

# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ  
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY  
DEPARTMENT OF COMPUTER SYSTEMS

## SYNCHRONIZACE ČASU V POČÍTAČOVÝCH SÍTÍCH

DIPLOMOVÁ PRÁCE

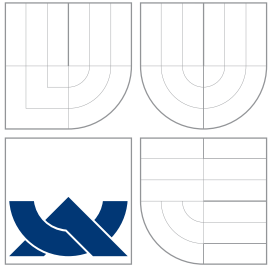
MASTER'S THESIS

AUTOR PRÁCE

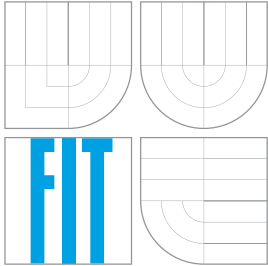
AUTHOR

Bc. DENIS MATOUŠEK

BRNO 2014



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**  
BRNO UNIVERSITY OF TECHNOLOGY



**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**  
**ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ**

FACULTY OF INFORMATION TECHNOLOGY  
DEPARTMENT OF COMPUTER SYSTEMS

# **SYNCHRONIZACE ČASU V POČÍTAČOVÝCH SÍTÍCH**

TIME SYNCHRONIZATION IN COMPUTER NETWORKS

**DIPLOMOVÁ PRÁCE**  
MASTER'S THESIS

**AUTOR PRÁCE**  
AUTHOR

**Bc. DENIS MATOUŠEK**

**VEDOUCÍ PRÁCE**  
SUPERVISOR

**Ing. TOMÁŠ MARTÍNEK, Ph.D.**

BRNO 2014

## Abstrakt

Diplomová práce se zabývá návrhem řešení pro synchronizaci času v počítačových sítích, což je klíčový problém mnoha síťových aplikací. Na základě analýzy protokolů pro synchronizaci času byl jako vhodný kandidát vybrán protokol PTP. Práce popisuje implementaci návrhu pro speciální síťovou kartu a ukazuje vlastnosti řešení na několika testech. Část řešení pro práci s přesnými časovými značkami byla implementována v čipu FPGA síťové karty, zatímco pro zpracování zpráv protokolu PTP je použita softwarová aplikace. Hodnoty konfigurovatelných parametrů aplikace byly určeny na základě analýzy vlastností síťové karty a výsledků jednotlivých testů. Dosažená přesnost se pohybuje v řádu desítek nanosekund.

## Abstract

The master's thesis deals with design of a solution for time synchronization in computer networks that is a crucial problem of many network applications. Based on analysis of protocols for time synchronization, PTP protocol was chosen as an appropriate candidate. The thesis describes the implementation of the design for a special network interface card and demonstrates features of the solution in several tests. A part of the solution processing precise timestamps was implemented in FPGA chip on the network card while PTP messages are processed in a software application. Values of configurable parameters of the application were determined based on analysis of the network card properties and results of particular tests. It was achieved accuracy in order of tens of nanoseconds.

## Klíčová slova

Synchronizace času, počítačové sítě, NTP, PTP, souběžný návrh HW/SW, FPGA, VHDL

## Keywords

Time synchronization, computer networks, NTP, PTP, HW/SW codesign, FPGA, VHDL

## Citace

Denis Matoušek: Synchronizace času v počítačových sítích, diplomová práce, Brno, FIT VUT v Brně, 2014

# Synchronizace času v počítačových sítích

## Prohlášení

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

.....  
Denis Matoušek  
19. května 2014

## Poděkování

Děkuji svému vedoucímu panu Tomáši Martínkovi za poskytnuté konzultace a rady, které mi poskytl během řešení diplomové práce.

© Denis Matoušek, 2014.

*Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.*

# Obsah

<b>1 Úvod</b>	<b>3</b>
1.1 Motivace . . . . .	3
1.2 Rozdělení dokumentu . . . . .	4
<b>2 Principy synchronizace času</b>	<b>5</b>
2.1 Reprezentace času . . . . .	5
2.2 Distribuce času . . . . .	6
2.3 Definice vybraných pojmů . . . . .	6
<b>3 Zpracování signálů ze zdrojů času</b>	<b>7</b>
3.1 FIR filtry . . . . .	8
3.2 IIR filtry . . . . .	8
3.3 Speciální filtry . . . . .	9
3.4 PID regulátory . . . . .	9
<b>4 Synchronizace času v počítačových sítích</b>	<b>12</b>
4.1 NTP . . . . .	12
4.2 PTP . . . . .	15
4.3 Porovnání protokolů NTP a PTP . . . . .	20
<b>5 Návrh řešení</b>	<b>21</b>
5.1 Přístup k návrhu řešení . . . . .	21
5.2 Návrh systému . . . . .	22
5.3 Volba platformy . . . . .	23
5.4 Návrh pro kartu COMBOv2 . . . . .	23
<b>6 Implementace</b>	<b>26</b>
6.1 Firmwareové komponenty . . . . .	26
6.2 Softwarové moduly a aplikace . . . . .	33
<b>7 Ověření funkčnosti a testování</b>	<b>42</b>
7.1 Ověření funkčnosti . . . . .	42
7.2 Testování . . . . .	43
<b>8 Závěr</b>	<b>53</b>
<b>A Příklad výpočtu zpoždění linky a offsetu protokolem PTP</b>	<b>56</b>
<b>B Příklad výpočtu hodnoty korekce po resetování hodin</b>	<b>57</b>

<b>C Makefile: cíle specifické pro kartu COMBOv2</b>	<b>58</b>
<b>D Obsah přiloženého CD</b>	<b>59</b>

# Kapitola 1

## Úvod

### 1.1 Motivace

Dnešní elektronické systémy pracují na frekvencích v řádu stovek megahertzů, což na takových zařízeních implikuje nutnost reprezentovat čas s přesností v řádu jednotek nanosekund, aby bylo možné od sebe odlišit dvě události, které se vyskytly v těsném sledu.

Jako příklad může posloužit přidělování časových značek paketům na síťové kartě pro sběr statistik o provozu na síti, např. pro měření zpoždění na lince. [21]

Dalším příkladem je behaviorální síťová analýza (angl. Network Behavior Analysis, NBA) využívající přesné časové značky paketů pro detekci neobvyklého chování, např. velkého nárůstu provozu v krátkém časovém intervalu, což může indikovat probíhající útok.

Časové značky se také ukládají spolu se zachycenými pakety pomocí knihovny Pcap<sup>1</sup> pro pozdější zpracování. Podporu časových značek s přesností v řádu nanosekund přidává nová verze knihovny PcapNg<sup>2</sup>.

Účelem synchronizace času v elektronickém zařízení je seřízení a kalibrace vnitřních hodin. Seřizování probíhá vůči referenčnímu zdroji času, přičemž snahou je přiblížit se mu s co nejmenší absolutní chybou. Pokud se k jednomu zdroji hodin synchronizuje více zařízení současně a v každém z nich je dosaženo určité přesnosti, sdílejí všechna tato zařízení jednotný čas. Je tedy možné určovat relativní pořadí výskytu událostí mezi všemi synchronizovanými zařízeními. Zobecněním tohoto přístupu lze vytvořit hierarchii časové synchronizace. Pokud na její vrchol postavíme zdroj oficiálního času (tj. UTC), celá hierarchie ho bude při úspěšné synchronizaci sdílet, což je požadovaný stav, kdy můžeme mluvit o skupině synchronizovaných zařízení. Na jednotlivých úrovních hierarchie se přitom mohou používat různé prostředky pro časovou synchronizaci. Na vrcholku hierarchie stojí primární zdroje času, typicky zařízení s atomovými hodinami. Na nižších úrovních se vyskytuje např. systém GPS, který je kromě přesného určování pozice používán i k distribuci přesného času. Ještě hlouběji se nacházejí počítačové systémy, ve kterých je synchronizace prováděna prostřednictvím počítačové sítě. Tato práce je zaměřena právě na tento způsob provádění synchronizace. Je třeba si uvědomit, že cestou od vrcholku hierarchie směrem k nižším úrovním se přesnost synchronizace postupně snižuje.

Problémem volně dostupných implementací je jejich čistě softwarový návrh, který neumožňuje pro dnešní dobu dostatečně přesnou synchronizaci. To je běžné především pro aplikace implementující NTP protokol, který se vyvíjel v době, kdy hardwarová podpora

---

<sup>1</sup><http://www.tcpdump.org/>

<sup>2</sup><http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html>

pro synchronizaci času nebyla dostupná. Volně dostupné implementace protokolu PTP jsou buď čistě softwarově založené, nebo je jejich hardwarová podpora omezená na úzký okruh zařízení. Analýzu proprietárních implementací s hardwarovou podporou komplikuje obtížná dostupnost materiálů popisujících jejich přesnost, vlastnosti a chování v různých situacích.

Přínosem této práce je vytvoření návrhu systému s hardwarovou podporou schopného přesné synchronizace času v rámci počítačové sítě. Pro ověření funkčnosti a dosažitelné přesnosti byla vytvořena referenční implementace, jejíž funkčnost byla ověřena a provedenými testy byla ukázána dosažitelná přesnost, která se pohybuje v řádu desítek nanosekund.

## 1.2 Rozdělení dokumentu

Cílem práce je pokrýt celý proces návrhu řešení pro synchronizaci času v počítačových sítích, včetně teoretické části nezbytné pro pochopení principů a požadavků kladených na takováto řešení.

Kapitola 2 je věnována času obecně, různým způsobům jeho reprezentace, měření a distribuce. Stručně zmiňuje standardy používané pro reprezentaci času a ukazuje, že otázka tohoto problému není tak zřejmá, jak by se na první pohled mohlo zdát. Stručně popisuje nejpoužívanější technologii pro přesnou distribuci času – systém GPS. Na konci kapitoly je uvedeno několik definic pojmů často používaných v textu dokumentu.

Aby bylo možné navrhnout co nejpřesnější a odolné systémy pro synchronizaci času, je nutné, s přihlédnutím na použité technologie, mít přehled o prostředích pro zpracování signálů ze zdrojů času. V kapitole 3 je popsáno několik typů digitálních filtrů a PID regulátory.

Kapitola 4 se již věnuje konkrétní aplikaci principů synchronizace času. Popisuje a porovnává dnes nejpoužívanější protokoly NTP a PTP pro synchronizaci času v počítačových sítích. Zmiňuje topologii, jakou používají pro distribuci času, způsob výpočtu základních parametrů a proces samotné synchronizace. Porovnání všech pozorovaných vlastností je shrnuto v tabulce.

Kapitola 5 popisuje důvody zvolení daného řešení a popisuje proces jeho návrhu. Shrnuje důležité vlastnosti, které musí systém splňovat, popisuje jeho strukturu, tj. části, ze kterých se skládá, včetně popisu jejich funkce. Obecný návrh je aplikován na zvolenou referenční platformu, na které byla provedena implementace.

Nejrozsáhlejší kapitola 6 se věnuje popisu vytvořených a upravených částí řešení. Jsou popsány firmwarové komponenty pro práci s PTP zprávami vytvořené pro zvolenou programovatelnou síťovou kartu. Ze softwarové části řešení se jedná o popis přidání podpory hardwarových časových značek do ovladače použité síťové karty, což umožňuje softwarové aplikaci vyčítat přesné časové značky generované přímo v hardwaru karty prostřednictvím dobře známého Linuxového síťového rozhraní. V kapitole jsou dále popsány jednotlivé moduly pro práci s firmwarovými komponentami a úpravy ve volně dostupné aplikaci zvolené jako základ pro zpracování zpráv protokolu PTP. Tyto úpravy spočívají ve využití vytvořených a upravených firmwarových komponent skrze jejich rozhraní.

Kapitola 7 popisuje způsob, jakým byla ověřena funkčnost aplikace. Nejprve popisuje ověření součinnosti všech částí aplikace a následně v podkapitole 7.2 testy pro dosažení co nejvyšší přesnosti. Testy spočívají v provádění měření, vizualizaci naměřených dat pomocí grafů a jejich zhodnocení.

Přínosy práce a výsledky dosažené implementovaným řešením jsou shrnuty v závěru v kapitole 8.

## Kapitola 2

# Principy synchronizace času

### 2.1 Reprezentace času

Z obecného hlediska lze pozorovat přesun od vnímání času na základě pozorování astronomických jevů a objektů k metodám více objektivním. Před standardizací byl čas vázán na pozorování přírodních jevů jako je východ a západ slunce. Postupem času se objevily metody založené na rotaci Země kolem své osy. Její úhlová rychlost ovšem kolísá stejně jako sklon její osy vůči rovině obíhání kolem slunce. Délka dne se tak neustále mírně mění a čas založený na těchto jevech tedy neplyne rovnoměrně. Aby byly při měření času odstraněny tyto nepravidelnosti, zavádějí se určité korekce. Jsou popsány pomocí několika časů označovaných jako UT (Universal Time) s dodatečným označením konkrétní verze. Čas *UT0* označuje čas přepočítaný podle zeměpisné délky na nultý poledník. Tento čas se na různých místech Země liší, jelikož osa zemské rotace se vzhledem k povrchu Země neustále mírně pohybuje. Čas *UT1*, či střední solární čas, dříve označován jako *GMT* (Greenwich Mean Time), aplikuje korekci tohoto pohybu zemských pólů. Jedná se o korekci v řádu desítek milisekund. Čas *UT1* je shodný na všech místech na Zemi. Jelikož však není rychlost rotace Země kolem své osy rovnoměrná, vznikají další nepravidelnosti v plynutí času *UT1* v řádu jednotek milisekund. Sezónní nepravidelnosti jsou korigovány časem *UT2*. Ten vyjadřuje co možná nejplynulejší reprezentaci času založenou na rotaci Země. I přesto obsahuje nepravidelnosti, které nejsou předvídatelné a ve výsledku vzniká chyba predikce asi 60 ms za rok. [8]

Snaha o co nejplynulejší reprezentaci času vedla ke zvolení jiného přírodního fenoménu než je vzájemný pohyb astronomických těles – jedná se o děj probíhající v atomu cesia, konkrétně je využíváno jeho rezonanční frekvence. Čas založený na tomto jevu, označovaný jako *UTC* (Universal Coordinated Time), byl roku 1972 zvolen jako celosvětový standard pro měření času. Aby bylo dosaženo co největší shody s časem *UT1* (maximální rozdíl 0,9 s), tedy tím, který vnímáme jako „přirozený“, jsou do *UTC* podle potřeby vkládány (nebo jsou z něj teoreticky odebrány) tzv. *přechodné sekundy* (z důvodu zpomalování rotace Země kolem vlastní osy). To se podle definice může uskutečnit na konci kalendářního měsíce, v praxi se tak děje na konci června nebo prosince. Pro dosažení co největší přesnosti je oficiální čas *UTC* vytvářen na základě mnoha hodin umístěných v laboratořích po celém světě (*mezinárodní soubor hodin*). Stejně tak délka trvání sekundy je určena na základě několika atomových hodin (*primárních zdrojů frekvence*). Čas vytvořený na základě mezinárodního souboru hodin a primárních zdrojů hodin je označován jako *TAI* (International Atomic Time). Délka trvání sekundy pro *UTC* i *TAI* je shodná. Nicméně do *TAI* nejsou vkládány (nebo z něj odebrány) přechodné sekundy, jako je tomu v *UTC*. Rozdíl mezi časy *UTC*

a TAI je tedy dán počtem vložených (nebo odebraných) přechodných sekund do času UTC od roku 1972, kdy byl výchozí rozdíl stanoven na 10 s. K poslednímu vložení sekundy došlo 30.6.2012, kdy rozdíl UTC vůči TAI činil 35 s.

Pro zmíněné standardy byla definována délka trvání sekundy. U středního solárního času je délka trvání sekundy definována jako  $1/86400$  délky trvání středního solárního dne. U času UTC je to doba trvání 9 192 631 770 přechodů mezi dvěma energetickými úrovněmi atomu cesia.

## 2.2 Distribuce času

V současné době se jako zdroj přesného času a současně jako prostředek distribuce času používá systém GPS (Global Positioning System). Ten využívá soubor umělých satelitů Země nesoucích atomové hodiny a řady algoritmů pro distribuci přesného času a určování přesné pozice na povrchu Země. Algoritmy využívají informací o chybě času a frekvence, odchylce (driftu) frekvence a Keplerovy elementy dráhy Země.

Čas používaný v systému GPS je odvozen od času UTC, s nímž byl sjednocen v roce 1980. Od té doby bylo do času UTC přidáno mnoho přechodných sekund. Pro účely systému GPS však není vkládání sekund přípustné, a proto čas GPS plyne stejně jako čas TAI s tím, že rozdíl mezi nimi je 19 s (což byl rozdíl mezi časy UTC a TAI v roce 1980 při zavedení času GPS) s maximální odchylkou jedné mikrosekundy. Ve skutečnosti je tato odchylka řádově menší – pohybuje se do 40 ns. [8]

Vztahy mezi jednotlivými časy jsou shrnuty v tabulce 2.1.

čas	vztah k ostatním časům	použití
UT1		1848–1972
UTC	(max. odchylka od UT1 0,9 s)	od 1972
TAI	UTC + 35 s (v roce 2013) (bez odchylky od UTC modulo 1 s)	od 1972
GPS	UTC + 16 s (v roce 2013), TAI + 19 s (max. odchylka 1 $\mu$ s od UTC/TAI modulo 1 s)	od 1972

Tabulka 2.1: Vztah jednotlivých reprezentací času

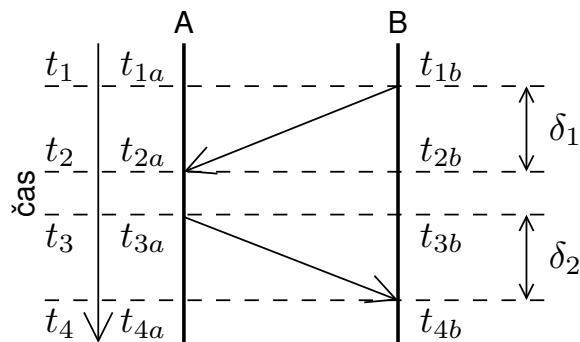
## 2.3 Definice vybraných pojmů

- *Frekvenční standard hodin* je oscilující zařízení pro určení délky sekundy.
- *Hodiny* jsou zařízení složené z frekvenčního standardu a čítače (integrátor, sčítačka, akumulátor). Při korekci hodin se setkáváme se seřizováním frekvenčního standardu hodin (angl. frequency slewing) a *krokováním* hodin (angl. clock stepping), čímž se rozumí skoková změna čítače hodin.
- *Přesnost času* je podle definice míra shody s časem UTC [8].
- *Syntonizace* je proces sjednocování frekvencí dvou zařízení.
- *Synchronizace* je proces sjednocování hodnot hodin dvou zařízení. V praxi se ale do tohoto procesu zahrnuje i proces syntonizace.

## Kapitola 3

# Zpracování signálů ze zdrojů času

Výběr prostředků pro zpracování signálů je třeba uzpůsobit formě zpracovávaných vzorků. V případě protokolů pro synchronizaci času v počítačových sítích jsou to především časové veličiny spojené s přenosem paketů po síti. Na obrázku 3.1 je zachycen průběh přenosu paketů, který je typický pro většinu dnešních protokolů používaných pro synchronizaci času (podrobněji budou popsány protokoly NTP v kapitole 4.1 a PTP v kapitole 4.2).



Obrázek 3.1: Typický průběh přenosu paketů mezi zařízeními A a B v protokolech pro synchronizaci času. Proměnné  $\delta_1$  a  $\delta_2$  označují jednosměrné zpoždění na lince. Časy s indexem  $a$  jsou relativní vůči zařízení A, časy s indexem  $b$  vůči zařízení B.

Proměnné  $\delta_1$  a  $\delta_2$  vyjadřují jednosměrné zpoždění na lince, po řadě ze zařízení B do zařízení A a ze zařízení A do zařízení B. V případě uvážení stejného zpoždění v obou směrech linky, lze vyjádřit *jednosměrné zpoždění* na lince aritmetickým průměrem  $\delta = \frac{\delta_1 + \delta_2}{2}$ . Obousměrné zpoždění je dáno vztahem  $\delta = \delta_1 + \delta_2$ . Hodnota této veličiny se měří během celé doby provádění synchronizace a je využívána při výpočtu samotného offsetu. Je třeba zajistit stabilitu hodin i v případě nestálostí sítě, např. jejího zahlcení, kdy hodnota zpoždění na lince výrazně kolísá. Pro tyto účely lze použít digitální filtry popsané v kapitolách 3.1, 3.2 a 3.3.

Další sledovanou veličinou je offset  $\theta$ , tedy rozdíl času mezi dvěma zařízeními. Ten je počítán na základě času zdroje hodin přenášeného v paketech a zpoždění  $\delta$  linky k tomuto zdroji. V případě zjištění nenulového offsetu je třeba korigovat hodiny. To musí být prováděno tak, aby nedocházelo ke skokovým nebo příliš rychlým změnám v rychlosti plynutí času na hodinách. Za tímto účelem je vhodné použít některý regulační mechanismus. V kapitole 3.4 jsou popsány v praxi nejpoužívanější PID regulátory.

### 3.1 FIR filtry

FIR (Finite Impulse Response) filtr je filtr s konečnou odezvou na vstupní impuls, tzn. že po určitém čase se ustálí na hodnotě 0. Odezva FIR filtru stupně  $N$  na jednotkový impuls na vstupu trvá  $N + 1$  vzorků. Výstup filtru je konvolucí vstupního signálu s impulzní odezvou, v případě diskrétního filtru se jedná o *vážený průměr* aktuálního vstupního vzorku a předchozích vstupních vzorků. Váhy jednotlivých vzorků pro výpočet hodnoty výstupu jsou určeny koeficienty filtru. Výstup diskrétního filtru je popsán diferenční rovnicí 3.1.

$$y[n] = \sum_{i=0}^N b_i x[n - i] \quad (3.1)$$

kde  $x[n]$  je vstupní signál,  $y[n]$  výstupní signál,  $b_i$  koeficienty filtru a  $N$  řád filtru.

Mezi základní vlastnosti FIR filtrů patří (1) *nerekurzivnost* (absence zpětné vazby, filtr se tedy nerozkmitá) a (2) *stabilita* (všechny póly jsou umístěny v počátku jednotkové kružnice, což je podmínka stability).

Jednoduchým, ale praktickým příkladem FIR filtru je *klouzavý průměr*. Všech  $N + 1$  koeficientů je nastaveno na hodnotu  $\frac{1}{N+1}$ , čímž se na výstupu získá průměrná hodnota z aktuálního vstupního vzorku a  $N$  předchozích vstupních vzorků. Zobecněním tohoto filtru lze implementovat *vážený klouzavý průměr*, který lze použít pro lineární vyhlazení skokových změn na vstupu.

### 3.2 IIR filtry

IIR (Infinite Impulse Response) filtr je filtr s nekonečnou odezvou na vstupní impuls. Na rozdíl od FIR filtrů se jejich výstup nikdy neustálí na hodnotě 0. Toto chování je způsobeno přítomností alespoň jedné zpětnovazební smyčky. Výstup filtru je popsán diferenční rovnicí 3.2.

$$y[n] = \frac{1}{a_0} \left( \sum_{i=0}^P b_i x[n - i] - \sum_{j=1}^Q a_j y[n - j] \right) \quad (3.2)$$

kde  $x[n]$  je vstupní signál,  $y[n]$  výstupní signál,  $b_i$  koeficienty dopředné části filtru,  $a_i$  koeficienty zpětnovazební části filtru,  $P$  řád dopředné části filtru a  $Q$  řád zpětnovazební části filtru.

Přítomnost zpětnovazebních smyček může způsobit problémy se stabilitou. Na rozdíl od FIR filtrů totiž všechny póly neleží v počátku jednotkové kružnice a v případě polohy mimo jednotkovou kružnici není filtr stabilní.

Výhodou IIR filtrů oproti FIR filtrům je (1) typicky ostřejší hranice přechodů ve frekvenční oblasti (angl. transition region roll-off) a (2) menší složitost výpočtů (menší řád filtru) pro dosažení shodného chování.

### 3.3 Speciální filtry

#### Huff-n'-puff filtr

Huff-n'-puff filtr<sup>1</sup> je filtr navržený pro protokol NTP. Slouží ke zmírnění důsledků zvýšení latence během zahlcení sítě. Využití najde především v sítích, kde se přenáší velké objemy dat. Filtr je založen na znalosti latence ve chvílích, kdy není síť zahlcena – zaznamená si minimální zpoždění v posledním časovém úseku. V případě zahlcení síťové linky filtr upraví naměřený offset za použití rozdílu aktuální latence a zaznamenané minimální latence v posledním časovém úseku.

Filtr je definován vztahy 3.3.

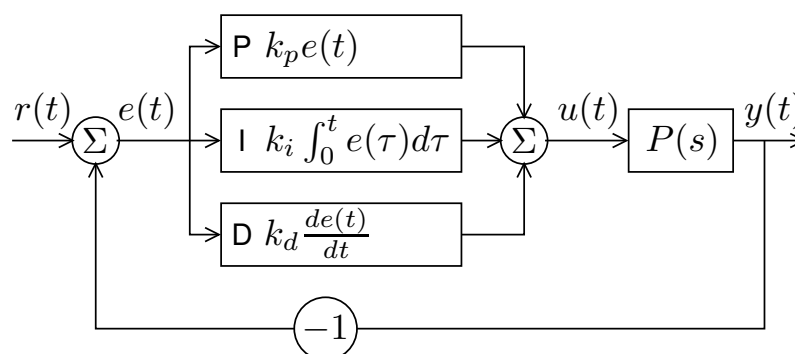
$$\begin{aligned} \theta &= y - \frac{x-x_0}{2} & \text{pro } y > y_0 \\ \theta &= y + \frac{x-x_0}{2} & \text{pro } y < y_0 \end{aligned} \quad (3.3)$$

kde proměnné  $x_0$  a  $y_0$  označují po řadě minimální zpoždění zaznamenané v posledním časovém úseku a odpovídající offset, proměnná  $x$  označuje aktuální naměřené zpoždění (obousměrné zpoždění  $\delta$ ) a  $y$  aktuální naměřený offset.

### 3.4 PID regulátory

PID regulátory jsou v průmyslové praxi nejčastěji používaným regulačním mechanismem. Uvádí se, že toto zastoupení je až 97%. Zkratka názvu vyjadřuje tři složky, které regulátor obsahuje – proporcionální, integrační a derivační. Většina v praxi používaných regulátorů tohoto typu uplatňuje pouze proporcionální a integrační složku a označují se jako PI regulátory. [9]

Schéma ideálního systému řízeného PID regulátorem je uvedeno na obrázku 3.2. Celkově se jedná o uzavřený zpětnovazební systém. Regulovaný systém je reprezentován prvkem  $P(S)$ . Toto označení zároveň reprezentuje přenosovou funkci regulovaného systému. Výstupem regulovaného systému je signál  $y$ , který reprezentuje hodnotu měřené veličiny. Vstupem systému je signál  $u$  – akční veličina. Pro zjednodušení návrhu PID regulátoru se předpokládá, že regulovaný systém je *časově invariantní* a *kauzální* (hodnota výstupu  $y$  v čase  $t_0$  závisí pouze na hodnotách vstupu  $u$  v čase  $t \leq t_0$ ). Referenční signál  $r$  (angl. setpoint) určuje požadovanou hodnotu výstupu regulovaného systému. Odchylka  $e$  výstupní hodnoty  $y$  od referenční hodnoty  $r$  je vstupem samotného PID regulátoru.



Obrázek 3.2: Schéma systému s PID regulací

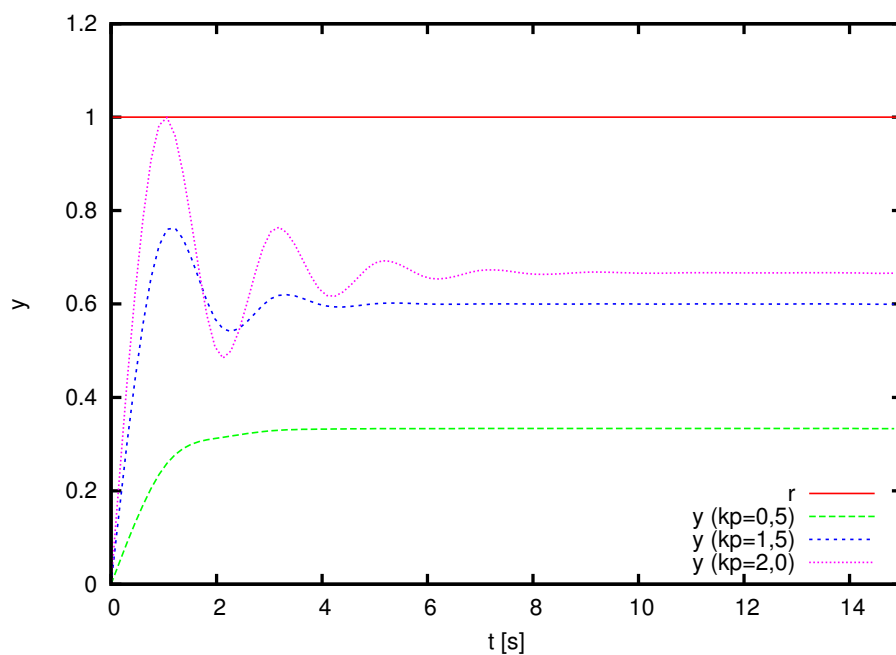
<sup>1</sup><http://www.eecis.udel.edu/~mills/ntp/html/huffpuff.html>

Dvě formy vyjádření výstupu PID regulátoru jsou uvedeny ve vztahu 3.4. První forma používá pro každou ze tří částí regulátoru multiplikativní konstantu: proporcionální zisk  $k_p$ , integrační zisk  $k_i$  a derivační zisk  $k_d$ . Druhá forma využívá proporcionální zisk  $k_p$  a dvě časové konstanty: integrační časovou konstantu  $T_i$  a derivační časovou konstantu  $T_d$ .

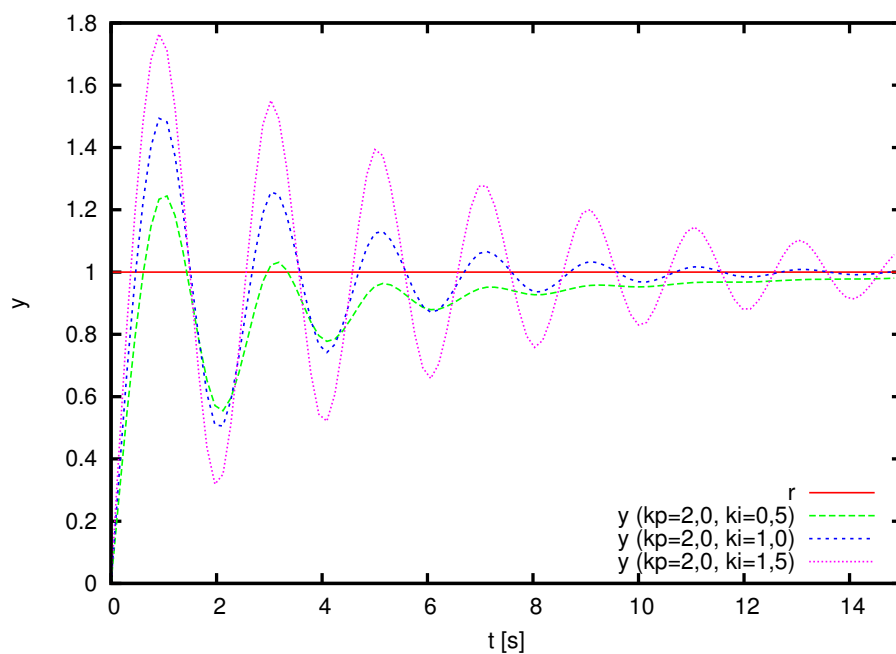
$$u = k_p e + k_i \int_0^t e(\tau) d\tau + k_d \frac{de}{dt} = k_p \left( e + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de}{dt} \right) \quad (3.4)$$

Každá ze složek PID regulátoru má svůj význam. Pro ilustraci funkce *proporcionální složky* je na obrázku 3.3 vynesena hodnota výstupního signálu systému regulovaného regulátorem, který používá pouze proporcionální složku. Je vidět, že hodnota výstupu se ustálí na určité hodnotě, která ale není požadovanou referenční hodnotou. Parametr  $k_p$  má vliv na velikost této stálé chyby a náchylnosti k oscilacím. Čím větší má hodnotu, tím menší je stálá chyba, ale výstup více osciluje. *Integrační složka* slouží k odstranění stálé chyby. Na obrázku 3.4 je vidět vliv parametru  $k_i$ . Čím vyšší má hodnotu, tím rychleji se výstup systému přiblíží referenční hodnotě, ale systém také více osciluje. *Derivační složka* vnáší do regulátoru prediktivní chování. Oscilující průběh se stává strmější a dochází tak k rychlejší konvergenci k referenční hodnotě. Pro efektivní regulaci je nutné parametry PID regulátor seřadit. Existuje několik metod, z nichž některé jsou manuální, jiné automatické.

Význam PID regulátoru v regulaci systému pro synchronizaci spočívá v přizpůsobování vlastností hodin systému (změna frekvence, příp. krokování) v závislosti na zjištěném offsetu. Problémem ale je, že regulovaný systém není kauzální – zjištěný offset je totiž odvozen z informací získávaných ze sítě a nezávisí tedy pouze na akční veličině. Tomu je potřeba podřídit zpracování informací ze sítě, aby byly získávány co nejpřesnější a nezkreslené informace pro synchronizaci.



Obrázek 3.3: Vliv parametru  $k_p$  na funkci PID regulátoru. Pokud se použije pouze proporcionální složka, výstupní hodnota  $y$  nemusí nikdy dosáhnout referenční hodnoty  $r$ .



Obrázek 3.4: Vliv parametru  $k_i$  na funkci PID regulátoru. Použití integrační složky zajistí, že výstupní hodnota  $y$  dosáhne referenční hodnoty  $r$ . Velikost parametru  $k_i$  má vliv na rychlost přibližování k referenční hodnotě a amplitudu výstupního signálu.

## Kapitola 4

# Synchronizace času v počítačových sítích

Potřeba synchronizace času v počítačových sítích se objevila už na počátku jejich rozvoje. Protokol NTP (Network Time Protocol), který vznikl za tímto účelem, je tak jedním z nejstarších protokolů používaných v síti Internet. S rozvojem a zrychlováním počítačových sítí se objevovaly nové požadavky na vlastnosti a splnění určitých parametrů. Tyto požadavky vyústily ve vytvoření specifikace protokolu PTP (Precision Time Protocol). Pro posuzování vlastností těchto protokolů je na začátek vhodné zmínit případy jejich nasazení a typicky dosažitelné přesnosti. Tyto vlastnosti jsou shrnuty v tabulce 4.1.

	<i>NTP</i>	<i>PTP</i>
nasazení	Internet, LAN	LAN
typicky dosažitelná přesnost	Internet: desítky ms LAN (ideální podmínky): 1 ms [20]	< 1 ms [4]

Tabulka 4.1: Typicky dosažitelné přesnosti protokolů NTP a PTP

Je patrné, že protokol PTP je navržen pro dosažení vyšší přesnosti než protokol NTP. Je to dáno oblastí jeho použití, kterou jsou lokální sítě. Ty obecně poskytují stabilnější komunikaci (menší výkyvy zpoždění přenosu paketů ap.) než v případě rozsáhlých sítí jakou je Internet. Protokol NTP však nabízí řadu užitečných vlastností, které ho činí odolným proti takovým nestabilitám. Navíc specifikuje kompletní sadu požadavků na algoritmy pro dosažení požadovaných vlastností. Naproti tomu standard protokolu PTP žádné algoritmy nespécifikuje. Specifikuje pouze způsob, formát a obsah komunikace mezi uzly. Použité algoritmy jsou tedy přenechány na návrháři řešení, což mu dává určitou svobodu při rozhodování a volbě algoritmů. Na druhou stranu to může přinést určité problémy při chybném návrhu, např. při zanedbání krajních situací.

### 4.1 NTP

Protokol NTP byl navržen Davidem L. Millsem [17] a jeho poslední verze je popsána v dokumentu RFC 5905 [16]. Prošel dlouhým vývojem a postupem času se v něm vystřídal mnoho konceptů a algoritmů. Aktuální je verze 4, která je popsána ve zmíněném dokumentu. Základní vlastností protokolu NTP je odolnost vůči proměnnému zpoždění na síti. Použité

algoritmy jsou schopné korigovat určité nepřesnosti či výpadky způsobené zahlcením či poruchami v síti. V následujícím textu budou zmíněny významné vlastnosti protokolu NTP a způsoby, jakými jsou v něm implementovány.

## Topologie distribuce

Synchronizace NTP protokolem probíhá v hierarchické topologii NTP serverů. Na vrcholu této topologie se nachází referenční zdroj hodin. Statická veličina popisující úroveň zanoření v této topologii, tedy vzdálenost od referenčního zdroje hodin, se nazývá *stratum*. Účelem označení NTP serveru úrovní zanoření je vyloučení vzniku smyček upřednostňováním NTP serverů s nižší hodnotou zanoření. Funkce zařízení podle úrovně zanoření je shrnuta v tabulce 4.2.

<i>stratum</i>	<i>funkce</i>
0	zařízení s atomovými hodinami, hodinami řízenými GPS přijímačem či rádiovým signálem, které nejsou synchronizovány přes síť
1	počítače pro připojení zdrojů hodin (zařízení stratum 0), fungují jako servery pro počítače stratum 2
2	I. počítače, které komunikují s několika počítači stratum 1 II. používají NTP algoritmy pro výběr nejlepších vzorků pro synchronizaci III. komunikují mezi sebou pro vytvoření stabilnějšího a odolnějšího času IV. fungují jako servery pro počítače stratum 3
3–14	I. počítače, které komunikují s NTP servery na nadřazené úrovni II. používají stejné NTP algoritmy jako počítače stratum 2 pro synchronizaci s NTP servery stratum 1
15	nesynchronizovaný NTP server v důsledku výpadku sítě

Tabulka 4.2: NTP stratum

Pro názornost je uvedena na obrázku 4.1 ukázková topologie s vyznačením vztahů mezi NTP servery.

## Výpočet zpoždění linky a offsetu

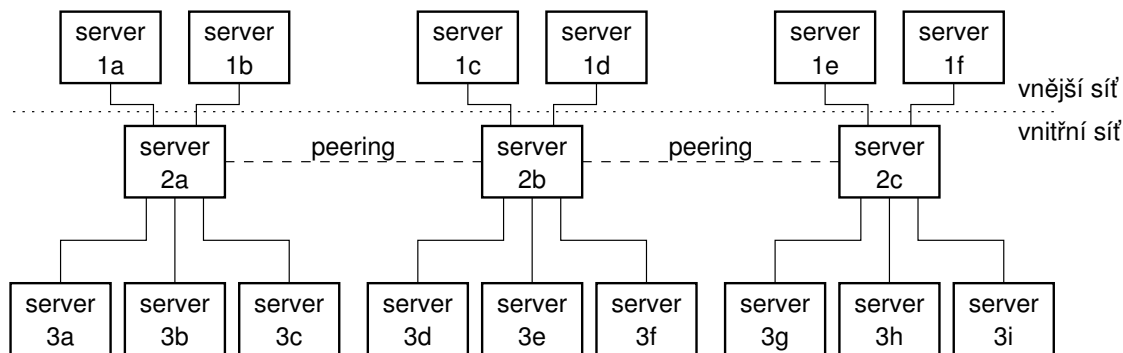
Principem výpočtu těchto dvou veličin je přenos dvou zpráv (paketů) mezi dvěma NTP servery a zaznamenání časových značek jejich odeslání a příjmu. Tento proces je zachycen na obrázku 4.2.

Zpáteční zpoždění  $\delta$  a offset  $\theta$  pro slave zařízení jsou dány vztahy 4.1 a 4.2.

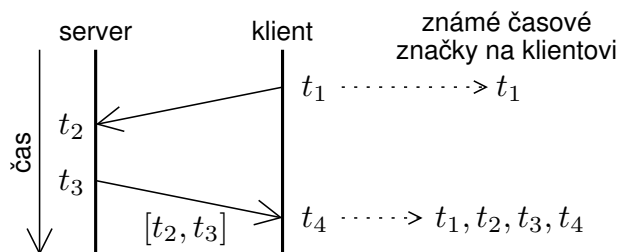
$$\delta = (t_4 - t_1) - (t_3 - t_2) \quad (4.1)$$

$$\theta = \frac{1}{2}[(t_2 - t_1) + (t_3 - t_4)] \quad (4.2)$$

Nutno podotknout, že výpočet offsetu slave zařízení vůči master zařízení předpokládá shodné zpoždění přenosu paketů v obou směrech linky. Můžeme si všimnout, že časové značky  $t_2$  a  $t_3$  je nutné dopravit na slave zařízení, aby na něm bylo možné výpočet uskutečnit. To lze provést jejich vložením do paketu zaslaného z master zařízení na slave zařízení.



Obrázek 4.1: Ukázková topologie propojení NTP serverů. Čísla označují hodnotu stratum. NTP servery stratum 0 nejsou zobrazeny. Písmena *a* až *i* označují servery na dané úrovni zanoření. *Peering* označuje sousedský vztah mezi dvěma NTP servery na stejné úrovni zanoření. Nevýhodou této topologie je jediný bod selhání mezi počítači stratum 3 a NTP servery stratum 2. Převzato z [20].



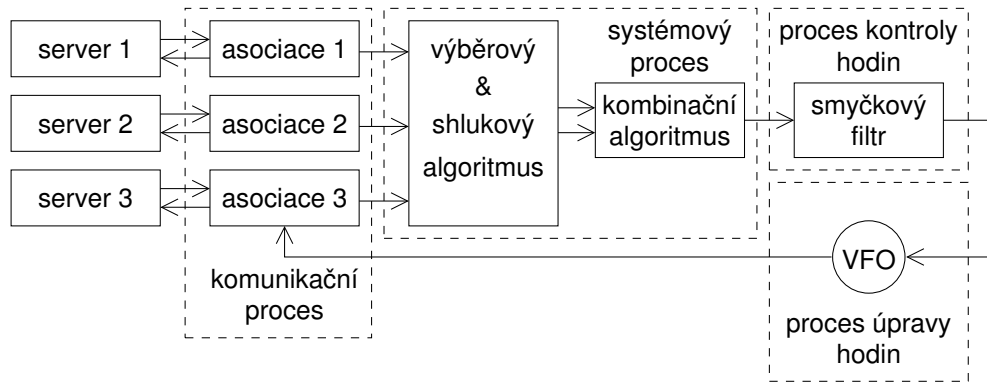
Obrázek 4.2: Přenos zpráv a zaznamenávání časových značek pro výpočet zpoždění a offsetu. Proměnné  $t_1$  až  $t_4$  reprezentují časové značky pořízené při odeslání a příjmu paketu. Každá časová značka se vztahuje k hodinám zařízení, na kterém byla zaznamenána. Hodiny obou zařízení tedy nemusí být synchronizovány. Časové značky v hranatých závorkách jsou přenášeny v tělech příslušných paketů.

Problém ale nastává s časovou značkou  $t_3$ , která reprezentuje čas odeslání paketu zasláného z master zařízení. Musí tedy existovat prostředek, který zajistí vložení odchozí časové značky do odesílaného paketu. Často používanou, a dříve také jedinou, variantou je zřeknutí se atomicity této operace a její rozdělení do dvou částí: 1) získání systémového času a 2) jeho vložení do paketu, který je následně odeslán. Toto zjednodušení s sebou však přináší nepřesnost, jelikož doba uběhnutá mezi oběma kroky se může lišit případ od případu (např. při výskytu přepnutí kontextu). Protokol PTP má prostředek pro řešení této situace, popsán je v kapitole 4.2.

## Proces synchronizace

Protokol NTP specifikuje sadu algoritmů pro výpočet finální hodnoty korekce systémových hodin. Součinnost algoritmů je zachycena na obrázku 4.3.

Systém se synchronizuje ke vzdáleným NTP *serverům* (s hodnotou stratum menší než hodnota stratum systému). *Komunikační proces* označuje funkce, které se starají o komunikaci po síti a jejichž účelem je udržování *asociací* – stavových spojení s NTP servery (nejedná se o stavovost ve smyslu protokolu TCP, ale ve smyslu udržování statistických informací mezi příjmem/zasíláním jednotlivých paketů). *Systémový proces* provádí algoritmy,



Obrázek 4.3: Schéma NTP systému, převzato z [17]

kteří se starají o samotnou synchronizaci. Jednotlivé algoritmy budou stručně popsány dále. *Proces kontroly hodin* vypočítá na základě výsledku ze systémového procesu hodnotu pro korekci hodin systému. O samotné přizpůsobení hodin se stará *proces úpravy hodin*.

Funkce jednotlivých algoritmů jsou následující:

- *filtrovací algoritmus* (angl. *clock filter algorithm*): součást komunikačního procesu, pro každou asociaci vybírá vzorky, které s největší pravděpodobností reprezentují přesný čas, využívá k tomu historii vzorků (offset, zpoždění na lince, rozptyl, čas příchodu paketu)
- *výběrový algoritmus*: ze všech asociací odstraňuje ty špatné (angl. *false tickers*) a zachovává ty dobré (angl. *true timers*)
- *shlukový algoritmus*: sérií iterací odstraňuje asociace, které jsou statisticky nejvzdálenější průměru, dokud není dosaženo postačujícího počtu asociací
- *kombinační algoritmus*: výsledkem je nejlepší odhad offsetu relativní k množině serverů, vypočtený na základě váženého průměru; vybírá jednu z asociací, která poskytuje nejlepší statistiky pro analýzu výkonu
- *algoritmus řízení hodin* (angl. *clock discipline algorithm*): na základě výsledného offsetu z kombinačního algoritmu vypočítá hodnotu pro korekci hodin systému; kombinuje koncept PLL (krokování, při vyšších hodnotách offsetu) a FLL (změna frekvence, při nižších hodnotách offsetu)

## 4.2 PTP

Protokol PTP (Precision Time Protocol) byl navržen pro synchronizaci zařízení připojených k počítačovým sítím s důrazem kladeným na jednoduchost implementace. Ta umožní protokol používat i v malých vestavěných zařízeních bez výkonné centrální výpočetní jednotky. Přesnost dosažitelná tímto protokolem se může pohybovat v řádu mikrosekund či ještě menším [4]. Této přesnosti lze ale dosáhnout pouze se zařízením s přesnými a stabilními hodinami v síti s malým kolísáním zpoždění při přenosu paketů. Svoje uplatnění nalezne tedy především v lokálních počítačových sítích. První verze standardu byla publikována v roce 2002 [3], druhá v roce 2008 [4]. Druhá verze přináší zvýšení přesnosti a robustnosti a možnost korekce pro asymetrické linky. Není zpětně kompatibilní s první verzí. Základní

principy jsou ale u obou verzí stejné a v dalším výkladu budeme uvažovat druhou verzi protokolu.

Standard protokolu PTP specifikuje následující oblasti: datové typy a jejich kódování, formát zpráv posílaných přes síť, synchronizační model, synchronizační entity, datové struktury, algoritmy pro výběr zdroje synchronizace, výpočty zpoždění a offsetů a způsob zpracování zpráv na zařízeních. Nespecifikuje však žádné metody ani algoritmy, které by zajišťovali odolnost proti nestálostem sítě.

Zprávy protokolu PTP se dělí do dvou skupin: (1) *Zprávy událostí* (angl. *event messages*), při jejichž odesílání a příjmu se generují časové značky. Slouží pro výpočet zpoždění na lince a offsetu. (2) *Obecné zprávy* (angl. *general messages*) přenášející informace, u kterých není vyžadováno rychlé zpracování ani přenos.

Pro zpřesnění hodnoty časových značek generovaných při příjmu či odesílání zpráv událostí se zavádějí parametry *ingressLatency* a *egressLatency*. První z nich umožňuje korigovat časovou značku generovanou při příjmu PTP zprávy tak, aby přesněji vyjadřovala čas příchodu zprávy na fyzické rozhraní PTP systému (místo pořizování časové značky, angl. *message timestamp point*, podrobněji popsáno v kapitole 5.1). Druhý z nich umožňuje provést stejnou korekci pro časové značky vkládané do odchozích zpráv. Naměřené časové značky jsou upravovány podle vztahů 4.3 a 4.4.

$$\text{příchozí časová značka} = \text{naměřená příchozí časová značka} - \textit{ingressLatency} \quad (4.3)$$

$$\text{odchozí časová značka} = \text{naměřená odchozí časová značka} + \textit{egressLatency} \quad (4.4)$$

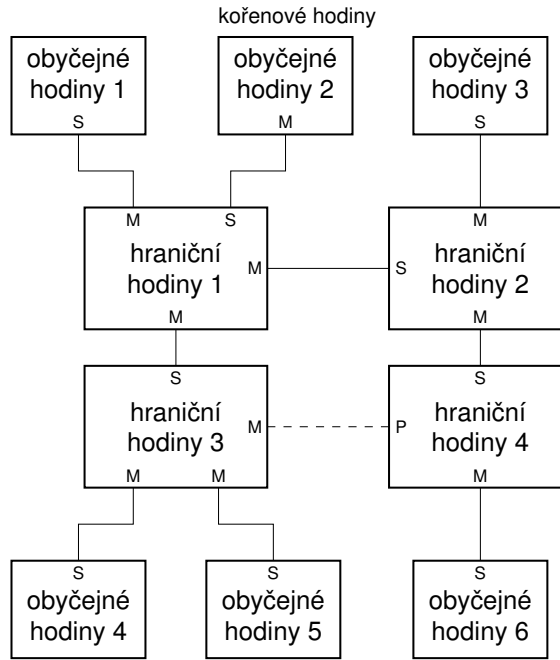
## Topologie distribuce

Na rozdíl od NTP hierarchie se jedno konkrétní slave zařízení synchronizuje k právě jednomu master zařízení a zařízení spolu nespolečně pracují pro zlepšení přesnosti synchronizace.

Stejně jako v případě protokolu NTP probíhá synchronizace v hierarchické topologii. Zařízení, které obsahuje jediný port, na kterém funguje protokol PTP, je označováno jako *obyčejné hodiny* (angl. *ordinary clock*). Toto zařízení může fungovat buď jako zdroj hodin (angl. *master clock*), nebo se samo může synchronizovat vůči jinému master zařízení (angl. *slave clock*). Pokud se jedná o master zařízení na samém vrcholku topologie, jedná se o *kořenové hodiny* (angl. *grandmaster clock*). Zařízení, které má více portů, na kterých funguje PTP protokol, je označováno jako *hraniční zařízení* (angl. *boundary clock*). Toto zařízení slouží k distribuci synchronizace z jednoho master zařízení k více slave zařízením. Z topologického hlediska se jedná o strom, jehož listy jsou tvořeny obyčejnými hodinami a vnitřní uzly hraničními hodinami.

V případě, že není zajištěna stromová topologie sítě jinými podpůrnými protokoly, *algoritmus nejlepšího zdroje* (angl. *the best master clock algorithm*) se postará o vytvoření stromové topologie na úrovni protokolu PTP (angl. *pruning mesh topology*). Pokud je detekována smyčka, přejde jeden z portů tvořících smyčku do pasivního stavu, kdy se neúčastní synchronizace. Ukázková topologie je zobrazena na obrázku 4.4.

Dalším prvkem v síti využívající synchronizaci PTP protokolem jsou *transparentní hodiny* (angl. *transparent clock*). Jedná se o zařízení, které se samo nesynchronizuje, ale zpracovává určitým způsobem procházející zprávy PTP protokolu. V případě *end-to-end* modelu zpoždění se jedná o korekci hodnoty zpoždění uložené ve zprávě přičtením času



Obrázek 4.4: Ukázková topologie PTP sítě. Symbol  $M$  označuje port v master módu, který funguje jako zdroj synchronizace. Symbol  $S$  označuje port v slave módu, který se synchronizuje. Symbol  $P$  označuje port v pasivním módu, který se neúčastní synchronizace (při detekované smyčce). Přerušovaný spoj vychází z portu v pasivním módu.

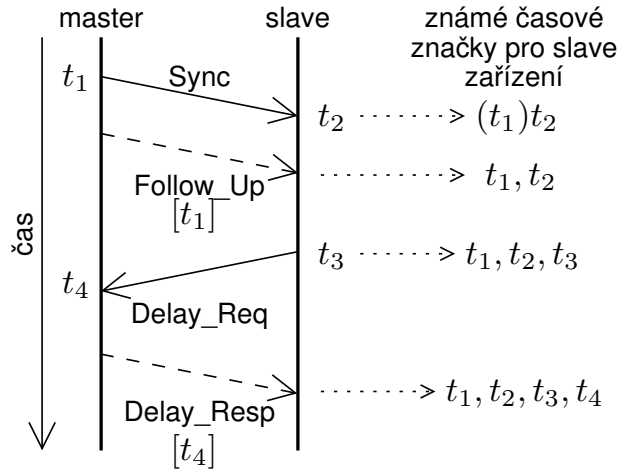
zpracování zprávy v samotném zařízení. Zařízení se tedy chová skutečně transparentně, jako kdyby se jednalo o část síťové linky. V případě *peer-to-peer* modelu si zařízení počítá zpoždění na každé lince *peer-delay* mechanismem. V procházejících PTP zprávách určených pro jiná zařízení potom přičítá toto zpoždění do speciálního korekčního políčka v PTP zprávě, které tak obsahuje sumu zpoždění na všech takových transparentních zařízeních. Tento údaj je poté použit pro výpočet celkového zpoždění k master zařízení.

## Výpočet zpoždění linky a offsetu

Princip výpočtu zpoždění a offsetu je podobný jako u protokolu NTP. Protokol PTP ale používá jiné zprávy a frekvence výpočtu zpoždění a offsetu se může lišit. Frekvence výpočtu zpoždění je typicky řádově nižší než frekvence výpočtu offsetu [4].

Jelikož uvedené vztahy počítají s odchozími časovými značkami generovanými na vysílajícím zařízení, je třeba nejprve ozřejmit způsob, jakým se řeší přenos těchto časových značek na přijímající zařízení. Problém nastává u zařízení, která nepodporují vkládání odchozí časové značky přímo do odchozí PTP zprávy události. Protokol PTP řeší tento problém zavedením příznaku *two-step*. Pokud odchozí zpráva události nemá tento příznak nastaven, je odchozí časová značka v ní obsažená platná. K tomu je však nutná podpora hardwaru, protože odchozí časová značka je pořizována co nejbližší fyzickému síťovému rozhraní systému. Pokud má zpráva události příznak *two-step* nastaven, odchozí časovou značku v sobě neobsahuje. V tomto případě je zpráva následována zprávou *Follow-Up*, která v sobě obsahuje odchozí časovou značku odpovídající zprávě události. Tím je zaručena možnost přenosu odchozí časové značky u systémů, které nepodporují její vkládání přímo do odchozí zprávy.

Přenos PTP zpráv pro výpočet zpoždění linky a offsetu pomocí *delay request-response* mechanismu je znázorněn na obrázku 4.5. Tento mechanismus slouží k měření zpoždění linky na slave zařízení. Princip výpočtu spočívá ve změření zpátečního zpoždění jako součtu zpoždění v jednom směru pomocí dvojice zpráv *Delay\_Req* a *Delay\_Resp* a ve druhém směru pomocí zprávy *Sync*. Zpráva *Sync* je používána i pro výpočet offsetu.



Obrázek 4.5: Přenos PTP zpráv pro výpočet zpoždění linky a offsetu pomocí *delay request-response* mechanismu. Plnou čarou je vyznačen přenos zpráv událostí, přerušovanou čarou přenos obecných zpráv. Časová značka  $t_1$  je v závorce, protože její přenos přímo v *Sync* zprávě závisí na podpoře vkládání odchozích časových značek do těla odesílaných paketů.

Výpočet *jednosměrného zpoždění*, v terminologii protokolu PTP označovaného jako *meanPathDelay*, je zachycen vztahem 4.5. Časové značky  $t_1$  až  $t_4$  mají význam jako na obrázku 4.5.  $\text{Sync}[c]$  reprezentuje korekční pole *Sync* zprávy,  $\text{Delay\_Resp}[c]$  korekční pole *Delay\_Resp* zprávy a  $\text{Follow\_Up}[c]$  korekční pole *Follow\_Up* zprávy. Hodnota  $\text{Follow\_Up}[c]$  se odečítá pouze pokud je nastaven *two-step* příznak v odpovídající *Sync* zprávě a ta je tedy následována *Follow\_Up* zprávou. Korekční políčka jsou upravována v transparentních zařízeních při výpočtu celkového zpoždění na složené lince mezi master a slave zařízením a při aplikování korekce na asymetrických linkách.

$$\text{meanPathDelay} = \frac{1}{2}[(t_2 - t_3) - (t_1 - t_4) - \text{Sync}[c] - \text{Delay\_Resp}[c] - \text{Follow\_Up}[c]^*] \quad (4.5)$$

Výpočet *offsetu*, v terminologii protokolu PTP označovaného jako *offsetFromMaster*, se provádí na základě příjmu zprávy *Sync* z master zařízení, které bylo naposledy zvoleno algoritmem nejlepšího zdroje. Hodnota offsetu je dána vztahem 4.6. Časové značky  $t_2$  a  $t_1$  mají význam jako na obrázku 4.5, parametr *meanPathDelay* má význam popsáný vztahem 4.5,  $\text{Sync}[c]$  reprezentuje korekční pole *Sync* zprávy a  $\text{Follow\_Up}[c]$  korekční pole *Follow\_Up* zprávy. Odečítání hodnoty  $\text{Follow\_Up}[c]$  je podmíněno stejně jako ve vztahu 4.5. Význam korekčních políček je také shodný s uvedeným vztahem.

$$\text{offsetFromMaster} = t_2 - t_1 - \text{meanPathDelay} - \text{Sync}[c] - \text{Follow\_Up}[c]^* \quad (4.6)$$

Pokud není zpoždění linky v obou směrech symetrické, lze parametrem *delayAsymmetry* určit rozdíl zpoždění v obou směrech. To je pro oba směry zachyceno vztahy 4.7 a 4.8.  $t_{ms}$

označuje zpoždění ve směru master–slave,  $t_{sm}$  zpoždění v opačném směru. Parametr *meanPathDelay* má význam popsáný vztahem 4.5. Výpočet hodnoty parametru *delayAsymmetry* není standardem definován a musí být zjištěn jinými prostředky.

$$t_{ms} = \text{meanPathDelay} + \text{delayAsymmetry} \quad (4.7)$$

$$t_{sm} = \text{meanPathDelay} - \text{delayAsymmetry} \quad (4.8)$$

Je vidět, že protokol PTP klade důraz na přesné generování časových značek a korekci všech nepřesností, které se mohou v průběh přenosu PTP zpráv sítí a jejich zpracování síťovými zařízeními vyskytnout. To ho předurčuje pro využití v sítích, kde lze tyto korekce exaktně vyjádřit, což jsou především lokální sítě.

## Proces synchronizace

Pro zahájení procesu synchronizace je nejprve nutné vybrat vhodný zdroj. K tomu slouží algoritmus nejlepšího zdroje. Standard IEEE 1588-2008 definuje jeho výchozí variantu. V podstatě se jedná o jednoduchý rozhodovací strom, který bere v úvahu následující parametry dvou porovnávaných zdrojů hodin v uvedeném pořadí:

- *priority1*: ručně zadaná hodnota vyjadřující preferenci
- *class*: parametr vyjadřující typ synchronizace (k primárnímu zdroji hodin, k aplikačně specifickému zdroji hodin atd.)
- *accuracy*: třída přesnosti
- *offsetScaledLogVariance*: rozptyl offsetu (logaritmické měřítko)
- *priority2*: ručně zadaná hodnota vyjadřující preferenci
- *identity*: identifikátor

Jedná se tedy o statické vyhodnocování vlastností zdrojů hodin, které není ovlivněné aktuální situací na síti.

Samotná synchronizace spočívá v neustálé aktualizaci hodnot parametrů *meanPathDelay* a *offsetFromMaster*. Hodnota parametru *meanPathDelay* je aktualizována na základě zasílání zpráv *Delay\_Req*, *Delay\_Resp* a *Sync*. Frekvence její aktualizace je dána parametrem *delayReqInterval* (frekvence odesílání zprávy *Delay\_Req* ze slave zařízení). Hodnota parametru *offsetFromMaster* je aktualizována na základě příjmu zpráv *Sync* z master zařízení. Frekvence její aktualizace je dána parametrem *syncInterval* (frekvence odesílání zprávy *Sync* z master zařízení).

Hodnota parametru *offsetFromMaster* se použije v samotném procesu úpravy systémových hodin. Tento proces již není standardem definován. Typicky se v něm využívá určitý mechanismus regulace.

Pokud je vyžadována pouze syntonizace, tedy dosažení shodné frekvence hodin s master zařízením, slave zařízení si vystačí pouze s příjmem *Sync* zpráv. Není tedy vyžadován výpočet parametru *meanPathDelay*.

### 4.3 Porovnání protokolů NTP a PTP

V tabulce 4.3 jsou shrnuty a porovnány vlastnosti popsané v předchozích kapitolách pro oba protokoly.

	<i>NTP</i>	<i>PTP</i>
<i>topologie distribuce času</i>	obecný graf [4.1]	strom [4.2]
<i>současné využití informací pro synchronizaci z více zdrojů</i>	ano (více zdrojů a spolupráce sousedních serverů) [4.1]	ne (vždy jen jeden master) [4.2]
<i>dosažení přesnosti</i>	sada algoritmů pro výběr dobrých vzorků, výběr vhodných zdrojů, odstranění odlehlých zdrojů, kombinace informací z několika zdrojů [4.1]	výběr zdroje založený na statických parametrech množiny zdrojů [4.2]
<i>výběr dobrých vzorků</i>	proces sousedních serverů: filtrovací algoritmus [4.1]	není definováno standardem, možnost využití FIR filtru pro úpravu offsetu, IIR filtru pro úpravu vypočteného zpoždění linky [3.1, 3.2]
<i>vyhlazení průběhu synchronizace</i>	systémový proces: algoritmus řízení hodin [4.1]	není definováno standardem, možnost využití PI(D) regulace [3.4]
<i>úprava offsetu</i>	úprava frekvence hodin / krokování času	není definováno standardem, možnost úpravy frekvence hodin / krokování času
<i>odolnost proti zahlcením sítě</i>	huff-n'-puff filtr [3.3]	není definováno standardem

Tabulka 4.3: Porovnání protokolů NTP a PTP

# Kapitola 5

## Návrh řešení

Cílem navrženého řešení by měla být specifikace systému, či subsystému, který bude synchronizovat systémové hodiny s externím zdrojem a toto řešení bude použitelné v běžných počítačových sítích. Jelikož má být důraz kladen na dosažitelnou přesnost, je potřeba uvažovat alespoň řád nanosekund. Tento požadavek plyne z nutnosti rozlišit v systému dvě těsně po sobě vyskytnuvší se události. To platí jak v případě procesorových systémů pracujících na frekvencích v řádu gigahertzů, tak v případě systémů pracujících na frekvenci v řádu stovek megahertzů (např. systémy implementované na čípech FPGA). Uvážíme-li konkrétní případ přijímání paketů na rychlých páteřních linkách s rychlostmi 10 Gb/s, je nejkratší doba mezi příchodem dvou paketů 67,2 ns [6]. V případě rychlejších sítích (40G a 100G Ethernet) se tato doba řádově snižuje. Z tohoto důvodu je pozornost zaměřena na protokol PTP, který poskytuje prostředky pro přesnost větší než v řádu nanosekund a který je určen právě pro takto náročné systémy z pohledu přesnosti reprezentace času.

Kromě přesnosti samotné je třeba přihlídnout k dalším vlastnostem, které utvářejí žádoucí chování synchronizovaného systému. Jsou to především robustnost vůči krátkodobým i dlouhodobým zahlcením sítě a rychlost dosažení určité přesnosti, bez velkých a skokových změn.

### 5.1 Přístup k návrhu řešení

Pro účely návrhu byla nejprve použita metodika analýzy systému shora dolů, tedy od větších celků ke konkrétním částem. V předchozích kapitolách o protokolech NTP a PTP bylo zmíněno několik konceptů, z nichž většina je nějakým způsobem závislá na časových značkách generovaných při odesílání či přijímání paketů. Z toho vyplývá, že pokud chceme dosáhnout určité přesnosti synchronizace systému, musíme tento požadavek reflektovat už od nejnižších vrstev. Současné operační systémy podporují generování časových značek s nanosekundovou přesností. Nicméně tyto časové značky jsou založené na získávání času v softwaru, které může být nepřesné z toho pohledu, že se čas fyzického příchodu paketu do systému může lišit od času vygenerování příslušné časové značky. To vnáší do procesu synchronizace chybu, která se dalším zpracováním zvětšuje. Moderní operační systémy nabízejí rozhraní, které umožňuje získat přesnou časovou značku fyzického příjmu, resp. odeslání, paketu. Příkladem může být rozhraní pro operační systém Linux<sup>1</sup>. Tuto funkcionalitu musí však podporovat přímo hardware, na kterém daný systém funguje.

---

<sup>1</sup><https://www.kernel.org/doc/Documentation/networking/timestamping.txt>

Místo, v němž dochází ke generování časové značky při příjmu či odeslání PTP zprávy (angl. message timestamp point) je definováno jako okamžik přechodu paketu přes hranici mezi systémem a sítí. Pokud síť používá vrstevný model, musí být definován bod v tomto modelu, kdy je časová značka pořízena. Časové značky je možné generovat na přechodu vrstev PHY a MAC nebo přímo ve vrstvě PHY, což se obejde bez účasti softwaru. Mluvíme-li o Ethernetu, časová značka by se měla generovat při příjmu prvního slova po symbolu SOF. Pokud je časová značka generována v jiném okamžiku, je možné provést korekci pomocí parametrů *egressLatency* (pro odchozí pakety) a *ingressLatency* (pro příchozí pakety), jak je popsáno v kapitole 4.2. [4]

Časové značky jsou následně zpracovávány vyššími vrstvami systému společně s dalšími statistickými údaji pro korekci vnitřních hodin systému tak, aby byly co nejpřesněji synchronizovány se zdrojem referenčních hodin.

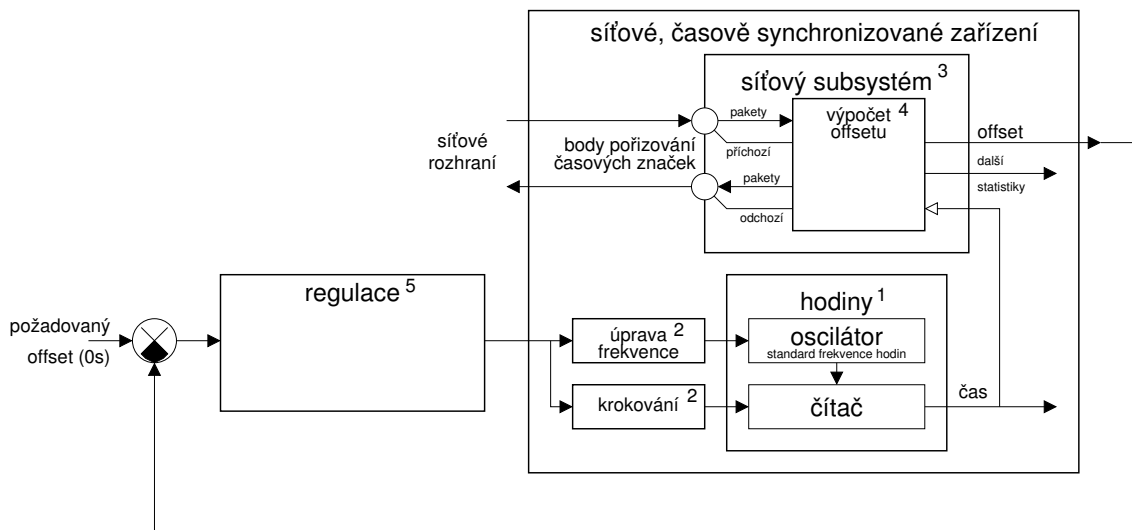
Při použití metodiky analýzy systému zdola nahoru vidíme systém provádějící netriviální výpočty pro korekci vnitřních hodin systému (filtrování vzorků, regulace...) Systém na nižších vrstvách využívá nízkoúrovňové funkce pro komunikaci po síti, pořizování časových značek odchozích a příchozích paketů a práci s vnitřními hodinami systému. Příklad systému, který celý proces implementuje v hardwaru, můžeme nalézt v [15]. Účelem takového systému je poskytnout zcela autonomní řešení, které bude fungovat bez nutnosti dalšího řízení. Tento požadavek však není vždy relevantní a v našem případě ho nebudeme brát v potaz. Pokud si uvědomíme podstatu výpočtů, zjistíme, že nejsou časově kritické a bylo by možné je implementovat v softwarové části systému. Tomu nahrává i složitost implementace operací s čísly v plovoucí řádové čárce v hardwaru.

Vezmeme-li v úvahu obě zmíněné metodiky, jednu reprezentující požadavky na pořizování časových značek na přesně definovaném místě v systému co nejbližší síťovému rozhraní, a druhou popisující systém z hlediska provádění výpočtů pro korekci vnitřních hodin systému, jako nejvhodnější přístup se jeví použití souběžného návrhu softwaru a hardwaru (angl. HW/SW codesign).

## 5.2 Návrh systému

Schéma systému je zobrazeno na obrázku 5.1.

1. *Vnitřní hodiny systému* se skládají z oscilátoru (standard frekvence hodin) a čítače (numerická reprezentace času) [8]. Frekvence oscilátoru je korigována funkcí úpravy frekvence a krokování hodin (2).
2. *Funkce úpravy frekvence a krokování hodin* má na vstupu hodnotu požadované korekce. Její přesný význam záleží na implementaci částí systému, které se této činnosti účastní. Funkce krokování skokově mění hodnotu čítače hodin. Bude nutné přesně vymezit situace, kdy se použije změna frekvence oscilátoru, skoková změna hodnoty čítače, nebo obojí současně.
3. *Síťový subsystém* komunikuje přes počítačovou síť zasláním a příjmem PTP zpráv. Kružnice reprezentují místa pro pořizování časových značek. PTP zprávy a časové značky jsou předávány do funkce výpočtu offsetu (4).
4. *Funkce výpočtu offsetu* na základě předaných informací vypočte aktuální offset a další statistické informace. Z hlediska chování se jedná o zpětnovazební systém.



Obrázek 5.1: Schéma systému

5. *Funkce regulace* bere jako vstup vypočtený aktuální offset, který na základě stavových informací transformuje na hodnotu pro korekci vnitřních hodin systému.

### 5.3 Volba platformy

Pro referenční implementaci jsem si zvolil síťovou kartu *COMBOv2*, vytvořenou ve spolupráci fakult FIT VUT v Brně, FI Masarykovy univerzity a organizace CESNET<sup>2</sup> v rámci projektu Liberouter<sup>3</sup>. Díky vývojovému frameworku *NetCOPE*, vyvinutého pro síťovou kartu COMBOv2 stejnou skupinou organizací, je možno implementovat jak softwarovou, tak i hardwarová část aplikace, konkrétně firmware pro FPGA čip umístěný na kartě. To dává uživateli svobodu při návrhu řešení a podporuje myšlenku souběžného návrhu hardwaru a softwaru, který byl vybrán jako vhodný přístup při návrhu řešení.

### 5.4 Návrh pro kartu COMBOv2

Návrh systému pro tuto platformu se bude skládat ze tří částí:

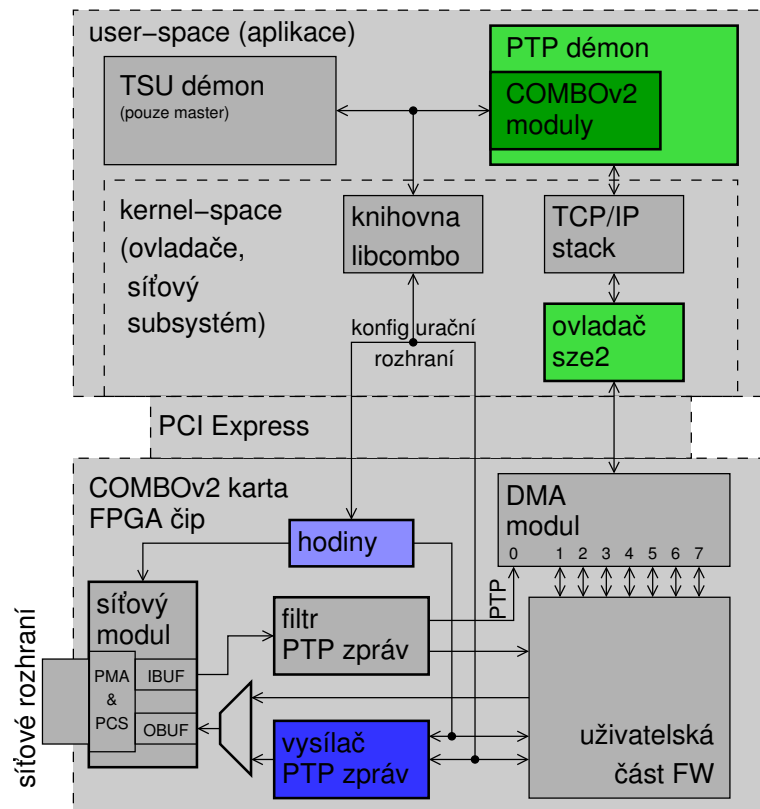
- *Firmwareové bloky pro FPGA čip:* (1) hodiny s funkcemi pro vlastní korekci, (2) vysílač PTP zpráv s podporou odchozích časových značek a (3) filtr PTP zpráv
- *Softwarová aplikace:* (1) zpracování PTP zpráv, (2) výpočet hodnot pro korekci hodin v FPGA čipu a (3) ovládání firmwareových bloků v FPGA čipu
- *Vrstva ovladačů pro operační systém:* přidání podpory hardwarových časových značek pro standardní linuxová rozhraní do ovladačů pro kartu COMBOv2

Schéma systému s vyznačením jednotlivých částí je zobrazeno na obrázku 5.2. Spodní část reprezentuje firmwareové bloky na FPGA čipu. *Síťové moduly* a *DMA moduly* jsou

<sup>2</sup><http://www.cesnet.cz/>

<sup>3</sup><http://www.liberouter.org/>

součástí frameworku NetCOPE. Síťové moduly přidělují příchozím paketům časové značky. *Hodiny* jsou také implementovány ve frameworku NetCOPE [18], bude ale potřeba modifikovat jejich funkci, aby bylo možné seřizovat je pomocí parametrů počítaných protokolem PTP. *Filtr PTP zpráv* slouží pro vyfiltrování PTP provozu, který bude dále zpracováván v softwaru. Do uživatelské části firmwaru tak PTP provoz vůbec nedorazí. V případě potřeby bude možné toto chování vypnout. *Vysílač PTP zpráv* slouží pro vysílání PTP zpráv s podporou pořizování časových značek. Důvod vytvoření takovéto komponenty je ten, že framework NetCOPE sám o sobě odchozí časové značky nepodporuje. Tato komponenta bude ovládána přes konfigurační rozhraní, včetně nahrávání dat samotné PTP zprávy. Důvod, proč není pro zaslání PTP zprávy použit DMA modul je ten, že by bylo obtížné asociovat odchozí časovou značku s danou PTP zprávou, jelikož rámec vyslaný z DMA modulu nemá jednoznačný identifikátor. Navržené řešení se samostatnou jednotkou nejprve načte PTP zprávu do bufferu, příkazem ji vyšle a odchozí časovou značku uloží do registru pro pozdější vyčtení.



Obrázek 5.2: Schéma systému na platformě NetCOPE. Modré bloky označují komponenty firmwaru, které je nutné vytvořit (tmavší odstín) nebo upravit (světlejší odstín). Zelené bloky vyznačují stejným způsobem softwarové části. Šedé bloky jsou implementovány frameworkem NetCOPE nebo uživatelem.

Framework NetCOPE zprostředkovává komunikaci přes *PCI Express sběrnici*. Na straně firmwaru se o komunikaci stará *DMA modul*, na straně operačního systému ovladač *sze2*. Ten poskytuje možnost síťová data zpřístupnit skrze systémový *TCP/IP stack*. Bude ale nutné přidat podporu pro hardwarové časové značky, jelikož klasické časové značky jsou generovány až samotným operačním systémem v softwaru.

Samotný proces zpracování PTP zpráv, provádění výpočtů nad extrahovanými daty, generování statistických informací a příkazů pro provádění samotné synchronizace, tzn. seřizování hodin, je prováděno v softwarové části, *PTP démonu*. Existuje několik volně dostupných implementací PTP démonů. Všechny však ve výchozí konfiguraci nepodporují hardwarové časové značky a neumí pracovat s komponentami na kartě COMBOv2. Příkladem takové implementace je aplikace *PTPd*<sup>4</sup>. Dosažená přesnost této ryze softwarové implementace se pohybuje v řádu desítek nebo i jednotek mikrosekund v závislosti na parametrech použitých filtrů [11], [12]. Další volně dostupná implementace *ptpv2d*<sup>5</sup> vychází z aplikace PTPd a přidává několik nových funkcí (podpora verze 2 protokolu PTP, podpora standardu IEEE 802.1AS).

---

<sup>4</sup><http://ptpd.sourceforge.net/>

<sup>5</sup><http://code.google.com/p/ptpv2d/>

# Kapitola 6

## Implementace

### 6.1 Firmwarové komponenty

V rámci této kapitoly bude popisován vývoj za použití frameworku NetCOPE. Ve firmwarové části se používají dvě typy sběrnic. Pro vysokorychlostní proudové přenosy mezi komponentami je to sběrnice *FrameLink*, která vychází ze sběrnice *LocalLink* používané v FPGA čípech firmy Xilinx. Druhou sběrnicí je *MI32* pro konfigurační, paměťové orientované přístupy. Více informací o obou sběrnicích je možné nalézt v [2].

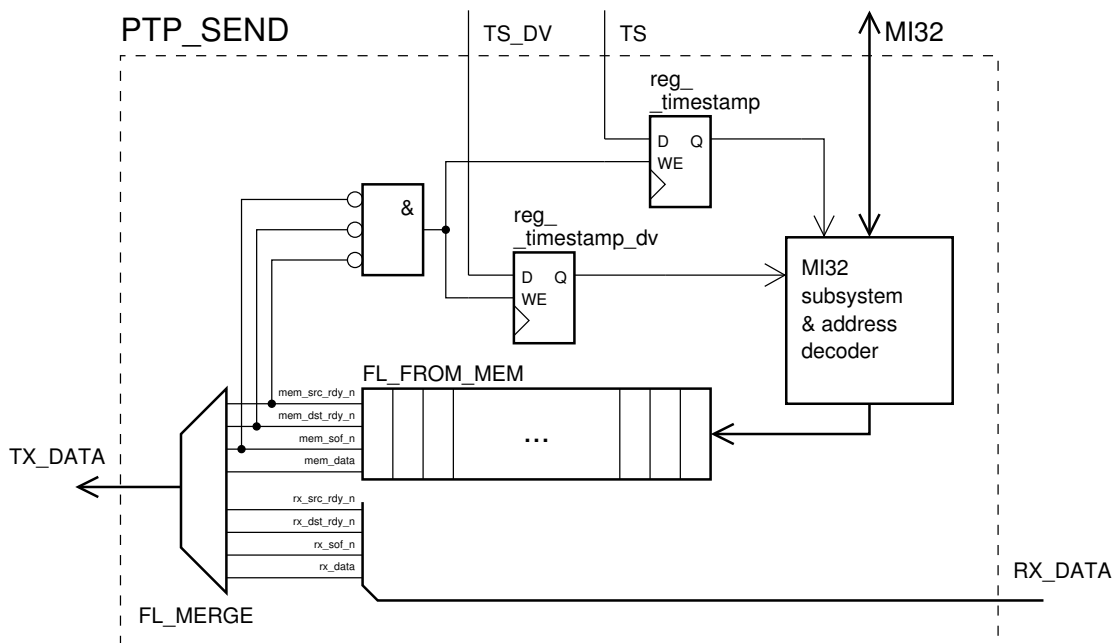
#### Komponenta PTP\_SEND

Kapitoly 5.1 a 5.4 popisují důvody pro vytvoření této komponenty. Pomocí ní je možné vkládat libovolná data do průchozí sběrnice *FrameLink*. V případě této aplikace je použita pro vkládání PTP paketů na sběrnici vedoucí do komponenty *OBUF* pro vysílání dat na síťové rozhraní. Toto využití určuje název komponenty *PTP\_SEND*.

Veškerá konfigurace probíhá přes rozhraní *MI32*. Jedná se o (1) nahrávání dat pro vysílání, (2) spuštění vysílání nahraných dat a (3) vyčtení odchozí časové značky. Dalším vstupem je signál s aktuální hodnotou hodin pro generování odchozích časových značek. Schéma komponenty je zobrazeno na obrázku 6.1.

První krok spočívá v nahrání dat pro vysílání. V případě Ethernetových sítí se jedná o Ethernetový rámec bez částí preamble, SFD (angl. zkratka start of frame delimiter), FCS (angl. zkratka frame check sequence) a mezipaketové mezery. Tato pole jsou automaticky doplněna komponentou *OBUF* a dalšími příslušnými částmi firmwaru a hardwaru. Nahrává se tedy rámec začínající cílovou a zdrojovou MAC adresou následovanou označením protokolu vyšší vrstvy a samotným obsahem rámce.

Jako příklad uveďme PTP paket Sync přenášený protokolem UDP. Na obrázku 6.2 je zobrazen rámec, který byl vyslán serverem Meinberg LANTIME M600 [5] umístěným v síti CESNET. Jednotlivé vrstvy L2 až L4 odpovídají linkové až aplikační vrstvě Internet modelu [10]. Vyznačena jsou jen důležitá pole z hlediska funkce protokolu PTP. Cílová adresu IP paketu je multicastová skupina 224.0.1.129, do které se zasílají PTP zprávy událostí, tedy ty, které při svém odeslání či příjmu vyžadují pořízení přesné časové značky. Na úrovni protokolu UDP jsou PTP zprávy událostí označeny portem s číslem 319. Pole typ protokolu PTP s hodnotou 0 označuje Sync zprávu. Pole *versionPTP* určuje verzi PTP protokolu. V tomto případě je to verze 2 podle IEEE 1588-2008. Délka určuje počet bajtů PTP zprávy včetně hlavičky. Hodnota 0200 příznaků v šestnáctkové soustavě určuje pouze jediný nastavený příznak, a to příznak *two-step* popsany v kapitole 4.2. Ten implikuje,



Obrázek 6.1: Schéma komponenty PTP\_SEND

<b>L2</b> Ethernet	Cílová MAC	Zdrojová MAC	Typ	Obsah rámce		
	01:00:5e:00:01:81	00:15:fa:87:31:00	0800	...		
<b>L3</b> IPv4	Verze	Protokol	Zdrojová IP	Cílová IP	Obsah paketu	
	4	...	17	195.113.147.10	224.0.1.129	
<b>L2</b> UDP	Zdrojový port	Cílový port	Délka	Kontrolní součet	Obsah datagramu	
	319	319	52	...	...	
<b>L4</b> PTP událost	Typ	Verze	Délka	Příznaky	časová značka [s]	časová značka [ns]
	0	2	44	0200	...	1397813862

Obrázek 6.2: Ukázková struktura PTP zprávy přenášené v UDP datagramu

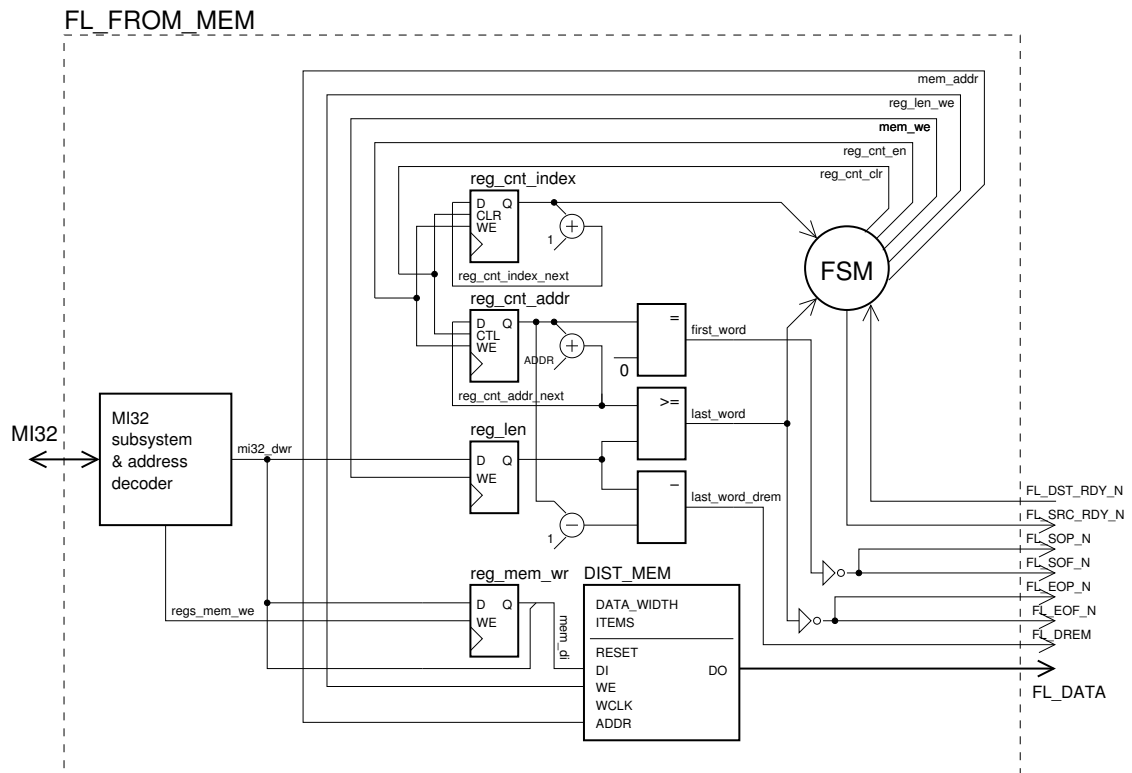
že časová značka obsažená v Sync zprávě nemá dostatečnou přesnost, nebo má nulovou hodnotu [4], a bude následována Follow\_Up zprávou obsahující přesnou hodnotu odchozí časové značky příslušné Sync zprávy.

Samotné nahrávání dat do komponenty PTP\_SEND a spuštění vysílání rámce na sběrnici FrameLink se provádí pomocí sběrnice MI32 a je prováděno podkomponentou FL\_FROM\_MEM. Ta je popsána dále v textu včetně konkrétního způsobu komunikace.

Časová značka je pořízena při začátku vysílání rámce na sběrnici FrameLink. Začátek vysílání je reprezentován logickým součinem signálů `mem_src_rdy_n`, `mem_dst_rdy_n` a `mem_sof_n`. Jelikož jsou uvedené signály aktivní v logické nule, jsou na vstupu logického součinu negovány. Výstup logického součinu je přiveden na vstup WE (write enable) registrů pro uchování časové značky (`reg_timestamp` a `reg_timestamp_dv`). Registr `reg_timestamp` je napojen na výstup TS (obsahuje aktuální časovou značku) jednotky TSU (řídí generování aktuálních časových značek). Registr `reg_timestamp_dv` je napojen na výstup TS\_DV jednotky TSU, který indikuje, zda je aktuální časová značka platná (TSU jednotka musí být synchronizována). Po ukončení vysílání rámce, které je detekovatelné z podkomponenty FL\_FROM\_MEM, je možné hodnoty těchto dvou registrů vyčíst a zjistit tak čas odeslání rámce z komponenty.

## Komponenta FL\_FROM\_MEM

Komponenta FL\_FROM\_MEM slouží k nahrání dat skrze sběrnici MI32 a jejich následnému vyslání sběrnici FrameLink. Schéma komponenty je zobrazeno na obrázku 6.3.

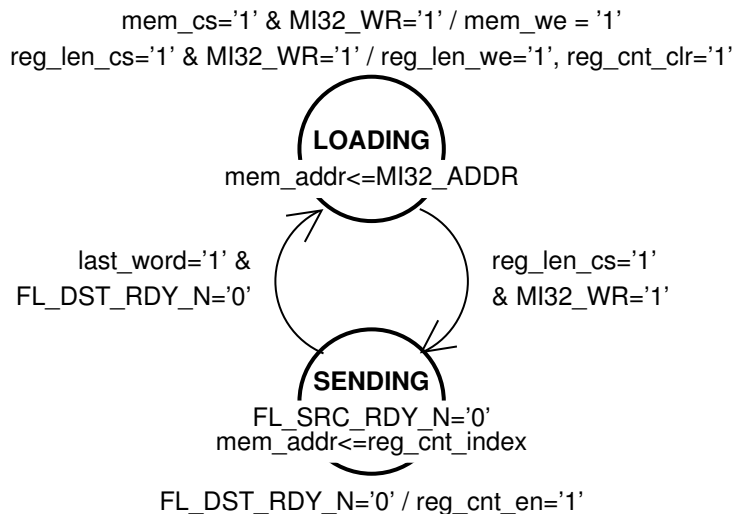


Obrázek 6.3: Schéma komponenty FL\_FROM\_MEM

Pro uchování dat je použita komponenta DISTMEM, která reprezentuje paměť na čipu realizovanou v logických buňkách. Šířka jedné položky této paměti je určena šířkou výstupní

sběrnice FrameLink, což je v případě této aplikace 128 bitů. Šířka sběrnice MI32 je 32 bitů a je tedy nutné nahrávat jednu položku paměti několikanásobným zápisem sběrnice MI32. K tomu slouží registry `reg_mem_wr`, do kterých se sběrnici MI32 zapisují jednotlivé části položky paměti. Zapsáním poslední části (s nejnižší adresou) je obsah registru `reg_mem_wr` zapsán do položky paměti. To implikuje způsob nahrávání dat do paměti. Ta se nahrávají od nejvyšší adresy po nejnižší. Registr `reg_cnt_index` uchovává index aktuálně nahrávané položky paměti. Registr `reg_cnt_addr` uchovává bajtovou adresu této položky. Index se při nahrávání/vysílání dekrementuje/inkrementuje o hodnotu 1, adresa o hodnotu 16 (datová šířka položky paměti v bajtech). Zahájení vysílání dat na sběrnici FrameLink a indikace ukončení vysílání je prováděna registrem `reg_len`. Po nahrání dat do paměti se do tohoto registru uloží přes sběrnici MI32 délka nahraných dat v bajtech a je spuštěno vysílání. Jak se postupně odesílají položky paměti přes sběrnici FrameLink, registr `reg_len` je dekrementován a při dosažení poslední položky obsahuje hodnotu 0, která indikuje dokončení vysílání. Poslední položku je nutné ošetřit speciálním způsobem, jelikož nemusí dosahovat celé šířky sběrnice. Počet bajtů posledního vysílaného slova je určen rozdílem hodnot registrů `reg_len` (délka nahraných dat) a `reg_cnt_addr` (délka odeslaných) dat.

Činnost komponenty je řízena konečným stavovým automatem. Jeho přechodový diagram je zobrazen na obrázku 6.4. Jedná se o Mealyho stavový automat, jelikož jeho výstupy závisí na aktuálním stavu a hodnotách vstupních signálů. Automat má dva stavy. Ve stavu `LOADING` probíhá nahrávání dat do paměti. Zápisem hodnoty do registru `reg_len` se provede přechod do stavu `SENDING` a je zahájeno vysílání dat. Jsou aktivovány čítače indexu a adresy položek paměti a data vyčítaná z paměti jsou vystavována na sběrnici FrameLink. Po dosažení poslední položky paměti je vysílání ukončeno a automat přejde do původního stavu `LOADING`, kdy je možné nahrát další data a spustit jejich vysílání.



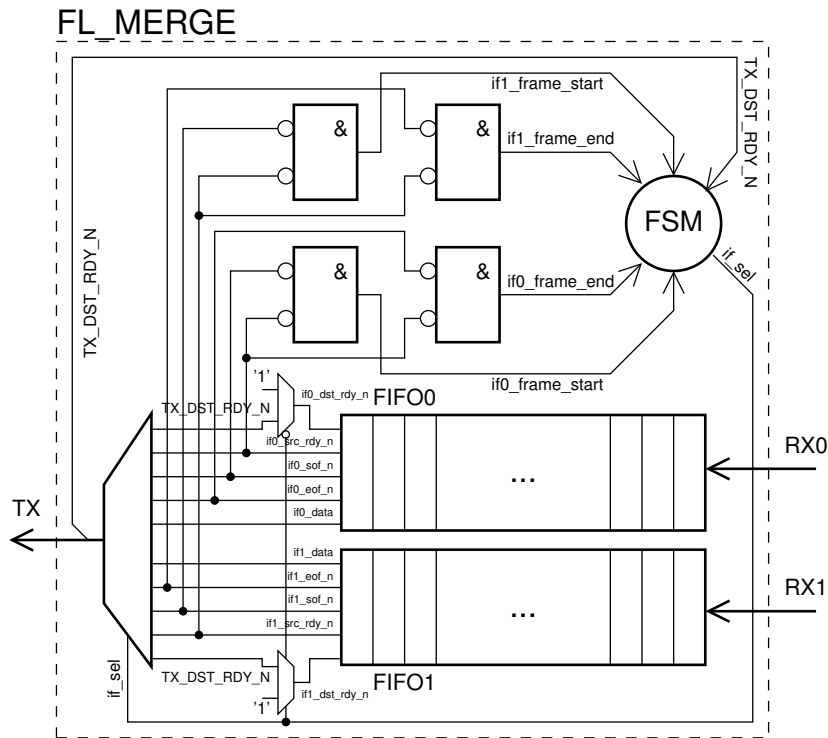
Obrázek 6.4: Přechodový diagram komponenty `FL_FROM_MEM`

## Komponenta `FL_MERGE`

Komponenta `FL_MERGE` slouží ke spojení dvou sběrnic FrameLink do jedné, přičemž jedna ze vstupních sběrnic má přednost před druhou. Pokud chce některá ze vstupních sběrnic začít přenášet data a výstupní sběrnice je volná, je přenos bez odkladu zahájen. Pokud

některá ze vstupních sběrnic právě přenáší data, je tento přenos dokončen. Pokud je výstupní sběrnice volná a obě vstupní sběrnice chtějí začít přenášet data najednou, má vždy přednost sběrnice první.

Schéma komponenty je zobrazeno na obrázku 6.5. Obě vstupní rozhraní mají možnost zapojení fronty proti případnému zahlcení cílové komponenty. Generickými parametry RX0\_BUF\_ITEMS a RX1\_BUF\_ITEMS je možné určit počet položek těchto front. Hodnotou 0 lze fronty vypustit.



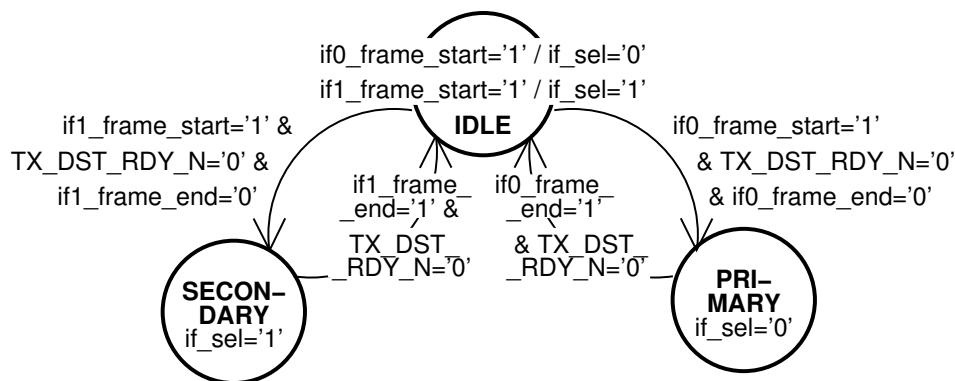
Obrázek 6.5: Schéma komponenty FL\_MERGE

Funkce komponenty je řízena konečným stavovým automatem, jehož přechodový diagram je zobrazen na obrázku 6.6. Jedná se o Mealyho automat, jelikož výstupy automatu závisí na aktuálním stavu a hodnotách vstupních signálů. Výchozím stavem je IDLE, kdy žádná ze vstupních sběrnic nepřenáší data na výstup. Jakmile chce některá ze vstupních sběrnic zahájit přenos dat, automat přejde do stavu PRIMARY nebo SECONDARY. Pořadí vyhodnocení podmínek žádostí o přenos obou vstupních sběrnic upřednostňuje vždy stejnou vstupní sběrnici. V této aplikaci je to využito pro prioritní vkládání PTP zpráv v případě, že by byla druhá sběrnice neustále plně vytížena.

### Hodiny s podporou korekce

Komponenta hodin s podporou korekce je založena na existující komponentě vyvinuté pro kartu COMBOv2, která má na vstupu signál PPS (pulse per second) jehož typickým zdrojem je GPS přijímač. [18]

Důvod použití již existující komponenty je sdílení mnoha společných vlastností. Hlavním rozdílem je absence signálu PPS, který je nahrazen synchronizačními zprávami protokolu PTP. Z nich je potřeba získat informaci použitelnou pro korekci hodin. Tato činnost je vy-



Obrázek 6.6: Přechodový diagram komponenty FL\_MERGE

konávána zčásti v čipu FPGA a zčásti v softwarové aplikaci. V čipu FPGA jsou pořizovány přesné časové značky, na jejichž základě je v softwarové aplikaci vypočtena hodnota pro korekci hodin, která je předána zpět do čipu FPGA.

## Komponenta TSU

Hlavním úkolem komponenty TSU je poskytnout podkomponentě TSU\_CV2\_CORE signály a informace nutné pro chod hodin a jejich korekci. Komponenta TSU provádí detekci zdroje PPS signálu (komponenta může být použita se speciální přídavnou kartou COMBOPTM nebo přímo s kartou COMBOv2 [18]) a vytváří asynchronní přechod mezi dvěma hodinovými doménami. Zatímco jádro komponenty TSU pracuje na frekvenci 166 MHz, komunikační subsystém pracuje na frekvenci aplikačního jádra, v tomto případě 175 MHz.<sup>1</sup> Komponenta TSU dále provádí konverzi desetinné části sekund na počet nanosekund. V této komponentě nebylo třeba provádět žádné změny. Všechna funkcionalita vyžadovaná pro činnost protokolu PTP byla implementována podkomponentou TSU\_CV2\_CORE.

## Komponenta TSU\_CV2\_CORE

Nejprve budou popsány základní bloky komponenty TSU\_CV2\_CORE, které nebylo třeba upravovat, protože pochopení jejich významu je klíčové pro popis změn provedených v této komponentě.

Komponenta TSU\_CV2\_CORE obsahuje samotné hodiny a prostředky pro jejich korekci. Jádrem komponenty jsou dva registry. Registr `reg_realtime` uchovává počet sekund od počátku epochy (používá Unixový/POSIXový čas, tedy UTC čas s počátkem 1.1.1970). Jedná se o desetinné číslo s pevnou řádovou čárkou, kdy 32 bitů reprezentuje celou část a 64 bitů reprezentuje desetinnou část. Jelikož je zvykem uvádět necelou část sekund ve formě počtu nanosekund, je potřeba desetinnou část násobit hodnotou  $10^9$ . Tato operace je implementována v nadřazené komponentě TSU pomocí násobiček DSP bloků FPGA čipu.

Druhý klíčový registr `reg_increment` určuje hodnotu, o kterou se zvětší hodnota registru `reg_realtime` každou periodu řídicího hodinového signálu. Rozsah tohoto registru je 64 bitů a uchovává pouze desetinnou část sekundy. Důvodem je přípustný rozsah, který při frekvenci 166 MHz nikdy nepřesáhne 1 s. Kvůli nepřesnostem řídicího hodinového signálu způsobenou nestálostmi externího oscilátoru je potřeba hodiny korigovat prostřednictvím

<sup>1</sup>Tyto frekvence jsou určeny externím oscilátorem spolu s konfigurací komponent subsystému pro generování hodinových signálů (komponenta CLK\_GEN\_CV2). [2]

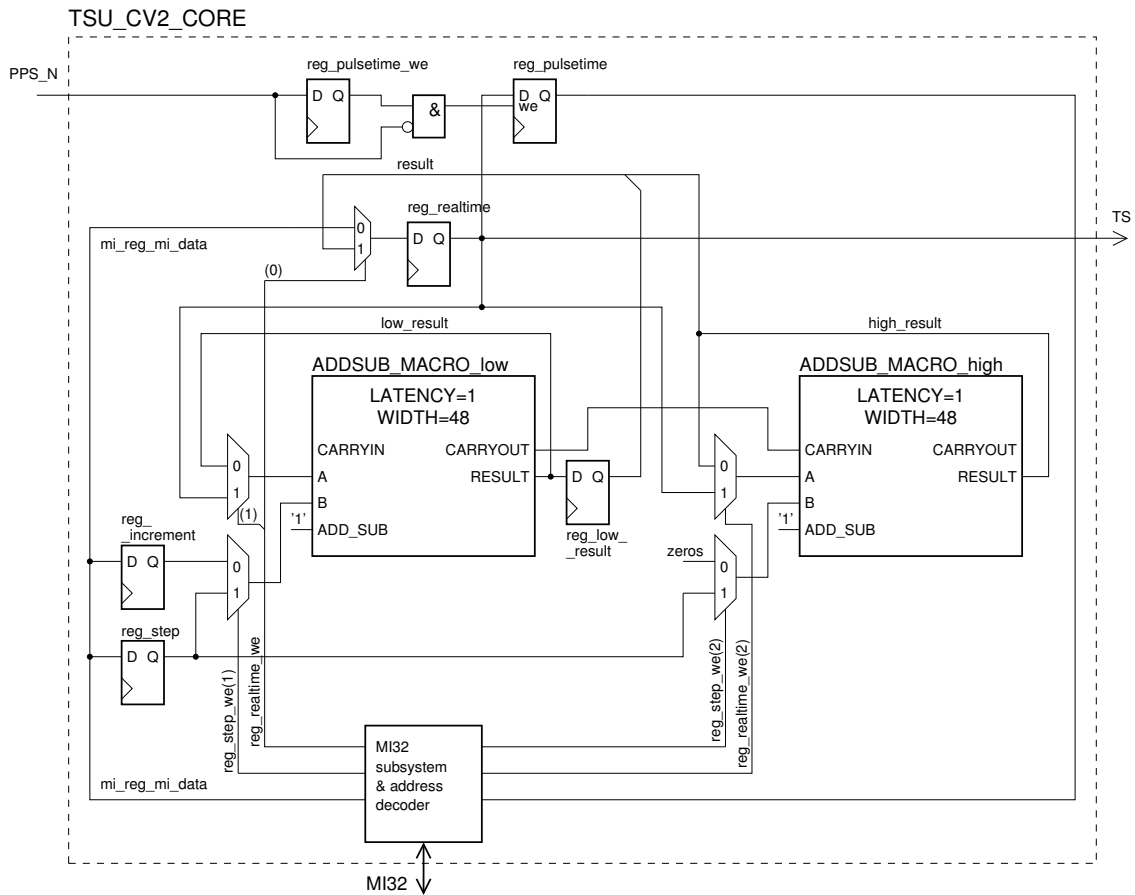
úpravy hodnot zmíněných registrů. Hodnota registru `reg_increment` vychází z rovnic 6.1. Proměnná `ns_per_tick` vyjadřuje počet nanosekund, které uběhnou za jednu periodu hodinového signálu. Tento počet nemusí být celé číslo. Proměnná `shift64` vyjadřuje tuto hodnotu jako desetinnou část sekundy se šířkou 64 bitů. Výsledná hodnota registru je vyjádřena proměnnou `reg_increment`.

$$\begin{aligned} ns\_per\_tick &= 10^9 \cdot T_{TSU} = 10^9 / f_{TSU} \\ shift64 &= ns\_per\_tick / 10^9 \cdot 2^{64} \\ reg\_increment &= shift64 = 2^{64} / f_{TSU} \end{aligned} \quad (6.1)$$

Při frekvenci 166 MHz je počáteční hodnota registru `reg_increment` dána rovnicí 6.2.

$$\begin{aligned} reg\_increment &= 2^{64} / 1,66 \cdot 10^6 = (111124964299,455130217)_{10} \\ &\sim (00000019\ c511ddf4)_{16} \end{aligned} \quad (6.2)$$

Schéma modifikované komponenty pro protokol PTP je zobrazeno na obrázku 6.7.



Obrázek 6.7: Schéma komponenty TSU\_CV2\_CORE

Výpočet nové hodnoty registru `reg_realtime` je prováděn sčítačkami zaobalenými komponentami `ADDSUB_MACRO`. Ty sčítají dva svoje vstupy A a B a poskytují výsledek spolu s informací o vzniklém přenosu. Jelikož je šířka registru `reg_realtime` 96 bitů, je potřeba použít dvě instance této komponenty zřetěžené za sebou. Druhá instance přitom musí čekat na přenos z první, aby byl výsledek korektní. Jelikož je zpoždění komponenty jeden hodinový cyklus, je potřeba výstup první instance uložit do registru (`reg_low_result`), aby byl

další hodinový cyklus platný spolu s výsledkem druhé instance komponenty (`high_result`). Sloučením těchto signálů je potom dána nová hodnota registru `reg_realttime`.

V dalším textu jsou popsány změny provedené v komponentě `TSU_CV2_CORE`. Jedná se především o přidání registru a logiky pro podporu krokovacího režimu. Bylo třeba upravit adresový dekodér a řídicí logiku ovládající multiplexory na vstupech sčítaček.

V případě normálního chodu komponenty je pomocí řídicích signálů ovládajících multiplexory vstupů sčítaček (`reg_realttime_we` a `reg_step_we`) vybrána předchozí vypočítaná hodnota pro vstupy A a výstup registru `reg_increment` pro vstupy B. Hodnotu registru `reg_increment` je možné transparentně upravovat za chodu komponenty, není třeba pozastavovat její činnost.

Dalším případem je nastavení hodnoty registru `reg_realttime` na určitou požadovanou hodnotu, např. při inicializaci komponenty. Softwarová aplikace nejprve zapíše korektní hodnotu do registru `reg_increment`. Poté zapíše hodnotu do registru `reg_realttime` a pomocí řídicích signálů ovládajících multiplexory vstupů sčítaček je pro vstupy A sčítaček zvolen výstup registru `reg_realttime` a pro vstupy B výstup registru `reg_increment`. Tím se komponenta inicializuje na daný čas zvětšený o čas jednoho cyklu hodin.

Posledním případem je krokování hodin, kdy se hodnota registru `reg_realttime` jednorázově zvětší o určitou požadovanou hodnotu. Tu nejprve softwarová aplikace uloží do registru `reg_step` a řídicími signály ovládajícími multiplexory vstupů sčítaček je vybrán výstup registru `reg_realttime` pro vstupy A a výstup registru `reg_step` pro vstupy B.

## 6.2 Softwarové moduly a aplikace

Klíčovou částí softwarové aplikace je zpracování zpráv protokolu PTP podle standardu IEEE 1588-2008. Pro tento účel byla zvolena volně dostupná implementace `ptpv2d` stručně popsaná v kapitole 5.4. Využívá standardní linuxová síťová rozhraní s nanosekundovými časovými značkami generovanými operačním systémem. Bylo proto nutné aplikaci upravit takovým způsobem, aby mohla využít časové značky generované kartou `COMBOv2`. Za tímto účelem byly upraveny ovladače karty `COMBOv2` (pro příchozí PTP zprávy) a vytvořen vysílač PTP zpráv (pro odchozí PTP zprávy). Dalším softwarovým modulem je modul pro inicializaci karty `COMBOv2` a modul pro řízení `TSU` jednotky s podporou protokolu PTP.

V následujícím seznamu jsou uvedeny jednotlivé části softwarové aplikace, které musely být vytvořeny nebo upraveny. Tyto části a provedené změny jsou blíže popsány v následujících kapitolách.

- *Přidání podpory hardwarových časových značek do ovladače karty `COMBOv2`*: Jedná se o extrakci hardwarové časové značky z hlavičky přidružené k příchozímu paketu a její transformace do struktur rozhraní systému Linux pro práci s hardwarovými časovými značkami.
- *Vytvoření modulů pro práci s firmwarovými komponentami*: Moduly slouží pro ovládání karty `COMBOv2` a ovládání vytvořených firmwarových komponent. Při implementaci byly použity funkce knihovny `libcombo` frameworku `NetCOPE`. Více informací o funkcích poskytovaných touto knihovnou je možné nalézt v [2].
- *Úprava aplikace `ptpv2d`*: Využití hardwarových časových značek generovaných na kartě `COMBOv2` vyžaduje upravit aplikaci `ptpv2d` tak, aby využívala rozhraní sys-

tému Linux pro vyčítání příchozích hardwarových časových značek a rozhraní vytvořených modulů pro práci s firmwarovými komponentami.

## Podpora hardwarových časových značek v ovladači karty COMBOv2

Ovladače karty COMBOv2 jsou napsány pro operační systém Linux. Ten implementuje podporu časových značek příchozích paketů pomocí dvou rozhraní: `SO_TIMESTAMP` a `SO_TIMESTAMPNS`. První rozhraní pracuje s přesností mikrosekund, druhé s přesností nanosekund. Obě jsou však založena na přiřazování časových značek v softwaru, což není vhodné z hlediska přesnosti. Od verze 2.6.30 linuxového jádra je k dispozici nové rozhraní `SO_TIMESTAMPING` [14], které podporuje jak softwarové, tak i hardwarové časové značky a stejně tak časové značky příchozích i odchozích paketů.

Rozhraní je definováno v hlavičkovém souboru `linux/net_tstamp.h`. [1] V něm jsou definovány datové typy a konstanty pro práci s přesnými časovými značkami. Způsob generování přesných časových značek je určen systémovým voláním `setsockopt`s s parametrem `SO_TIMESTAMPING`. Další parametry tohoto systémového volání určují požadovaný režim pořizování časových značek. Hodnoty těchto parametrů jsou shrnuty v tabulce 6.1. Tučně jsou zvýrazněné příznaky, které se v obslužném programu využívající upravené ovladače používají.

<i>Parametr</i>	<i>Popis</i>
<code>SO_TIMESTAMPING_</code>	
<code>_TX_HARDWARE</code>	Odchozí časové značky v hardwaru
<code>_TX_SOFTWARE</code>	Odchozí časové značky v softwaru
<b><code>_RX_HARDWARE</code></b>	<b>Příchozí časové značky v hardwaru</b>
<code>_RX_SOFTWARE</code>	Příchozí časové značky v softwaru
<b><code>_RAW_HARDWARE</code></b>	<b>Původní časová značka pořízená v hardwaru</b>
<b><code>_SYS_HARDWARE</code></b>	<b>Časová značka převedená do systémového času</b>
<code>_SOFTWARE</code>	Časová značka vygenerovaná v softwaru

Tabulka 6.1: Režim pořizování časových značek rozhraní `SO_TIMESTAMPING`

Generování přesných časových značek musí být aktivováno v každém síťovém zařízení zvlášť pomocí systémového volání `ioctl` s parametrem `SIOCSHWTSTAMP` a strukturou určující další vlastnosti generování přesných časových značek. Tyto vlastnosti jsou shrnuty v tabulce 6.2. Tučně jsou zvýrazněné příznaky, které se v obslužném programu využívající upravené ovladače používají. Důvod nevyužití podpory přesných odchozích časových značek rozhraní `SO_TIMESTAMPING` je uveden v kapitole 5.4.

<i>Parametr HWTSTAMP_</i>	<i>Popis</i>
<b>_TX_OFF</b>	<b>Časové značky nebudou generovány pro odchozí pakety</b>
_TX_ON	Časové značky budou generovány pro vybrané pakety
_TX_ONESTEP_SYNC	Časové značky budou generovány jako s příznakem HWTSTAMP_TX_ON a navíc budou pro PTP Sync zprávy vkládány přímo do paketu, jak je popsáno v kapitole 4.2 o příznaku two-step
<i>Parametr HWTSTAMP_</i>	<i>Časové značky budou generovány pro...</i>
_FILTER_NONE	žádné příchozí pakety
<b>_FILTER_ALL</b>	<b>všechny příchozí pakety</b>
_FILTER_SOME	vyžádané příchozí pakety a některé další
_FILTER_PTP_V1_L4_EVENT / SYNC / DELAY_REQ	PTP v1 UDP zprávy událostí / pouze Sync zprávy / pouze DelayReq zprávy
_FILTER_PTP_V2_L4_EVENT / SYNC / DELAY_REQ	PTP v2 UDP zprávy událostí / pouze Sync zprávy / pouze DelayReq zprávy
_FILTER_PTP_V2_L2_EVENT / SYNC / DELAY_REQ	PTP v2 Ethernet/802.1AS <sup>1</sup> zprávy událostí / pouze Sync zprávy / pouze DelayReq zprávy
_FILTER_PTP_V2_EVENT / SYNC / DELAY_REQ	PTP v2 zprávy událostí / pouze Sync zprávy / pouze DelayReq zprávy (na rozdíl od předchozích dvou pro jakýkoli typ PTP v2 zpráv)

<sup>1</sup>Standard 802.1AS definuje profil protokolu IEEE 1588-2008 pro použití v heterogenních lokálních sítích (Ethernet, Wi-Fi...) [19]

Tabulka 6.2: Volby parametru SIOCSHWTSTAMP systémového volání ioctl

## Modul pro inicializaci karty COMBOv2

Zdrojové kódy tohoto modulu jsou umístěny v podsložce `dep/` PTP aplikace, jelikož jsou závislé na použité platformě. Jedná se o hlavičkový soubor `combov2.h` a implementační soubor `combov2.c`. Seznam funkcí modulu s krátkým popisem je uveden v tabulce 6.3.

<i>Veřejné funkce</i>	<i>Popis</i>
<code>comboSetVerbosity</code>	Nastavuje podrobnost výpisů do logovacího souboru
<code>comboDeviceInit</code>	Inicializuje kartu COMBOv2 podle parametrů příkazové řádky: 1. Načte seznam komponent z konfiguračního XML souboru 2. Namapuje komponenty TSU a PTP_SEND 3. Inicializuje komponentu TSU
<code>comboDeviceFree</code>	Uvolní paměť alokovanou při inicializaci karty COMBOv2

Tabulka 6.3: Funkce modulu pro inicializaci karty COMBOv2

## Modul pro práci s komponentou PTP\_SEND

Zdrojové kódy modulu se nacházejí v hlavičkovém souboru `combov2-ptp.h` a implementačním souboru `combov2-ptp.c` v podsložce `dep/`. Seznam funkcí s popisem je uveden v tabulce 6.4.

<i>Veřejné funkce</i>	<i>Popis</i>
<code>comboMsgPackRawHeader</code>	Vytváří Ethernetovou hlavičku pro PTP paket. Jedná se tedy o přenos PTP paketu na síťové vrstvě bez přítomnosti IP a UDP hlavičky. (Dodatek F standardu [4], např. pro IEEE 802.1AS)
<code>comboMsgPackHeader</code>	Vytváří IPv4 a UDP hlavičku, bez kontrolních součtů
<code>comboMsgPackFinish</code>	Doplňuje kontrolní součty do IPv4 a UDP hlaviček
<code>comboNetSendEventTimestamp</code>	Vysílá PTP zprávu, provádí konverzi parametrů pro privátní funkci <code>comboPtpSend</code>
<i>Soukromé funkce</i>	<i>Popis</i>
<code>rfc1071_ipv4_checksum</code>	Provádí výpočet kontrolního součtu IPv4 paketu podle RFC1071
<code>rfc768_udp_checksum</code>	Provádí výpočet kontrolního součtu UDP datagramu podle RFC768
<code>comboPtpSend</code>	Vysílá PTP zprávu

Tabulka 6.4: Funkce modulu pro práci s komponentou PTP\_SEND

Funkce `comboPtpSend` komunikuje přes sběrnici MI32 s komponentou PTP\_SEND. Provádí následující akce:

1. Nahrání dat do paměti komponenty PTP\_SEND
2. Zápis délky nahraných dat, což způsobí zahájení vysílání dat na sběrnici a následně do sítě
3. Čekání na dokončení vysílání, to se děje vyčítáním hodnoty registru délky nahraných dat, dokud nemá hodnotu 0
4. Vyčtení registrů s časovou značkou odeslání dat

## Modul pro práci s komponentou TSU

Zdrojové kódy modulu se nacházejí v hlavičkovém souboru *combov2-tsu.h* a implementačním souboru *combov2-tsu.c* v podsložce *dep/*. Seznam funkcí s popisem je uveden v tabulce 6.5.

Funkce `comboTsuInit` nejprve vyčte frekvenci, na které pracuje komponenta TSU. Ta je uložena v registru této komponenty. Poté nastaví výchozí hodnotu registru `reg_increment` podle rovnice 6.2.

Funkce `comboTsuSetTime` a `comboTsuGetTime` přijímají jako parametr reprezentaci času v sekundách a nanosekundách, kterou je třeba konvertovat na sekundy s desetinnou částí. To je provedeno funkcemi `nsec_to_frac` a `frac_to_nsec`. Dále je potřeba zkonvertovat systémový čas UTC na čas TAI používaný protokolem PTP, což je popsáno v kapitole 2.1. Protokol PTP pro tuto konverzi zavádí pole *currentUtcOffset* zprávy `Announce`.

Funkce `comboTsuApplyOffset` nejprve zkontroluje korektnost offsetu funkcí `checkOffset`, offset normalizuje funkcí `normalizeOffset` a nahraje ho do registru `reg_step`, což způsobí přičtení offsetu k aktuálnímu času komponenty TSU.

Funkce `comboTsuSetFreqAdj` vypočítá novou hodnotu pro inkrementační registr na základě zadané hodnoty korekce podle rovnice 6.3. Proměnná *adj* vyjadřuje hodnotu korekce.

$$reg\_increment = nsec\_to\_frac(adj / f_{TSU}) \quad (6.3)$$

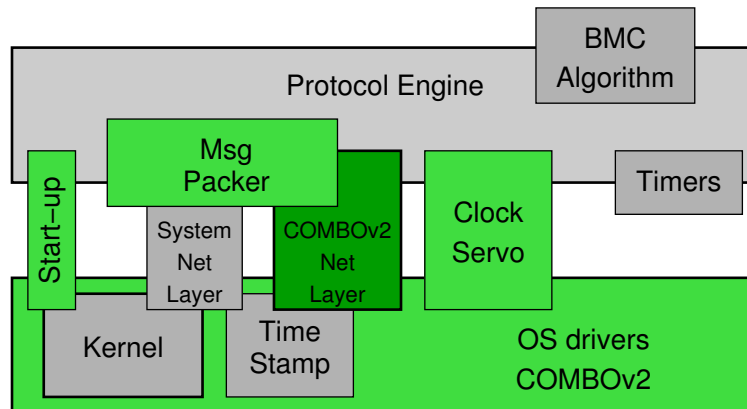
<i>Veřejné funkce</i>	<i>Popis</i>
comboTsuInit	Inicializuje komponentu TSU
comboTsuSetTime	Nastaví komponentu TSU na požadovaný čas
comboTsuGetTime	Zjistí z komponenty TSU aktuální čas
comboTsuResetIncrementReg	Nastaví inkrementační registr na výchozí hodnotu vypočtenou při inicializaci komponenty TSU
comboTsuApplyOffset	Upraví čas komponenty TSU o požadovaný offset
comboTsuSetFreqAdj	Upraví hodnotu inkrementačního registru na základě zadané hodnoty pro korekci
comboTsuValidate	Označí generované časové značky jako platné
comboTsuInvalidate	Označí generované časové značky jako neplatné
<i>Soukromé funkce</i>	<i>Popis</i>
nsec_to_frac	Převádí počet nanosekund (desetinné číslo s plovoucí řádovou čárkou) na desetinnou část sekundy uloženou na 64 bitech
frac_to_nsec	Převádí desetinnou část sekundy uloženou na 64 bitech na počet nanosekund (desetinné číslo s plovoucí řádovou čárkou)
comboSetRealtimeReg	Nastaví registr <code>reg_realtime</code> na požadovanou hodnotu (32 bitů sekundy a 64 bitů desetinné části sekundy)
comboGetRealtimeReg	Vyčte hodnotu registru <code>reg_realtime</code> (32 bitů sekundy a 64 bitů desetinné části sekundy)
comboSetIncrementReg	Nastaví registr <code>reg_increment</code> na požadovanou hodnotu (64 bitů desetinné části sekundy)
comboGetIncrementReg	Vyčte hodnotu registru <code>reg_increment</code> (64 bitů desetinné části sekundy)
checkOffset	Zkontroluje, zda je zadaná reprezentace času platná (počet sekund a nanosekund musí mít stejné znaménko)
negativeOffset	Zkontroluje, zda reprezentace času vyjadřuje záporný čas
normalizeOffset	Upravuje reprezentaci času v případě, že je počet nanosekund sekund záporného času kladný ( $-10\text{ s}:1\text{ ns} \rightarrow -9\text{ s}:999\,999\,999\text{ ns}$ ) nebo je počet nanosekund větší než $10^9$ ( $10\text{ s}:1\,000\,000\,000\text{ ns} \rightarrow 11\text{ s}:0\text{ ns}$ )

Tabulka 6.5: Funkce modulu pro práci s komponentou TSU

## Modifikace aplikace ptpv2d

### Modulární struktura aplikace

Na obrázku 6.8 je zobrazeno schéma upravené aplikace zobrazující jednotlivé subsystémy. Porovnáním s původním schématem uvedeným v [7] lze určit rozsah provedených změn. Ty jsou na obrázku vyznačeny zelenou barvou a týkají se všech tří subsystémů zprostředkujících komunikaci mezi aplikací samotnou a jádrem operačního systému:



Obrázek 6.8: Schéma modulární struktury upravené aplikace ptpv2d, převzato z [7] a upraveno. Zeleně vyznačené bloky označují části aplikace, které bylo třeba vytvořit (tmavší odstín) nebo upravit (světlejší odstín).

1. *Start-up*: inicializační funkce (alokace paměti, inicializace komponent)
2. *Net Layer*: síťová komunikace
  - (a) *Time Stamp*: generování časových značek
3. *Clock Servo*: výpočet parametrů pro korekci hodin

### Změny v modulární struktuře aplikace

V případě subsystémů *Time Stamp* a *Clock Servo* byla interakce aplikace s jádrem kompletně nahrazena komunikací s ovladači karty *COMBOv2* a s kartou samotnou.

V případě časových značek pro příchozí PTP zprávy (subsystém *System Net Layer*) se vyčítání provádí standardními linuxovými knihovními funkcemi. Za účelem vyčtení hardwarové časové značky je použita funkce *recvmsg*, která umožňuje aplikaci předat dodatečná data o přijatém paketu ve struktuře typu *struct msghdr*. Každá taková předaná informace je přiřazena na určitou úroveň a je jí přiřazen určitý typ. V případě hardwarových časových značek se jedná o úroveň *SOL\_SOCKET* a typ *SCM\_TIMESTAMPING*. Jsou předány celkem tři časové značky ve strukturách datového typu *struct timespec*. Jedná se o neupravenou časovou značku vyčtenou z hardwaru a časové značky konvertované do systémového času. Jelikož *TSU* jednotka poskytuje časové značky ve formě systémového času, jsou všechny tři předané časové značky stejné.

Časové značky pro odchozí PTP zprávy jsou generovány subsystémem *COMBOv2 Net Layer*. Pro vysílání PTP zpráv a vyčítání jejich odchozích časových značek využívá modul

pro práci s komponentou PTP\_SEND popsaný v kapitole 6.2. Na rozdíl od subsystému System Net Layer, který obsahuje několik funkcí pro odesílání PTP zpráv odlišujících obecné zprávy, zprávy událostí a zprávy na linkové vrstvě (netSendEvent, netSendGeneral, netSendRaw), subsystém COMBOv2 Net Layer obsahuje pouze jednu funkci, *comboNetSendEventTimestamp*, která pracuje se všemi typy PTP zpráv.

Další změny se týkají subsystému Msg Packer, který slouží k vytváření PTP zpráv. Použitím komponenty PTP\_SEND se obchází standardní linuxový síťový subsystém, který se stará o vytváření IP hlaviček na síťové vrstvě a UDP hlaviček na transportní vrstvě. Tuto činnost je proto třeba vykonávat přímo v aplikaci. Jsou k tomu využity funkce modulu pro práci s komponentou PTP\_SEND, konkrétně *comboMsgPackRawHeader*, *comboMsgPackHeader* a *comboMsgPackFinish*. První ze jmenovaných funkcí slouží pro vytvoření Ethernetové hlavičky v případě přenosu PTP zpráv na síťové vrstvě. Funkce *comboMsgPackHeader* se použije pro vytvoření IPv4 a UDP hlavičky. Pole obsahující kontrolní součty zůstanou prázdná, jelikož ještě není znám obsah datagramu, což je PTP zpráva vytvořená funkcemi subsystému Msg Packer, konkrétně *msgPackV2Header* pro vytvoření hlavičky PTP zprávy, *msgPackAnnounce*, *msgPackV2Sync*, *msgPackV2FollowUp*, *msgPackV2PDelayReq*, *msgPackV2DelayResp* a *msgPackV2PDelayRespFollowUp* pro vytvoření obsahu konkrétního typu PTP zprávy.

### Přidané parametry příkazové řádky

Pro potřeby modulů pro práci s kartou COMBOv2 bylo přidáno několik parametrů příkazové řádky. Přidané parametry jsou shrnuty v tabulce 6.6.

<i>Parametr</i>	<i>Význam</i>	<i>Výchozí hodnota</i>
-B <bázová-adresa>	Bázová adresa pro mapování komponent karty COMBOv2	–
-C <zařízení>	Cesta ke speciálnímu souboru zařízení karty COMBOv2	/dev/combosix/0
-I <PTP-index>	Index jednotky PTP_SEND	0
-J <TSU-index>	Index jednotky TSU	0
-X <design-xml>	Cesta k XML souboru s mapováním komponent karty COMBOv2 do adresního prostoru	–
-Z (sys ptp)	Způsob inicializace hodin komponenty TSU	bez resetu

Tabulka 6.6: Přidané parametry příkazové řádky

Pokud je ve firmwaru karty COMBOv2 přítomno více jednotek PTP\_SEND, resp. TSU, parametr -I, resp. -J určí, se kterou bude aplikace pracovat. Paramter -C se použije v případě, že je v systému přítomno více karet COMBOv2. Každá karta je v systému reprezentována speciální souborem ve složce /dev/combosix/ pojmenovaným pořadovým číslem karty. Pokud není zadán parametr -Z, hodiny nejsou při inicializaci resetovány a pracují s časem, který byl v jednotce TSU před spuštěním aplikace.

### Resetování hodin komponenty TSU

Aplikace nabízí dvě možnosti resetování hodin komponenty TSU po spuštění aplikace. První možností je využít systémový čas. Pokud je systémový čas korigován protokolem NTP, po-

skytuje dostatečně přesný výchozí bod pro korekci hodin protokolem PTP (odchylka od UTC typicky nižší než 1 s). Druhou možností je nastavení hodin komponenty TSU podle časové značky uložené v první PTP zprávě Sync, resp. odpovídající zprávě Follow\_Up (pokud byl ve zprávě Sync nastaven příznak two-step), přijaté po spuštění aplikace.

### Možnosti korekce hodin

Korekce hodin se provádí v subsystému Clock Servo na několika místech. Nejprve budou popsány možnosti korekce hodin a poté případy, kdy jsou které možnosti využívány.

Existují dvě možnosti korekce hodin. První je *změna hodnoty registru `reg_increment`*. Pokud se frekvence řídicího hodinového signálu sníží (např. vlivem vnějšího prostředí), bude hodnota registru `reg_increment` zvýšena, aby se odchylka kompenzovala. V opačném případě se hodnota registru `reg_increment` sníží. Druhou možností je *krokování*, kdy se hodnota registru `reg_realtime` zvyšuje či snižuje krokově. Tento přístup ale může způsobit velmi rozdílné časové značky dvou událostí, které se v čase vyskytly velmi blízko sebe. Typicky se používá v případech, kdy je odchylka hodin příliš velká pro použití první možnosti. Pro podporu druhé možnosti je v komponentě TSU zaveden registr `reg_step`. Po zápisu do tohoto registru je jeho hodnota přičtena k hodnotě registru `reg_realtime`. Je možné použít i kombinaci obou možností.

Korekce hodin se provádí po přijetí Sync zprávy, resp. Follow\_Up zprávy ve funkci `updateClock`. V případě, že je zjištěný offset větší než 1 s, provede se jedna z následujících možností:

1. Pokud to není zakázáno parametrem příkazové řádky, provede se resetování hodin, v případě komponenty TSU provedené krokováním času
2. Provede se přizpůsobení hodnoty registru `reg_increment`
  - (a) Pokud je resetování hodin zakázáno, použije se maximální povolená hodnota korekce
  - (b) Pokud není resetování hodin zakázáno, hodnota korekce se vypočítá na základě časových značek dvou posledních přijatých Sync zpráv, resp. Follow\_Up zpráv podle vztahu 6.4.  $\Delta t_{TX}$ , resp.  $\Delta t_{RX}$ , je rozdíl odchozích, resp. příchozích, časových značek poslední a předposlední přijaté PTP zprávy Sync. Příklad výpočtu hodnoty korekce v tomto případě je uveden v příloze B.

$$adj = \frac{\Delta t_{TX} - \Delta t_{RX}}{\Delta t_{RX}} \quad (6.4)$$

V případě, že je offset menší než 1 s, provede se přizpůsobení hodnoty registru `reg_increment` na základě výstupu PI regulátoru.

## Kapitola 7

# Ověření funkčnosti a testování

V průběhu návrhu aplikace a její implementace bylo třeba řešit způsob, jakým se bude ověřovat její funkčnost. Samostatně byly testovány vyvinuté či upravené firmwarové jednotky, upravené ovladače karty COMBOv2 pro operační systém a softwarové moduly, včetně upravené aplikace PTP démona. Dalším krokem bylo otestovat interakci mezi jednotlivými částmi, tzn. mezi firmwarovými bloky a příslušnými obslužnými softwarovými moduly, firmwarovými bloky a ovladači pro operační systém a softwarovými moduly a hlavní aplikací.

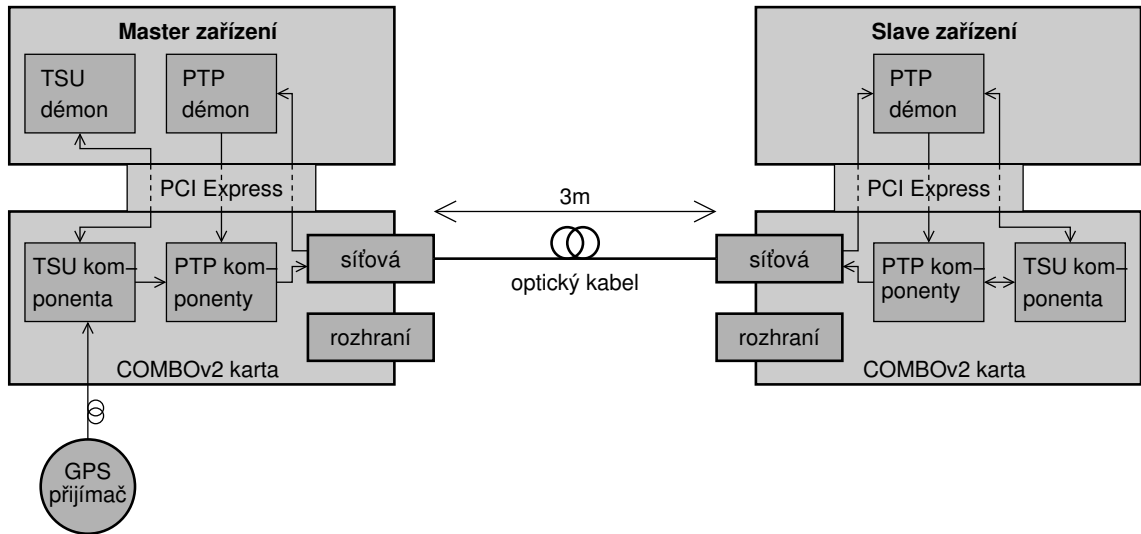
Po celkové integraci bylo potřeba ověřit funkčnost aplikace jako celku, tj. ověřit, že aplikace splňuje požadavky kladené zadáním, což je schopnost časové synchronizace. Za tímto účelem bylo vytvořeno testovací prostředí sestávající ze dvou serverů se vzájemně propojenými kartami COMBOv2. Schéma testovacího prostředí je zobrazeno na obrázku 7.1. Jeden z těchto serverů pracoval v režimu master, tedy jako zdroj informací pro časovou synchronizaci. Jeho hodiny byly synchronizovány nezávislým zdrojem synchronizace z GPS přijímače. Druhý server pracoval v režimu slave a synchronizoval se podle informací ze serveru pracujícího v režimu master. Karty COMBOv2 byly propojeny přímým optickým kabelem.

Pro účely synchronizace vnitřních hodin karty COMBOv2 na master zařízení je potřeba, aby na něm byl spuštěn TSU démon, který zpracovává informace z připojeného GPS přijímače. PTP démon má na master zařízení funkci poskytování informací pro synchronizaci prostřednictvím protokolu PTP na základě informací z jednotky TSU. Na slave zařízení se PTP démon stará jak o komunikaci prostřednictvím protokolu PTP, tak i o samotnou synchronizaci vnitřních hodin karty COMBOv2 na základě informací obdržených protokolem PTP. TSU démon není na slave zařízení potřeba, jelikož synchronizace na něm neprobíhá pomocí GPS přijímače.

### 7.1 Ověření funkčnosti

Funkčnost byla ověřována spouštěním aplikací na obou propojených serverech a sledováním logovacích souborů obou instancí aplikace. Průběh komunikace a přechod mezi stavy na master a slave zařízení je zobrazeno na schématu na obrázku 7.2.

Na serveru pracujícím v režimu master byla navíc spouštěna aplikace *tsuctl* pro synchronizaci hodin vůči GPS přijímači. Master zařízení začíná ve stavu LISTENING. Po uplynutí intervalu pro hledání jiných zdrojů synchronizace PTP protokolem (na obrázku 7.2 označeno jako *announceReceiptTimeout*), které v tomto případě neexistují, se master zařízení



Obrázek 7.1: Schéma propojení karet COMBOv2 v testovacím prostředí spolu s vyznačením komunikace mezi firmwarovými jednotkami a softwarovými démoni, kteří se účastní samotné synchronizace.

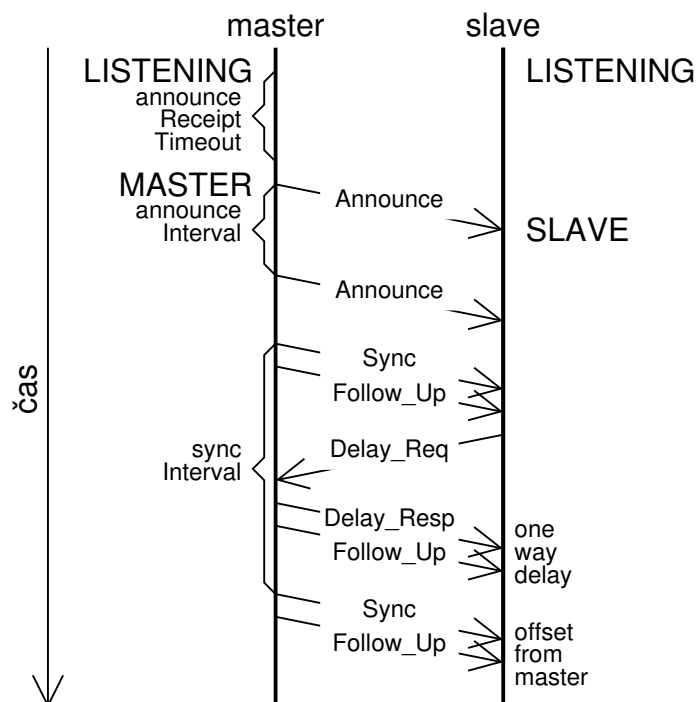
přepne do stavu MASTER, kdy začne pravidelně vysílat PTP zprávy Sync a Follow\_Up a odpovídat na PTP zprávy Delay\_Req zprávami Delay\_Resp a Delay\_Resp\_Follow\_Up.

Zařízení slave začíná stejně ve stavu LISTENING. Jelikož má nastaven příznak indikující práci pouze v režimu slave, nikdy nedojde k jeho přepnutí do režimu MASTER. Po obdržení zprávy Announce od master zařízení se přepne do stavu SLAVE, kdy vypočítává zpoždění na lince vysláním Delay\_Req zpráv a přijímáním Delay\_Resp a Delay\_Resp\_Follow\_Up zpráv, dále vypočítává offset od času na master zařízení na základě přijímaných zpráv Sync a Follow\_Up a z těchto údajů vypočítává hodnoty použité pro korekci svých hodin. Na obrázku 7.2 je první okamžik výpočtu označen nápisem *one-way delay* (jsou totiž známy všechny potřebné časové značky, jak je popsáno v kapitole 4.2) a první okamžik výpočtu offsetu od času na master zařízení *offset from master* (je známo zpoždění linky a byla obdržena Sync zpráva).

V logovacích souborech můžeme sledovat výměnu PTP zpráv včetně detailních údajů o časových značkách, přechod mezi stavy aplikace a na slave zařízení také statistiky zahrnující vypočítané zpoždění na lince, offset od času master zařízení a hodnotu pro korekci hodin. Z analýzy logovacích souborů vyplývá, že master zařízení po čase přejde do stavu MASTER a vysílá příslušné zprávy, slave zařízení přejde do stavu SLAVE a zasílaným zprávám z master zařízení rozumí a provádí výpočet zpoždění na lince, offsetu vůči času master zařízení a na základě těchto hodnot provádí korekci svých hodin.

## 7.2 Testování

Činnost aplikace je ovlivnitelná několika parametry. U obou zařízení master a slave je možné nastavit parametry *ingressLatency* a *egressLatency* popsané v kapitole 4.2. Hodnota těchto parametrů ovlivňuje všechny výpočty, které v sobě zahrnují časové značky generované při příjmu či vysílání PTP zpráv, tzn. výpočet zpoždění na lince, výpočet offsetu hodin slave zařízení vůči hodinám master zařízení a v důsledku také hodnotu pro korekci hodin slave



Obrázek 7.2: Přechod mezi stavy při komunikaci master a slave zařízení

zařízení. U slave zařízení je dále možné nastavit parametry použitých filtrů a regulátorů a také hodnotu korigující vypočítanou hodnotu pro korekci hodin.

Tabulka 7.1 zachycuje vliv jednotlivých parametrů na výpočet koncových hodnot časových značek na master i slave zařízení a na výpočet zpoždění linky a offsetu hodin slave zařízení vůči hodinám master zařízení.

<i>Parametr</i>	<i>časové značky</i>	<i>one-way delay</i>	<i>offset from master</i>
<i>ingressLatency</i>	✓	✓	✓
<i>egressLatency</i>	✓	✓	✓
tuhost IIR filtru	×	✓	✓
parametry PI regulátoru	×	✓	✓
parametr <i>baseAdjustment</i>	×	×	×

Tabulka 7.1: Vliv parametrů na výpočet zpoždění linky a offsetu hodin slave zařízení vůči hodinám master zařízení

Při testování je tedy nejprve nutné stanovit metodiku úpravy parametrů. Jelikož jsou master a slave zařízení propojeny přímým kabelem a výkyvy ve zpoždění jsou minimální, můžeme si dovolit vynechat IIR filtr používaný pro výpočet zpoždění linky nastavením tuhosti IIR filtru na hodnotu 1. Dalším krokem bude nalezení koeficientu integrační složky PI regulátoru používaného pro výpočet offsetu. Jeho hodnota má vliv na míru přiblížení výstupní hodnoty regulátoru k referenční hodnotě, jak je popsáno v kapitole 3.4. Po zjištění těchto hodnot se můžeme přesunout k získání hodnot parametrů *ingressLatency* a *egressLatency* tak, aby vypočítané zpoždění linky odpovídalo skutečnému zpoždění a offset hodin slave zařízení byl co nejmenší se střední hodnotou v co nejbližší hodnotě 0 ns. V případě,

že by se nepodařilo dosáhnout uspokojivých výsledků, byl by prozkoumán vliv parametru *baseAdjustment*.

Test č. 1 byl proveden s výchozími hodnotami všech parametrů. Parametry *ingressLatency* a *egressLatency* byly ponechány na hodnotě 0 ns, tuhost IIR filtru na hodnotě 1, což ho vyřadí z činnosti. Koeficient proporcionalní části byl nastaven na hodnotu  $\frac{1}{10}$  a koeficient integrační části na hodnotu  $\frac{1}{1000}$ . Parametr *baseAdjustment* nebyl použit (nastaven na hodnotu 0).

V grafu na obrázku 7.3 je vidět průběh synchronizace při testu č. 1. Na vodorovné ose je vynesena čas v sekundách od počátku měření na master zařízení, který v tomto případě slouží jako referenční. Vychází z časových značek pořizovaných na master zařízení při odesílání Sync zpráv. Levá svislá osa slouží pro zpoždění linky (one-way delay, červené hodnoty) a offset hodin slave zařízení vůči hodinám master zařízení (offset from master, zelené hodnoty). Jednotky levé svislé osy jsou nanosekundy. Pravá svislá osa slouží pro zobrazení velikosti driftu hodin slave zařízení. Jedná se o bezrozměrnou veličinu, která se používá pro výpočet hodnoty korekce hodin.

## Parametry PI regulátoru

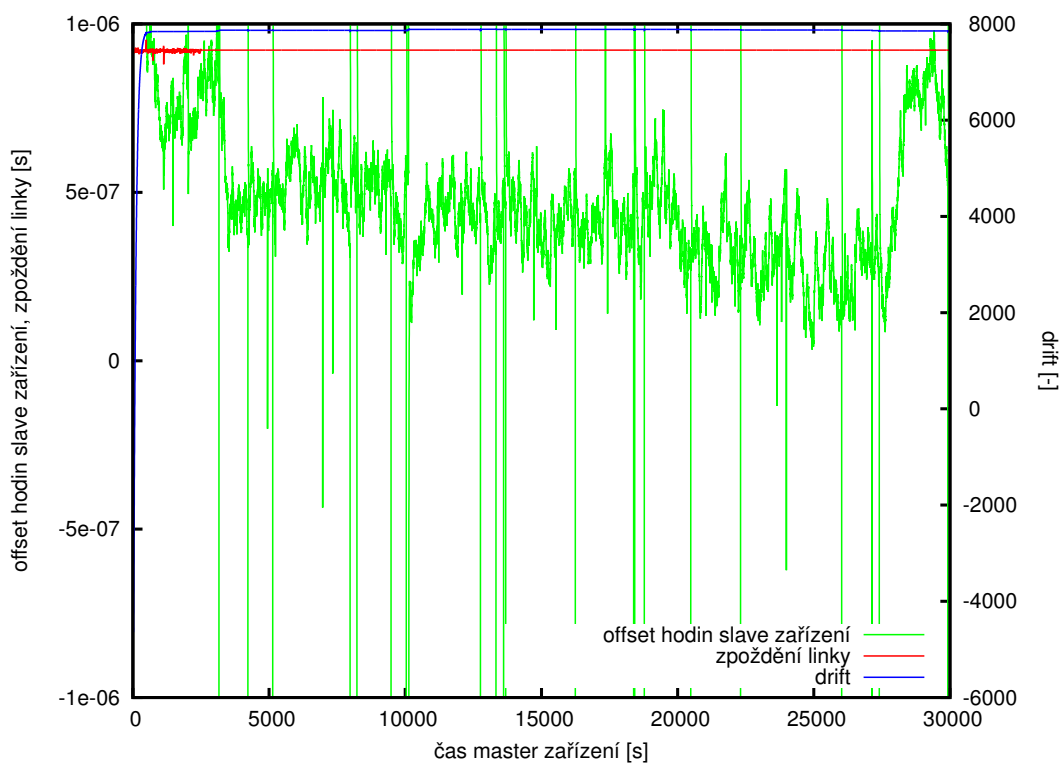
Střední hodnota naměřených hodnot offsetu po ustálení systému v testu č. 1 je 461,18 ns. Z grafu na obrázku 7.3 je vidět nestabilita systému při zvolených hodnotách parametrů a navíc není dosaženo nulové střední hodnoty. Příčinu můžeme hledat v nastavení parametrů PI regulátoru. V kapitole 3.4 je popsáno chování PID regulátoru v závislosti na nastavení koeficientů jeho složek. V tomto případě se jedná o stálou chybu, kdy se výstupní hodnota neustálí na referenční hodnotě. To je ovlivněno koeficientem integrační složky. Jeho hodnota závisí na konkrétním případě a požadovaném chování. Při prvním měření byla nastavena na  $\frac{1}{1000}$  což je tedy příliš malá hodnota na to, aby se výstupní hodnota ustálila na referenční hodnotě. Je třeba si uvědomit, že pracujeme s nanosekundami a výpočty probíhají s poměrně vysokými hodnotami. V testu č. 2 hodnotu koeficientu zvýšíme na  $\frac{1}{10}$  a budeme pozorovat, jaký vliv bude mít tato změna na průběh synchronizace. Ten je zachycen v grafu na obrázku 7.4. Můžeme pozorovat, že se offset skutečně přiblíží a osciluje kolem nulové hodnoty. V testu č. 2 bylo dosaženo střední hodnoty offsetu  $-0,022$  ns. Tím jsme tedy dosáhli požadovaného efektu průběhu offsetu a můžeme se přesunout k analýze druhé veličiny, zpoždění linky.

## Parametry *ingressLatency* a *egressLatency*

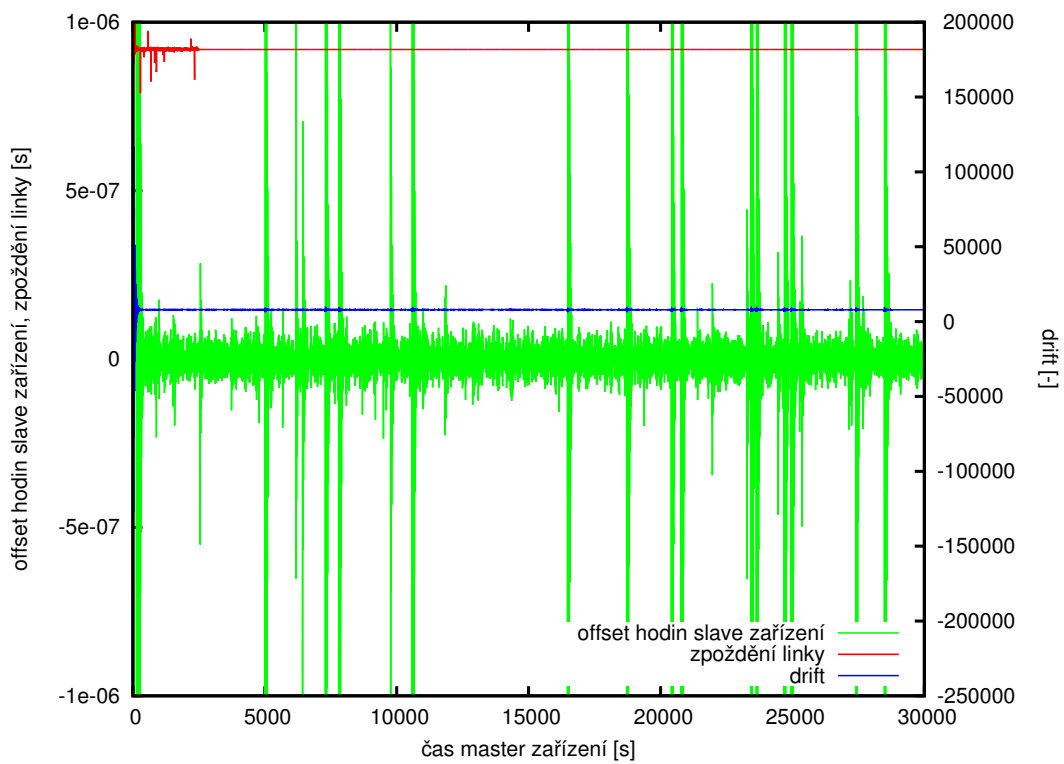
Střední hodnota vypočteného zpoždění linky v testu č. 2 byla 919,02 ns. Střední hodnota by měla vyjadřovat zpoždění PTP zpráv od výstupu z fyzického rozhraní zdrojového zařízení do příchodu na fyzické rozhraní cílového zařízení. V našem případě se jedná o zpoždění na optickém spoji, jehož délku známe a jsme ho tedy schopni vyčíslit. Pro výpočet v rovnici 7.1 bude použita rychlost šíření signálu v optickém vlákne  $2 \cdot 10^8$  m s<sup>-1</sup> [13] a délka použitého optického spoje 3 m.

$$t = \frac{s}{v} = \frac{3}{2 \cdot 10^8} = 15 \text{ ns} \quad (7.1)$$

Hodnota 15 ns je tedy požadovaná hodnota zpoždění na lince, již však neodpovídá naměřená hodnota v testu č. 2 s nulovými hodnotami parametrů *ingressLatency* a *egressLatency*. Je potřeba nalézt vhodné hodnoty zmíněných parametrů, při čemž využijeme sché-

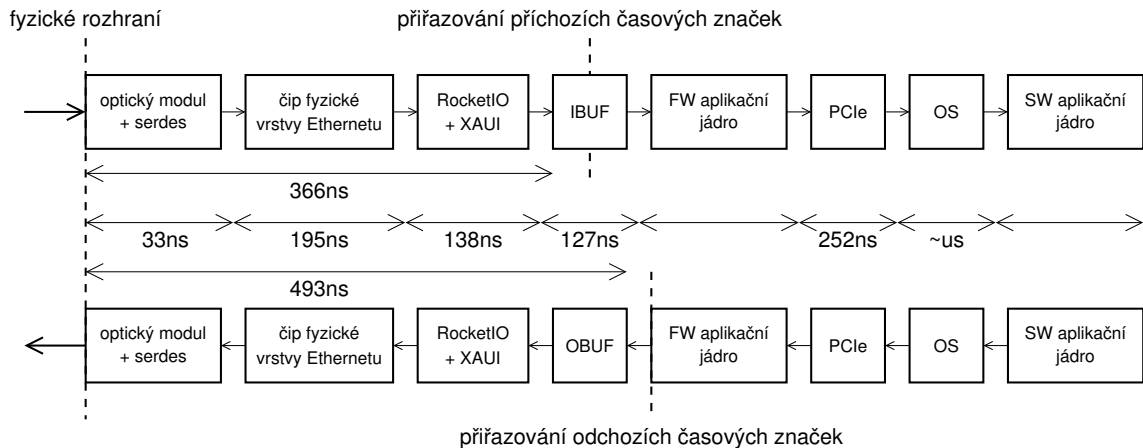


Obrázek 7.3: Test č. 1:  $k_i = \frac{1}{1000}$ ,  $ingressLatency=0$  ns,  $egressLatency=0$  ns



Obrázek 7.4: Test č. 2:  $k_i = \frac{1}{10}$ ,  $ingressLatency=0$  ns,  $egressLatency=0$  ns

matu zobrazujícího zpoždění jednotlivých částí karty COMBOv2 zpracovávajících příchozí a odchozí PTP zprávy. Schéma je zobrazeno na obrázku 7.5.



Obrázek 7.5: Zpoždění jednotlivých částí karty COMBOv2 při zpracování příchozích a odchozích PTP zpráv, převzato z [2]

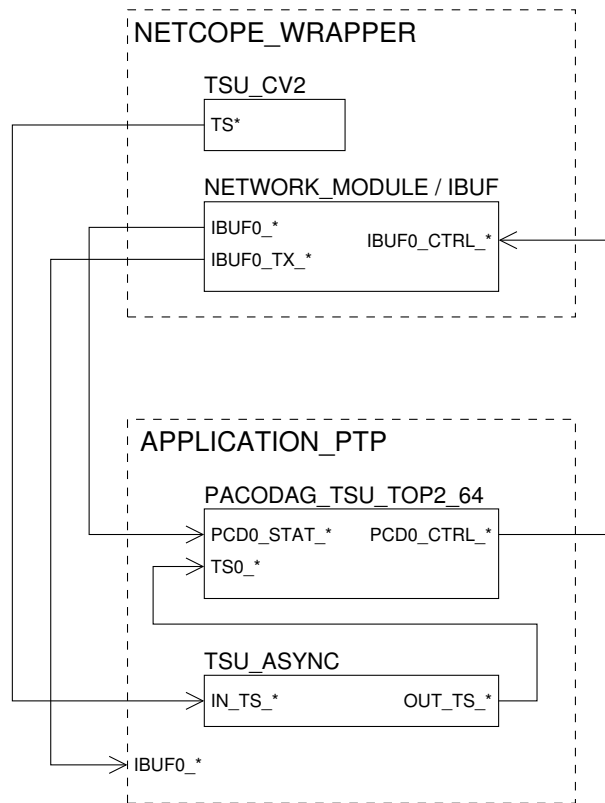
Při provádění úprav hodnot parametrů *ingressLatency* a *egressLatency* je třeba dbát na to, aby byly nastaveny na stejnou hodnotu na slave i master zařízení, jelikož obě zařízení mají stejnou strukturu firmwaru.

Pokud vycházíme ze schématu zpoždění částí karty COMBOv2, zpoždění od fyzického rozhraní do aplikačního jádra firmwaru je 493 ns. Test č. 3 spočívá v použití této hodnoty pro oba parametry *ingressLatency* a *egressLatency*. Průběh synchronizace je zachycen v grafu na obrázku 7.7. Střední hodnota zpoždění linky je v tomto testu  $-63,47$  ns. Záporné zpoždění je zjevně chybná hodnota.

Analyzujeme podrobněji místa přiřazování časových značek ve firmwaru karty COMBOv2. Odchozí časové značky jsou přiřazovány komponentou PTP\_SEND, konkrétně na výstupu podkomponenty FL\_FROM\_MEM, jak je popsáno v kapitole 6.1. V našem případě nejsou použity fronty na vstupech podkomponenty FL\_MERGE a samotná komponenta FL\_MERGE nevkládá do cesty žádné zpoždění. Z toho vyplývá, že odchozí časová značka je přiřazována na samém okraji firmwarové aplikační části, zpoždění k fyzickému rozhraní je tedy uvedených 493 ns. Naproti tomu příchozí časové značky jsou přiřazovány uvnitř komponenty IBUF, jak je vidět ze schématu propojení komponent spolupracujících na přiřazování časových značek na obrázku 7.6. Správná hodnota parametru *egressLatency* tedy leží v intervalu  $\langle 366 \text{ ns}; 493 \text{ ns} \rangle$ . Test č. 4 spočíval v použití nejnižší hodnoty intervalu pro hodnotu parametru *ingressLatency*. Průběh synchronizace je zobrazen v grafu na obrázku 7.8.

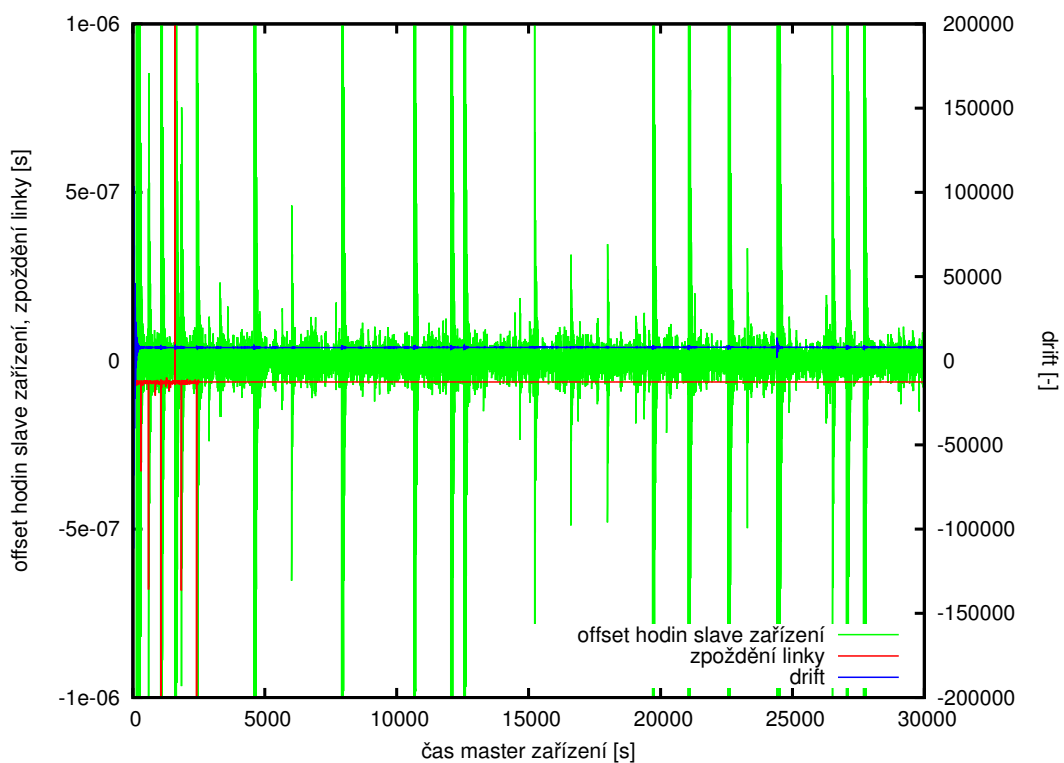
V testu č. 4 je střední hodnota zpoždění linky 53,45 ns. Z předchozích změn hodnoty parametru *egressLatency* můžeme vyvodit závislost zpoždění linky na jeho hodnotě. Čím větší je hodnota parametru *egressLatency* (při pevně zvolené hodnotě parametru *ingressLatency*), tím menší je vypočítané zpoždění linky (při hodnotě parametru *egressLatency* 493 ns v testu č. 3 byla střední hodnota  $-63,47$  ns).

Test č. 5 spočíval v provedení několika dalších měření pro nalezení správné hodnoty parametru *egressLatency*. Nejlepší výsledky byly dosaženy s hodnotou 413 ns. Průběh synchronizace je zobrazen v grafu na obrázku 7.9. S uvedenými hodnotami parametrů *ingressLatency* a *egressLatency* byla střední hodnota zpoždění linky 13,02 ns a střední hodnota offsetu  $-0,024$  ns.

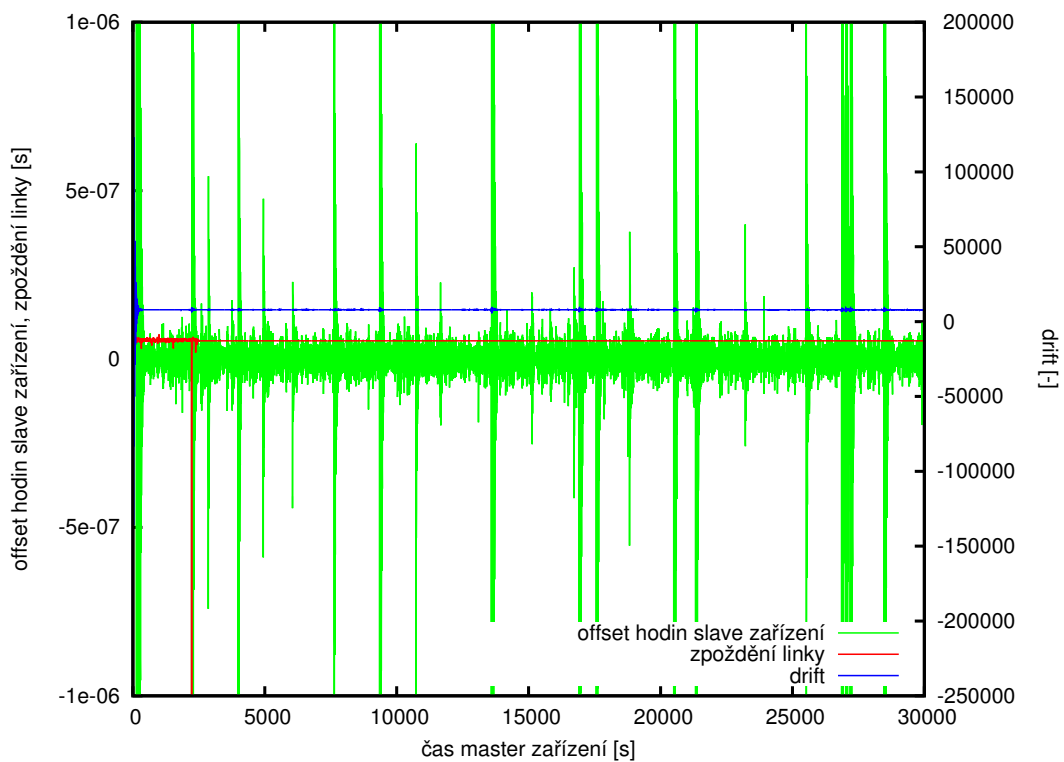


Obrázek 7.6: Schéma propojení komponent spolupracujících na přiřazování příchozích časových značek

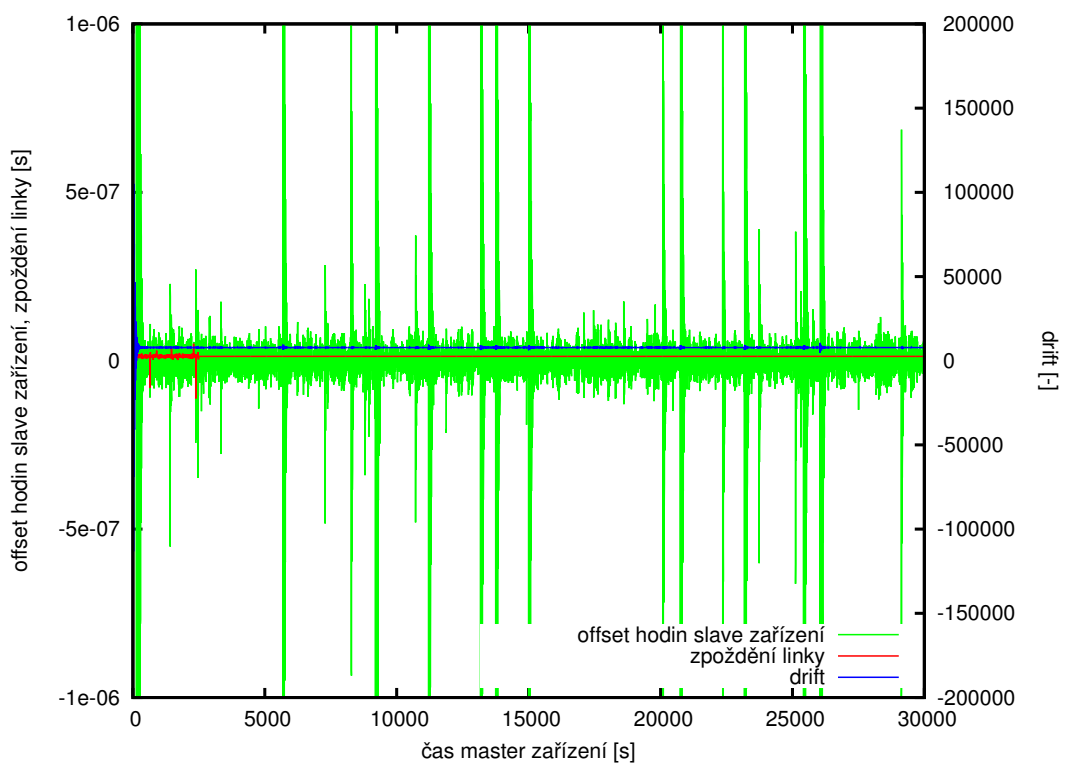
Na obrázku 7.10 je zobrazen detail počáteční fáze synchronizace v testu č. 5. Na začátku můžeme pozorovat oscilaci počítané hodnoty zpoždění linky a offsetu. Tato oscilace se postupně stabilizuje a v čase kolem 400s se již obě hodnoty ustálily a oscilují s přibližně stálou odchylkou. Systém se tedy na nezátížené lince ustálí do sedmi minut.



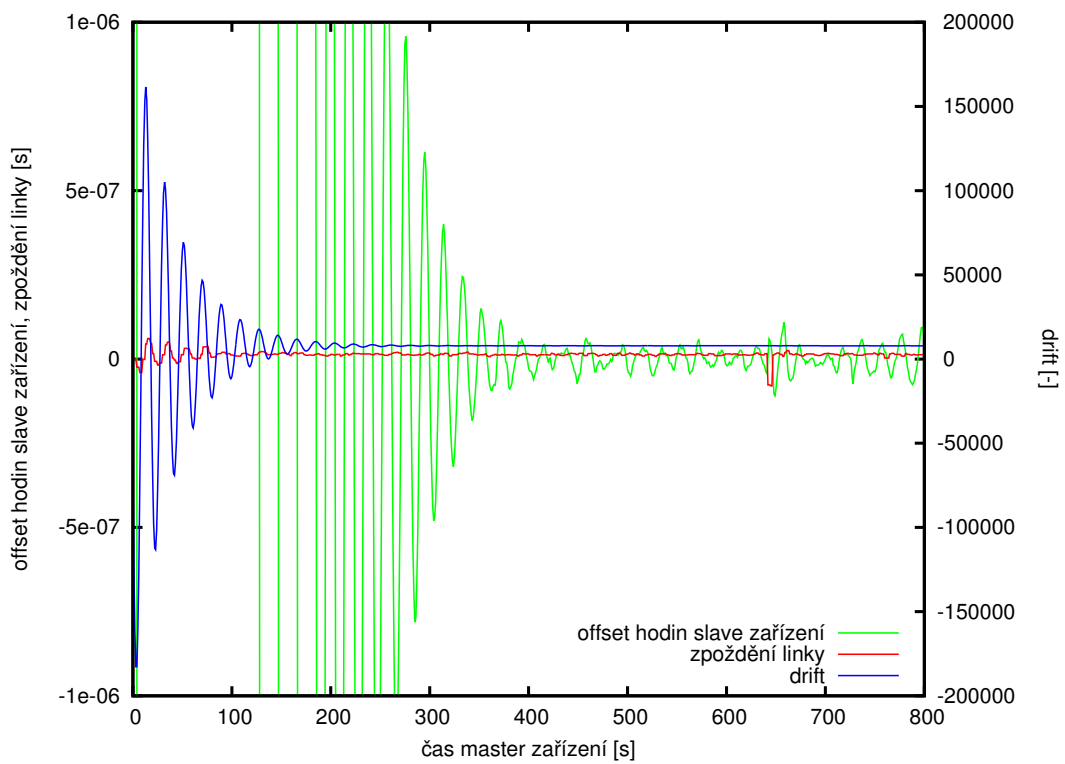
Obrázek 7.7: Test č. 3:  $ingressLatency=493$  ns,  $egressLatency=493$  ns



Obrázek 7.8: Test č. 4:  $ingressLatency=366$  ns,  $egressLatency=493$  ns

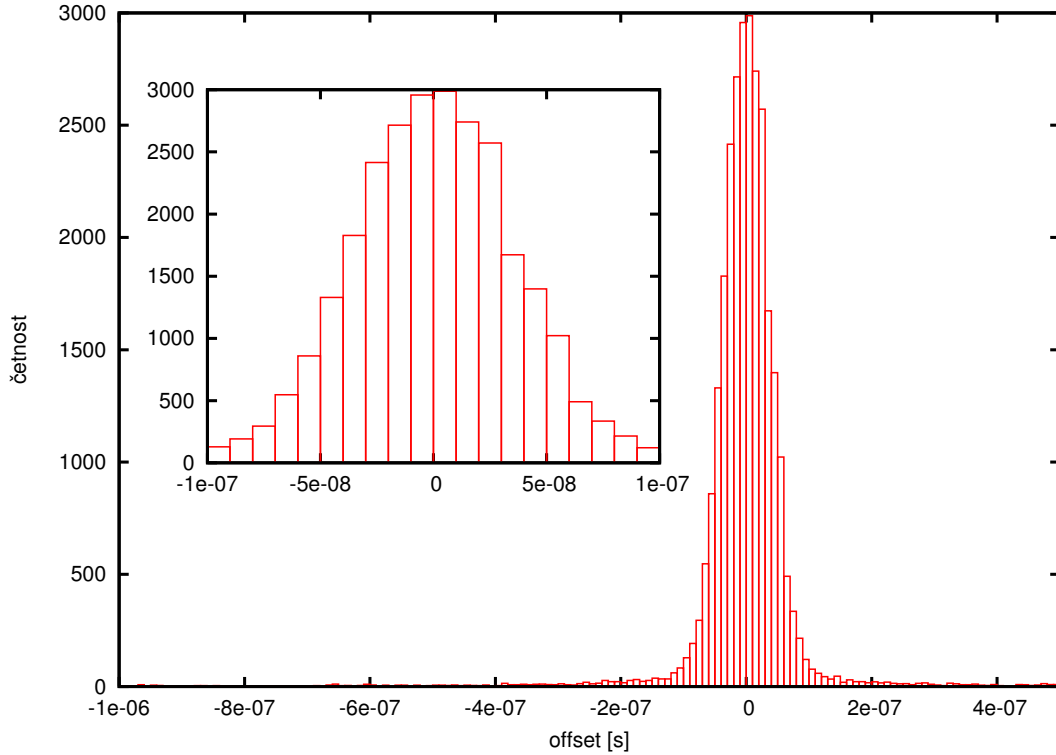


Obrázek 7.9: Test č. 5:  $ingressLatency=413$  ns,  $egressLatency=493$  ns



Obrázek 7.10: Detail počáteční fáze synchronizace v testu č. 5

Na obrázku 7.11 je zobrazen histogram hodnot offsetu hodin slave zařízení z testu č. 5. Jsou použity vzorky od času 1000 s, kdy se již systém nachází v ustáleném stavu, do času 30 000 s. Šířka koše je 10 ns. V levé horní části je zobrazen detail histogramu pro rozsah  $\langle -1 \cdot 10^{-7} \text{ s}; 1 \cdot 10^{-7} \text{ s} \rangle$ . Histogram je přibližně symetrický kolem hodnoty 0 ns, čemuž odpovídá i umístění střední hodnoty  $-0,024 \text{ ns}$  v koši přiléhajícímu hodnotě 0 ns.



Obrázek 7.11: Histogram offsetů pro test č. 5

Z histogramu lze získat následující informace o rozložení hodnot offsetu hodin slave zařízení:

- 78,33 % hodnot leží v intervalu  $\langle -50 \text{ ns}; 50 \text{ ns} \rangle$
- 92,54 % hodnot leží v intervalu  $\langle -100 \text{ ns}; 100 \text{ ns} \rangle$
- 97,11 % hodnot leží v intervalu  $\langle -500 \text{ ns}; 500 \text{ ns} \rangle$

## Souhrn testování

Výsledky provedených testů jsou shrnuty v tabulce 7.2. Pro výpočet střední hodnoty a směrodatné odchylky jsou použity vzorky od času 1000 s, kdy se již systém nachází v ustáleném stavu, do času 30 000 s. Ve všech testech je tuhost IIR filtru nastavena na hodnotu 1 a koeficient proporcionální složky PI regulátoru na  $\frac{1}{10}$ . Koeficient integrační složky PI regulátoru je v testu č. 1 nastaven na  $\frac{1}{1000}$ , v ostatních testech na  $\frac{1}{10}$ .

Velké hodnoty směrodatné odchylky offsetu hodin slave zařízení jsou způsobeny občasnými výraznými výkyvy offsetu. Jak ale vyplynulo z histogramu, 92,54 % hodnot leží v intervalu  $\langle -100 \text{ ns}; 100 \text{ ns} \rangle$ .

<i>Test</i>	<i>ingress-Latency</i>	<i>egress-Latency</i>	zpoždění linky		offset	
	[ns]	[ns]	$\bar{x}$ [ns]	$\sigma_x$ [ns]	$\bar{y}$ [ns]	$\sigma_y$ [ns]
1	0	0	921,89	0,93	461,18	410,51
2	0	0	919,02	1,40	-0,022	448,62
3	493	493	-63,47	68,29	0,102	659,40
4	366	493	53,45	50,07	0,001	435,34
5	413	493	13,02	1,99	-0,024	443,39

Tabulka 7.2: Výsledky provedených testů

Pro porovnání s jinou implementací jsou dále uvedeny výsledky popsané v článku [11]. Je zde použit démon PTPd na vestavěné platformě EX1048. Po dvou minutách je dosaženo přesnosti kolem 100  $\mu$ s a po ustálení po deseti minutách přesnosti 10  $\mu$ s. Z grafů a histogramů uvedených v článku je navíc vidět, že se střední hodnota offsetu neustálí blízko nulové hodnotě.

V článku [12] byly testovány implementace démonu PTPd upravené pro podporu hardwarových časových značek. Na nezatížené lince bylo dosaženo střední hodnoty offsetu 47 ns se směrodatnou odchylkou 38 ns.

# Kapitola 8

## Závěr

Cílem práce je shrnout současný stav problematiky synchronizace času v počítačových sítích. Nejprve se věnuje obecným tématům nutným pro pochopení principů, které mají vliv na návrh koncového řešení. Dále seznamuje čtenáře s mechanismy používanými v systémech pro synchronizaci času a také s konkrétními protokoly používanými v současných počítačových sítích.

Na základě nabytých znalostí byl vytvořen návrh systému pro synchronizaci času v počítačových sítích protokolem PTP. Při návrhu byl kladen důraz na dosažitelnou přesnost, čemuž byla podřízena i použitá platforma, kterou je speciální programovatelná síťová karta. Ta umožňuje implementovat kritické části systému v FPGA čipu na kartě a dosáhnout tak mnohem větší přesnosti než v případě čistě softwarových řešení.

Hlavním přínosem práce je návrh řešení přizpůsobený speciální platformě a jeho následná implementace, která umožňuje při správné konfiguraci dosáhnout vysoké přesnosti synchronizace. Byly navrženy a implementovány firmwarové bloky pro podporu protokolu PTP na zvolené platformě, upraveny ovladače operačního systému pro podporu přesných hardwarových časových značek a upravena volně dostupná implementace protokolu PTP tak, aby mohla spolupracovat se zvolenou platformou.

Implementované řešení je možné konfigurovat řadou parametrů a ovlivňovat tak jeho chování v průběhu synchronizace. Aby bylo řešení použitelné v praxi, byla provedena série testů pro určení hodnot konfigurovatelných parametrů. V průběhu testování bylo třeba analyzovat různé vlastnosti použitých komponent a algoritmů a na základě měření ověřovat předpokládané hodnoty parametrů.

Po ustálení, které trvá na nezatížené lince do deseti minut, dosahuje vhodně nakonfigurované řešení přesnosti v řádu desítek nanosekund.

Z diskuze nad vytvořeným řešením vyplynulo několik návrhů na úpravy, které by s sebou měly přinést možnost snadnější a transparentnější integrace do uživatelských aplikací. Jedním takovým návrhem je vyhnout se použití jednoho (z typicky osmi) DMA kanálů pro přenos příchozích PTP zpráv do softwarové části aplikace. Uživatelská aplikace již není schopna tento kanál využívat pro svoje účely, což může omezit propustnost přenosu uživatelských dat do softwaru. Řešením by bylo vytvořit speciální jednotku napojenou na vstupní rozhraní, která by detekovala příchozí PTP zprávy a přenášela je skrze rozhraní MI32 do softwarové části aplikace. Bylo by také vhodné provést další testy, především v živé síti, které by ověřily chování řešení v reálných podmínkách.

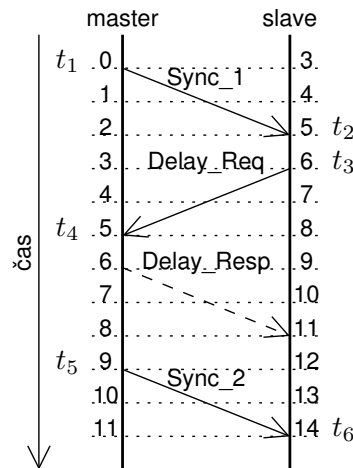
# Literatura

- [1] Linux kernel: timestamping. Linux kernel documentation.  
URL <https://www.kernel.org/doc/Documentation/networking/timestamping.txt>
- [2] Skupina projektu Liberouter: Projektová dokumentace frameworku NetCOPE. CESNET, veřejně nedostupné.  
URL <https://www.liberouter.org/technologies/netcope/>
- [3] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2002*, 2002, doi:10.1109/IEEESTD.2002.94144.
- [4] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, 2008, doi:10.1109/IEEESTD.2008.4579760.
- [5] LANTIME M600 GPS PTPv2: Network Time Server and PTPv2 Grandmaster Clock. 2011-10-27, Meinberg Radio Clocks GmbH & Co. KG.  
URL [http://www.meinbergglobal.com/download/docs/manuals/english/m600\\_mrs\\_ptpv2.pdf](http://www.meinbergglobal.com/download/docs/manuals/english/m600_mrs_ptpv2.pdf)
- [6] How to Test 10 Gigabit Ethernet Performance. *White Paper*, 2012, Spirent.  
URL [http://ekb.spirent.com/resources/sites/SPIRENT/content/live/FAQS/10000/FAQ10597/en\\_US/How\\_to\\_Test\\_10G\\_Ethernet\\_WhitePaper\\_RevB.PDF](http://ekb.spirent.com/resources/sites/SPIRENT/content/live/FAQS/10000/FAQ10597/en_US/How_to_Test_10G_Ethernet_WhitePaper_RevB.PDF)
- [7] PTPd Source Code Documentation. [cit. 2014-04-27].  
URL <http://ptpd.sourceforge.net/doc.html>
- [8] Allan, D. W.; Ashby, N.; Hodge, C. C.: The Science of Timekeeping. *Application Note 1289*, 1997, Hewlett-Packard Company.  
URL [http://www.allanstime.com/Publications/DWA/Science\\_Timekeeping/TheScienceOfTimekeeping.pdf](http://www.allanstime.com/Publications/DWA/Science_Timekeeping/TheScienceOfTimekeeping.pdf)
- [9] Aström, K. J.; Murray, R. M.: *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton Univeristy Press, 2009, ISBN 9781400828739, 294–314 s.  
URL [http://www.cds.caltech.edu/~murray/books/AM08/pdf/am08-complete\\_28Sep12.pdf](http://www.cds.caltech.edu/~murray/books/AM08/pdf/am08-complete_28Sep12.pdf)
- [10] Braden, R.: Requirements for Internet Hosts – Communication Layers. *Request for Comments: 1122*, October 1989, Internet Engineering Task Force.  
URL <https://tools.ietf.org/rfc/rfc1122.txt>

- [11] Correll, K.; Barendt, N.: Design Considerations for Software Only Implementations of the IEEE 1588 Precision Time Protocol. In *In Conference on IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, 2006.
- [12] Kovacshazy, T.; Ferencz, B.: Performance evaluation of PTPd, a IEEE 1588 implementation, on the x86 Linux platform for typical application scenarios. In *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*, 2012, ISSN 1091-5281, s. 2548–2552, doi:10.1109/I2MTC.2012.6229387.
- [13] Larson, D. R.; Jr., N. G. P.: A measurement of propagation delay. *Metrologia*, ročník 44, č. 1, 2007: str. 64.  
URL <http://stacks.iop.org/0026-1394/44/i=1/a=009>
- [14] Li, Z.; lin Zhong, Z.; chun Zhu, W.; aj.: A Hardware Time Stamping Method for PTP Messages Based on Linux system. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, ročník 11, č. 9, 2013: s. 5105–5111, doi:10.11591/telkomnika.v11i9.3255.
- [15] Meier, S.; Weibel, H.; Weber, K.: IEEE 1588 Syntonization and Synchronization Functions Completely Realized in Hardware. In *Precision Clock Synchronization for Measurement, Control and Communication*, ISPCS, 2008, s. 1–4, doi:10.1109/ISPCS.2008.4659209.
- [16] Mills, D.; Martin, J.; Burbank, J.; aj.: Network Time Protocol Version 4: Protocol and Algorithms Specification. *Request for Comments: 5905*, June 2010, Internet Engineering Task Force, ISSN 2070-1721.  
URL <https://tools.ietf.org/rfc/rfc5905.txt>
- [17] Mills, D. L.: Network Time Protocol Version 4: Reference and Implementation Guide. *NTP Working Group: Technical Report 06-6-1*, 2006, University of Delaware.  
URL <http://www.eecis.udel.edu/~mills/database/reports/ntp4/ntp4.pdf>
- [18] Smotlacha, V.; Novotný, J.; Martínek, T.: Hardware Supported Precise Timestamps Generation. In *CESNET technical report 11/2008*, 2008.  
URL <http://archiv.cesnet.cz/doc/techzpravy/2008/hardware-supported-timestamp-generation/>
- [19] Stanton, K. B.: 802.1AS Tutorial. *November 2008 Plenary*, 2008-11-13, 802.1 AVB TG.  
URL <http://www.ieee802.org/1/files/public/docs2008/as-kbstanton-8021AS-overview-for-dot11aa-1108.pdf>
- [20] Windl, U.; Dalton, D.; Martinec, M.; aj.: The NTP FAQ and HOWTO. 2006-11-21 [cit. 2013-11-20].  
URL <http://www.ntp.org/ntpfaq/NTP-a-faq.htm>
- [21] Čejka, T.; Benáček, P.; Štěpán Friedl; aj.: COMET – COMBO Ethernet Tester. In *CESNET technical report 6/2012*, 2012.  
URL <http://www.cesnet.cz/wp-content/uploads/2013/03/comet.pdf>

## Příloha A

# Příklad výpočtu zpoždění linky a offsetu protokolem PTP



Obrázek A.1: Průběh komunikace při výpočtu offsetu protokolem PTP

Pro jednoduchost jsou z výpočtů vypuštěny hodnoty z korekčních políček jednotlivých PTP zpráv.

Nejprve je nutné vypočítat zpoždění na lince, tj. parametr  $meanPathDelay$ . Pro jeho výpočet je použita první zpráva Sync spolu s dvojicí zpráv Delay\_Req a Delay\_Resp. Výpočet je zachycen rovnicí A.1.

$$meanPathDelay = \frac{1}{2}[(t_2 - t_3) - (t_1 - t_4)] = \frac{1}{2}[(5 - 6) - (0 - 5)] = 2 \quad (\text{A.1})$$

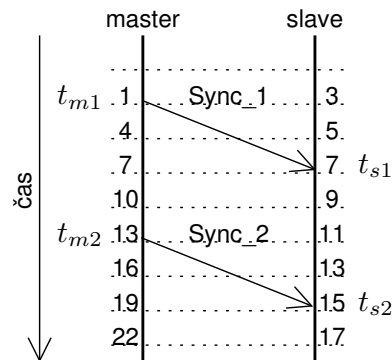
Pomocí druhé zprávy Sync již můžeme spočítat offset, tj. parametr  $offsetFromMaster$ , slave zařízení vůči master zařízení. Výpočet je zachycen rovnicí A.2.

$$offsetFromMaster = t_6 - t_5 - meanPathDelay = 14 - 9 - 2 = 3 \quad (\text{A.2})$$

Výsledná hodnota parametru  $offsetFromMaster$  již může být použita pro synchronizaci hodin slave zařízení. Pokud by však hodiny byly krokově změněny o vypočítanou hodnotu 3 a současně s tím byla dvakrát vyčtena jejich hodnota, mohlo by se stát, že by výsledný rozdíl vyčtených časů byl záporný. Proto je pro korekci hodin nutné použít důmyslnější techniky.

## Příloha B

# Příklad výpočtu hodnoty korekce po resetování hodin



Obrázek B.1: Průběh komunikace při výpočtu hodnoty korekce po resetování hodin

Po resetování hodin jsou použity hodnoty ze dvou posledních Sync zpráv (resp. odpovídajících Follow\_Up zpráv). Proměnné  $t_m$  reprezentují odchozí časové značky Sync zpráv na master zařízení, proměnné  $t_s$  příchozí časové značky na slave zařízení. Proměnná  $adj$  je hodnota určená pro korekci hodin a je vypočítána podle rovnice B.1.

$$adj = \frac{\Delta t_m - \Delta t_s}{\Delta t_s} \quad (\text{B.1})$$

kde  $\Delta t_m = t_{m2} - t_{m1}$  a  $\Delta t_s = t_{s2} - t_{s1}$ , tedy

$$adj = \frac{(13 - 1) - (15 - 7)}{15 - 7} = \frac{12 - 8}{8} = \frac{1}{2} \quad (\text{B.2})$$

Což vyjadřuje, že chod hodin je třeba urychlit o  $\frac{1}{2}$ -násobek současné frekvenci hodin. To je vidět i z odpovídajících si časů na schématu na obrázku B.1, kdy na master zařízení uběhnou tři časové jednotky, zatímco na slave zařízení pouze dvě časové jednotky.

## Příloha C

# Makefile: cíle specifické pro kartu COMBOv2

<i>Cíl</i>	<i>Popis</i>
combo	Přeloží aplikaci ptpctl pro kartu COMBOv2 Ve všech zdrojových souborech definuje makro CONFIG_CV2, která zajistí použití funkcí modulů pro kartu COMBOv2 místo standardních systémových funkcí
clean-combo	Smaže všechny dočasné soubory vzniklé při překladu aplikace ptpctl pro kartu COMBOv2
run-combo	Spouští aplikaci ptpctl s parametry definovanými pomocí proměnných Makefile, používá se pro spouštění aplikace v režimu master
run-combo-slave	Spouští aplikaci ptpctl s parametry definovanými pomocí proměnných Makefile, je zakázán přechod do master režimu, používá se pro spouštění aplikace v režimu slave
run-combo-slave-csv	Spouští aplikaci ptpctl pomocí cíle run-combo-slave a navíc vytváří speciální logovací soubor obsahující informace o aktuálních hodnotách parametrů synchronizace
clean-combo-slave-csv	Smaže logovací soubory aplikace ptpctl
<i>Pomocné cíle</i>	<i>Popis</i>
iptables	Konfiguruje síťový filtr iptables pro příjem PTP provozu, povoluje příchozí UDP provoz na portech 319 (PTP zprávy událostí) a 320 (obecné PTP zprávy)
parse-log	Z logovacího souboru aplikace ptpctl extrahuje informace o vypočítaném zpoždění na lince, offsetu hodin vůči hodinám master zařízení a driftu hodin

Tabulka C.1: Makefile: cíle specifické pro kartu COMBOv2

## Příloha D

# Obsah přiloženého CD

<i>Složka nebo soubor / Popis</i>
/ Kořenový adresář CD
/xmatou08.pdf Technická zpráva ve formátu PDF
/tex/ Zdrojové kódy technické zprávy ve formátu L <sup>A</sup> T <sub>E</sub> X
/tex/fig/ Schémata použitá v technické zprávě a jejich zdrojové soubory
/tex/graph/ Grafy použité v technické zprávě a jejich datové soubory
/tex/graph/ptpctl.(eps run hist stats).plt Definice pro program gnuplot pro generování grafů v EPS formátu (eps) a na X11 výstup (run), pro generování histogramu v EPS formátu (hist) a statistických dat (stats)
/literature/ Dostupná citovaná literatura v textovém formátu nebo ve formátu PDF
/fw/ Firmwarová část aplikace
/fw/scripts/ Skripty pro práci s kartou COMBOv2 (inicializace, nahrávání firmwaru, zavádění ovladačů OS, inicializace síťových rozhraní OS)
/fw/tsu/ Zdrojové kódy jednotky TSU
/fw/tsu/tsu_cv2/comp/tsu_cv2_core/ Zdrojové soubory upravené části jednotky TSU
/fw/tsu/tsu_cv2/comp/tsu_cv2_core/sim/ Simulační soubory upravené části jednotky TSU
/fw/fw-combov2-10g2-stdtsu/ Zdrojové soubory firmwaru s podporou protokolu PTP v master režimu (s neupravenou jednotkou TSU)

/fw/fw-combov2-10g2-ptptsu/ Zdrojové soubory firmwaru s podporou protokolu PTP ve slave režimu (s upravenou jednotkou TSU)
/fw/fw-combov2-10g2-(std ptp)tsu/src/applications/application_ptp.vhd Hlavní zdrojový soubor firmwarové části aplikace
/fw/fw-combov2-10g2-(std ptp)tsu/src/applications/design_ptp.xml Konfigurační soubor adresového prostoru firmwarové části aplikace
/fw/fw-combov2-10g2-(std ptp)tsu/src/applications/ptp_send/ Zdrojové soubory komponenty PTP_SEND
/fw/fw-combov2-10g2-(std ptp)tsu/src/applications/ptp_send/sim/ Simulační soubory komponenty PTP_SEND
/fw/fw-combov2-10g2-(std ptp)tsu/src/applications/ptp_send/comp/fl_from_mem/ Zdrojové soubory komponenty FL_FROM_MEM
/fw/fw-combov2-10g2-(std ptp)tsu/src/applications/ ptp_send/comp/fl_from_mem/sim/ Simulační soubory komponenty FL_FROM_MEM
/fw/fw-combov2-10g2-(std ptp)tsu/src/applications/ptp_send/comp/fl_merge/ Zdrojové soubory komponenty FL_MERGE
/fw/fw-combov2-10g2-(std ptp)tsu/src/applications/ptp_send/comp/fl_merge/sim/ Simulační soubory komponenty FL_MERGE
/sw/ Softwarová část aplikace
/sw/drivers/dkms-combo-driver-0.7.8-1/ Ovladače karty COMBOv2 s přidanou podporou hardwarových časových značek
/sw/filter-pid/ Zdrojové soubory obecného IIR filtru a PID regulátoru
/sw/ptpctl/ Zdrojové soubory upravené aplikace ptpv2d
/sw/ptpctl/Makefile Soubor pro překlad aplikace ptpctl, specifické cíle pro kartu COMBOv2 popsány v příloze <b>C</b>
/sw/ptpctl/dep/combov2.[hc] Zdrojové kódy modulu pro pro inicializaci karty COMBOv2 popsáno v kapitole <b>6.2</b>
/sw/ptpctl/dep/combov2-ptp.[hc] Zdrojové kódy modulu pro práci s komponentou PTP_SEND popsáno v kapitole <b>6.2</b>
/sw/ptpctl/dep/combov2-tsu.[hc] Zdrojové kódy modulu pro práci s komponentou TSU popsáno v kapitole <b>6.2</b>
/sw/ptpctl/dep/net_tstamp.h Linuxové API pro práci s hardwarovými časovými značkami, v systémových adresářích některých distribucí tento hlavičkový soubor chybí, i když jsou hardwarové časové značky jádrem podporované
/sw/ptpctl/dep/pcap.h Pomocný hlavičkový soubor s definicí datových struktur pro formát PCAP

<p>/sw/ptpctl/protocol.c</p> <p>Zdrojový soubor implementující subsystém zpracování zpráv protokolu PTP, v případě karty COMBOv2 jsou volány speciální funkce pro konstrukci PTP zpráv a práci s hardwarovými časovými značkami z modulů pro práci s komponentou PTP_SEND a TSU popsaných v kapitolách 6.2 a 6.2</p>
<p>/sw/ptpctl/dep/servo.c</p> <p>Zdrojový soubor implementující algoritmy pro korekci hodin, v případě karty COMBOv2 jsou volány speciální funkce pro korekci hodin z modulu pro práci s komponentou TSU popsaného v kapitole 6.2</p>
<p>/sw/ptpctl/dep/startup.c</p> <p>Zdrojový soubor implementující inicializační funkce a zpracování parametrů příkazové řádky, přidané parametry specifické pro kartu COMBOv2 jsou popsány v kapitole 6.2</p>

Tabulka D.1: Obsah příloženého CD