoru. Z tohoto důvodu jsem se rozhodl udělat několik dalších testů.

Tyto testy byly prováděny za použití JavaScript Restrictoru, který zaokrouhloval čas získaný z klienta na celé sekundy a k tomu přidával náhodnou hodnotu. Prvním problémem je implementace zjišťování míry zaokrouhlení. Popsaná v kapitole 5.3.1. Tento algoritmus pouze zkoumá počet nul obsažených v získaném čase z klienta a na základě této informace rozhoduje o tom, která míra zaokrouhlení je použita. Při použití „fuzzy time“, však nedostane očekávané nuly, místo nich je vložený tzv. „noise“. To způsobí, že algoritmus pokaždé identifikuje míru zaokrouhlování jako úroveň 0, tedy bez zaokrouhlení. Přičemž použité zaokrouhlování může být kterékoliv. Jelikož se identifikuje míra zaokrouhlení jako bez zaokrouhlení, je použita pro hodnotu **x** hodnota 300.

Ze všech udělaných testů se podařilo jen jednomu serveru přiblížit k očekávané hodnotě posunu, při pohledu na graf se však s největší pravděpodobností jednalo pouze o šťastnou náhodu. Ostatní testy dávaly výsledky hodně odlišné od očekávaných, v některých případech byl rozdíl očekávaného a výsledného posunu větší než 400 ppm. Grafy všech měření, dokonce i na obou počítačích, si byly podobné. Jeden takový graf je zobrazen na obrázku 7.7. Je zde vidět, že vypočítaná přímka závisí pouze na několika bodech. Při pohledu na graf by se mohlo zdát, že při použití horního ohraničení a odstranění extrémů nad linií bodů, by mohlo přinést lepší výsledky. Toto je pouze zdání, při použití horního ohraničení dostaneme posun, který je blíže k 0 ppm, ale tento posun není nijak podobný očekávanému. Například při měření na *počítači 2*, kde je očekávaný posun 11,4 ppm dostaneme při použití horního ohraničení posun -30 ppm.

Navzdory faktu, že při těchto měřeních přiřadil algoritmus hodnotě **x** hodnotu 300, nejednalo se o krátká měření. Příčinou toho, že se nejedná o krátká měření, je vysoká



Obrázek 7.7: Výsledný graf měření při přidání „fuzzy time“ k času klienta zaokrouhlovaného na celé sekund.

šance změny přímky pří získávání nových bodů. Tato měření při učení konečného posunu, potřebovali volat výpočet posunu přibližně 4 - 7 krát.

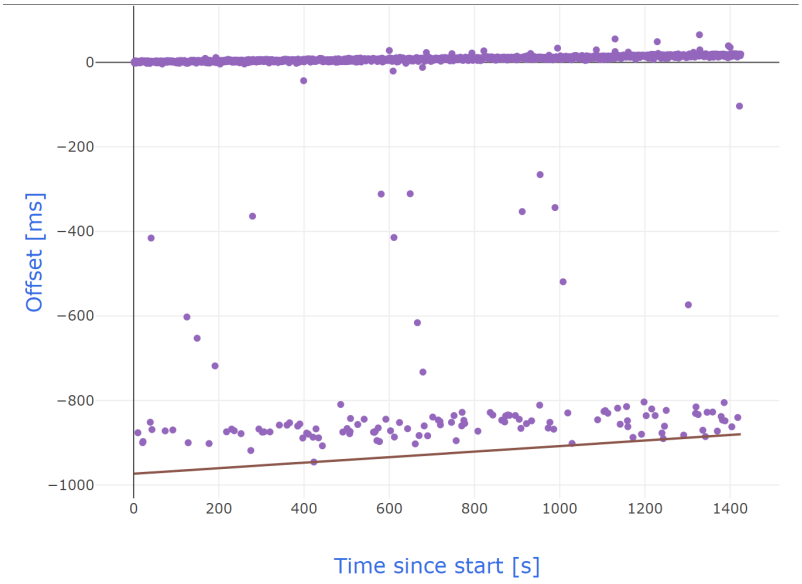
Při pokusu zvýšit hodnotu **x** na hodnotu 1000, aby měření trvalo déle, nebyla zjištěna žádná změna v úspěšnosti identifikace. Rozdíly výsledných a očekávaných posunů jsou stále příliš veliké. Většina rozdílů byla větší jak 30 ppm. Doba měření při volání výpočtu posunu jednou za přibližně 1000 s, se prodloužila, výpočet byl stále volán přibližně 4 - 7 krát.

Použití zaokrouhlování úrovně 1 s přidání „fuzzy time“.

Další bylo testováno přidávání „fuzzy time“ s použitím menší úrovně zaokrouhlení. Jako menší úroveň zaokrouhlení jsem vybral úroveň 1, tedy zaokrouhlování na desítky milisekund.

Výsledný graf je zobrazen na obrázku 7.8. Na první pohled je vidět zmenšení rozptýlení jednotlivých offsetů, důležitá část je horní shluk bodů připomínající úsečku. Stránka na ostatní body nepatřící do této úsečky není připravena. Bylo by ovšem možné tyto výkyvy offsetů identifikovat a body těchto offsetů odstranit. Kdyby tyto body byly odstraněny zbyl by pouze shluk bodů připomínající úsečku. Tato úsečka svým tvarem připomíná očekávanou přímku dolního ohraničení s korektním posunem. Lze tedy předpokládat, že při odstranění výkyvů a vypočítání přímky dolního ohraničení, by bylo možné získat očekávaný posun a počítač tedy identifikovat. Předpokládat lze ale také, že tato identifikace by trvala déle, než při použití pouze zaokrouhlování úrovně 1.

Přidání „fuzzy time“ je méně efektivní při použití nižších úrovní zaokrouhlování. Při použití nižší úrovně, v našem případě při použití úrovně 1, kde se čas z klienta zaokrouhlí na desetiny milisekund, lze ze získaných časových informací z klienta předpokládat, že přidání „fuzzy time“ ovlivňuje pouze část, která byla původně ztracena zaokrouhlováním. Tedy při použití zaokrouhlování na desítky milisekund, může přidat náhodnou složku nejvýše do jednotek milisekund. Pokud tomu tak doopravdy je, teoreticky tedy ovlivňuje jednotlivé offsety přidáním čísla z rozsahu $\langle 0, 10 \rangle$ ms, což je obrovský rozdíl oproti použití úrovně 3, kde tento rozsah může být teoreticky $\langle 0, 1000 \rangle$ ms.



Obrázek 7.8: Výsledný graf měření při přidání „fuzzy time“ k času klienta zaokrouhlovaného na desítky milisekund.

Doba trvání těchto testů se nelišila od testů s použitím zaokrouhlování úrovně 3. Počítání posunu bylo při měření voláno 4 - 7 krát a to v případě kdy hodnota **x** měla hodnotu jak 300 tak 1000.

Kapitola 8

Možná vylepšení

Optimalizace grafických prvků

U delších měření, kde bylo získáno například více jak 4000 časových údajů, při získávání dalších časových údajů, stráví prohlížeč nejvíce času aktualizováním polí pro časové informace. Po vykreslení grafu s tímto vysokým počtem bodů, je graf zpomalený a občas nereaguje na kliknutí. Pro někoho mohou být pole časových informací nicneříkající a proto zbytečná. Ve finální verzi stránky byly pole ponechány, protože si myslím, že obsahují zajímavé informace a také, protože i přesto, že stránka stráví více času aktualizováním těchto polí, měření to nijak neovlivnilo. Co se týče optimalizace grafu, bylo by možné nevykreslovat nezajímavé body.

Přidat uživateli možnost konfigurovat funkcionalitu stránky

Stránka při fungování pracuje s řadou předem daných konstant. Tyto konstanty jsou:

- *intervalTime*, který určuje jak často se získávají nové časové informace.
- *minPacketCount*, určuje minimální počet časových informací, které musí server mít, aby mohl počítat posun.
- *minTime*, určuje jak dlouho se bude čekat, než se začne počítat posun.
- *callSkewComputeTime*, proměnná, která obsahuje hodnotu **x** v kódu.
- *endWhen*, dává maximální dobu trvání měření, pokud se překročí, skončí měření s chybou.
- *defaultGraphUpdate*, udává jak často se aktualizuje graf.
- *graphUpdate*, určuje po jaké době se poprvé zobrazí graf.

Kdyby byla dána možnost uživateli tyto proměnné měnit, mohl by simulovat situace, které ho zajímají a dostat výsledky, které potřebuje. Uživatel by si mohl například ověřit testy provedené v kapitole 7.2. Pokud například uzná, že úspěšnosti identifikací, měřené v kapitole 7.3, pro různé úrovně jsou moc nízké, měl by možnost změnit hodnotu **x** dle libosti. Bylo by také možné udělat podporu změny serveru, uživatel by mohl stránce sdělit jméno a adresu jiného serveru vůči, kterému chce počítat posun a stránka by nahradila jeden z existujících serverů serverem vybraným uživatelem.

Vylepšení určování výsledných posunů

Finální verze stránky funguje na principu, že zavolá několikrát výpočet posunu s určitými časovými rozestupy a pokud jsou si dva za sebou vypočítané posuny dosti podobné určí se poslední jako konečný posun. Při tomto určování, který posun je konečný by bylo možné použít i jiné užitečné informace. Například z testů vyplývá, že posuny jednotlivých serverů jsou si velice podobné. Při určování končených posunů, by se mohl posun kontrolovat nejen s minule vypočítaným, ale i s posuny ostatních serverů. Může se ovšem stát, že posuny dvou serverů jsou si podobné i když jsou dále od očekávaného posunu než posun serveru třetího.

Stejně tak by se pro určování konečných posunů, mohli použit výsledky předchozích měření. Nastane ovšem podobný problém, že nevíme, který z předchozích výsledků je správný. Můžeme brát měření s nejdelší dobou trvání, u kterých očekáváme, že s prodlouženou dobou měření se výsledný posun přiblíží očekávanému, to však nemůžeme na jistotu nikdy zaručit. Může se stát, že kratší měření vypočítalo posun, který je bližší očekávanému. Bylo by možné využít vlastnosti výsledných posunů z kapitoly 7.2.6. Kde bylo zjištěno, že velice podobný posun očekávanému se dá získat průměrem velkého počtu výsledných posunů, s tím, že nezáleží na rozdílech těchto posunů oproti očekávaným posunům. Předpokládá se, že se tyto rozdíly vysokým počtem měření navzájem vykompenzují.

Kapitola 9

Závěr

Práce se zabývala tématem identifikace počítače pomocí zjištění jedinečného posunu vnitřních hodin. Tento výpočet byl teoreticky popsán a následně použit pro tvorbu webové stránky, která je při odevzdávání této práce dostupná na adrese[1]. Tato stránka používá tři servery pro zjištění aktuálního času, který následně použije pro výpočet posunu počítače vůči jednotlivým serverům. Stránka se liší od ostatních programů, které počítají posuny vnitřních hodin tím, že tyto programy získávají čas z měřeného počítače a porovnávají ho s reálným časem na měřícím počítači. Tyto programy jsou spuštěny na počítači, který měří ostatní počítače. Stránka je spuštěna na měřeném počítači a reálný čas získává posíláním dotazů na servery. Tento reálný čas pak porovnává s časem získaným z klienta pomocí funkce *performance.now*.

Obecně je předpokládáno, že reálný čas s vyšší přesností dá přesnější posun. Byly použity servery JSONTest, který dává čas s přesností na milisekundy a WorldClock, který dává čas s přesností na stovky nanosekund. Rozdíl mezi získanými časy jsou 4 řády. Přesto server JSONTest dopadl ve většině testů lépe než server WorldClock. Předpokládám, že to bylo dáno lokací těchto serverů. Server WorldClock se nachází ve státě Iowa ve Spojených státech amerických a server JSONTest se nachází v Irsku. Od měřeného počítače jsou vzdáleny vzdušnou čarou přibližně, server WorldClock 7700 km a server JSONTest 1600 km. Předpokládám, že tento velký rozdíl ve vzdálenosti vynahradil jejich rozdíl v přesnosti podávaných časů. Nejlépe dle očekávání dopadl server Eva, má totiž největší přesnost času, konkrétně dává čas s přesností na nanosekundy a také je server nejblíže testovanému počítači. Vzdušnou čarou byl server od testovaného počítače vzdálen pouhých 90 km. Těmito výsledky se podařilo potvrdit předpoklady [8] sekce „7.3, Accuracy of clock skew measurements“. Která předpokládá, že s větší vzdáleností od serveru rostou možné odchylky, což zapříčiní nutnost zvýšení doby trvání měření.

Tato stránka byla použita pro testování bezpečnostního rozšíření JavaScript Restrictor. Toto rozšíření, mimo jiné, dělá to, že zaokrouhluje čas získaný z javascriptových funkcí. Obsahuje různé úrovně zabezpečení, konkrétně 4. Tyto úrovně poskytují různé míry zaokrouhlování času. Kde úroveň s nejmenším zabezpečením čas nezaokrouhluje a úroveň s nejvyšším zabezpečením zaokrouhluje čas na celé sekundy. Na otestování funkčnosti tohoto zaokrouhlování při identifikaci počítače byly vytvořeny dva typy testů. První typ testoval účinnost zaokrouhlování při pevně dané délce měření, druhý typ pak testoval účinnost při měření s proměnnou délkou.

[1] http://www.stud.fit.vutbr.cz/~xjires02/pcfjs.html

Výsledky prvního typu testů naznačují, že zaokrouhlování času snižuje úspěšnost identifikace při stejné době měření. Rozdíly mezi každou úrovní byly patrné u serverů JSONTest a WorldClock. U serveru Eva tyto rozdíly mezi prvními třemi úrovněmi zabezpečení nebyly tak velké. Rozdíl mezi poslední úrovní zabezpečení a ostatními byl patrný i u serveru Eva. U poslední úrovně zabezpečení byla úspěšnost identifikace napříč všemi servery a všemi testovanými počítači 28 %. Většina výsledných posunů pak měla moc velký rozdíl oproti očekávaným posunům. Bylo však zjištěno, že při vypočítání průměrů těchto posunů na jednom počítači, se tento průměr přiblíží do dostatečné blízkosti, očekávaného posunu, potřebné pro identifikace. Při velkém počtu provedených měření, je jedno kolik úspěšných identifikací bylo provedeno, protože při počítání průměru se chyby v neúspěšných měřeních s velkou pravděpodobností vykompenzují.

Druhý typ testů zkoumal jak se musí prodloužit doby měření, aby bylo dosaženo dostatečné úspěšnosti identifikace. Při nejnižší úrovni bylo zjištěno, že doba měření se u každého serveru liší. Server Eva, podle očekávání, měl doby měření nejkratší. Servery WorldClock a JSONTest pak měly doby měření přibližně podobné. Bylo zjištěno, že při použití nejnižší úrovně zabezpečení doby měření byly pro servery JSONTest a WorldClock přibližně 1050 sekund. Pro server Eva byla tato doba přibližně 810 sekund. Testy ukázaly, že při zvyšování úrovní se zvýší doba trvání měření jednotlivých serverů přibližně o 400 sekund pro každou přidanou úroveň. Jediný rozdíl byl znovu u nejvyšší úrovně zabezpečení, kde bylo nutné změnit i frekvenci získávání časových informací ze serverů. Doba trvání měření byla pro servery JSONTest a WorldClock přibližně 4300 sekund. Pro server Eva byla tato doba 3600 sekund.

Testy prokázaly, že zaokrouhlování času samotné, nezabrání úspěšné identifikaci počítače. Při zvýšení míry zaokrouhlení stačí pro úspěšnou identifikaci zvýšit dobu trvání měření. Použitím zaokrouhlování se sníží šance úspěšné identifikace pro krátkodobé měření.

Literatura

[1] Bray, T. *The JavaScript Object Notation (JSON) Data Interchange Format* [RFC 8259]. RFC Editor, prosinec 2017. DOI: 10.17487/RFC8259. Dostupné z: https://rfc-editor.org/rfc/rfc8259.txt.

[2] Červinka, Z. *Rozšíření pro webový prohlížeč zaměřené na ochranu soukromí*. Brno, CZ, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/114921.

[3] Franková, B. *Určování identity počítače pomocí odchylky vnitřních hodin*. Brno, CZ, 2013. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/14704/.

[4] Graham, R. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*. 1972, sv. 1, č. 4, s. 132 – 133. DOI: https://doi.org/10.1016/0020-0190(72)90045-2. ISSN 0020-0190. Dostupné z: http://www.sciencedirect.com/science/article/pii/0020019072900452.

[5] Horňák, P. *Přenos bezpečnostních opatření z Chrome Zero do JavaScript Restrictor*. Brno, CZ, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

[6] Jirásek, J. *Využití časových informací pro identifikaci počítače*. Brno, CZ, 2012. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/79043.

[7] Kohno, T., Broido, A. a Claffy, K. C. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*. April 2005, sv. 2, č. 2, s. 93–108. DOI: 10.1109/TDSC.2005.26. ISSN 2160-9209.

[8] Polčák, L. *Lawful Interception: Identity Detection*. Brno, CZ, 2017. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/phd-thesis/679/.

[9] Polčák, L. a Franková, B. Clock-Skew-Based Computer Identification: Traps and Pitfalls. *Journal of Universal Computer Science*. 2015, sv. 21, č. 9, s. 1210–1233. ISSN 0948-6968. Dostupné z: https://www.fit.vut.cz/research/publication/10725.

[10] Schwarz, M., Lipp, M. a Gruss, D. *JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks* [online]. Network and Distributed Systems Security (NDSS) Symposium, San Diego, CA, USA, únor 2018 [cit. 2019-10-02]. Dostupné z: https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_07A-3_Schwarz_paper.pdf.

[11] Timko, M. *Vylepšení rozšíření pro omezení volání JavaScriptu.* Brno, CZ, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/122066.

[12] Huang, D., Yang, K., Ni, C., Teng, W., Hsiang, T. et al. Clock skew based client device identification in cloud environments. In: *Proceedings - 26th IEEE International Conference on Advanced Information Networking and Applications, AINA 2012.* Květen 2012, s. 526–533. Proceedings - International Conference on Advanced Information Networking and Applications, AINA. DOI: 10.1109/AINA.2012.51. ISBN 9780769546513.

Příloha A

Obsah paměťového média

Součástí paměťového média, přiloženého k této bakalářské práci, jsou zdrojové soubory, jak stránky pro klienta tak skriptu umístěný na serveru Eva. Dále médium obsahuje zdrojové soubory pro LaTeX a jeho možné přeložení, již výsledný text práce ve formátu *pdf* a soubor README.txt, který obsahuje doplňující informace.

Paměťové médium má tuto strukturu:

|- latex/	Složka obsahující zdrojové soubory latex.
|- src-klient/	Obsahuje zdrojové soubory stránky.
|- styles/	Složka obsahující ikony a soubor css.
|- fce.js	Obsahuje funkce pro práci s GUI.
|- pcf.js	Obsahuje funkce pro výpočet posunu.
|- pcfjs.html	HTML soubor stránky.
|- plotly.min.js	Použitá knihovna pro tvorbu grafů.
|- src-server/	Obsahuje php skript umístěný na serveru.
|- index.php	Skript generující JSON s aktuálním časem.
|- README.txt	Soubor obsahující doplňující informace.
|- xjires02.pdf	Text bakalářské práce.