České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra měření

Synchronizace času v distribuovaných heterogenních měřicích a řídicích systémech

Disertační práce

Jan Breuer

Školitel
doc. Ing. Jaroslav Roztočil, CSc.

Školitel specialista
doc. Ing. Jan Fischer, CSc.

Srpen, 2016

Doktorský studijní program
Elektronika a informatika

Studijní obor
Měřicí technika

Poděkování

Na tomto místě bych rád poděkoval všem, kteří mě v mé práci podporovali a věnovali mi čas při konzultacích. Především děkuji svému školiteli doc. Ing. J. Roztočilovi, CSc. za podporu a cenné rady, doc. Ing. J. Fischerovi, CSc. za odborné konzultace a Ing. V. Vignerovi za rozsáhlé rozpravy nad tématikou, které mně pomohly se lépe zorientovat. Dále bych chtěl poděkovat spolupracovníkům z Katedry měření za podporu formou rad, jmenovitě pak prof. Ing. V. Haaszovi, CSc. za podporu především při dokončování práce. Neméně chci poděkovat svým blízkým, zejména mé ženě Zdeňce za poskytnutí zázemí a času věnovat se vědecké činnosti.

Čestné prohlášení

Jméno a příjmení doktoranda: Jan Breuer

Prohlašuji, že jsem svou disertační práci vypracoval samostatně a v předložené práci důsledně citoval použitou literaturu.

Specifikace autorského podílu:

Podíl na všech popisovaných experimentálních zařízení je 50 %. Ostatní části předložené disertační práce a prezentovaných výsledků vznikly bez autorské spoluúčasti jiných osob.

Praha dne ....

podpis

Abstrakt

Cílem této práce je nalezení způsobu měření různých zdrojů chyb synchronizace hodin a způsobu jejich vzájemného odlišení. Práce se zabývá synchronizací hodin především v paketových sítích za použití protokolu podle standardu IEEE 1588. Významným zdrojem chyb v takových sítích je změna přenosového zpoždění paketů.

Byla navržena a realizována metoda pro přesné měření přenosového zpoždění, která vychází z protokolu pro přesnou časovou synchronizaci podle IEEE 1588. Pouhé vzájemné porovnání například 1 PPS výstupů nestačí k měření parametrů sítě, protože v sobě obsahuje např. i vlastnosti filtrování zpoždění paketů na straně podružných hodin.

Metoda spočívá v implementaci protokolu na daném komunikačním médiu tak, že pakety slouží pouze pro měření zpoždění, protože komunikující protistrany jsou synchronizovány jinak. Bylo vytvořeno zařízení pro lokální měření s rozlišovací schopností 8 ns, které vnitřně synchronizuje komunikační porty. Dále bylo vytvořeno zařízení pro vzdálené měření, které pro vlastní synchronizaci na obou koncích využívá GPS přijímač.

Zařízení bylo kalibrováno na sítích se známým zpožděním. První kalibrace spočívaly v měření zpoždění na samotném kabelu, další měření byla provedena na simulátoru zpoždění přenosové cesty. Zařízení bylo použito pro změření vlastností různých síťových prvků. Bylo naměřeno např. zpoždění běžného síťového přepínače, které vykazovalo změny zpoždění řádově stovky ns bez zátěže. Síťový přepínač s podporou protokolu IEEE 1588 zapisoval informaci o zpoždění do přenášených zpráv, a tím bylo možné změny zpoždění kompenzovat.

Dalším významným zdrojem chyb synchronizace může být vlastní implementace protokolu. Byl navržen a implementován nástroj pro sledování komunikace, který automaticky detekuje chyby ve vzájemných výměnách paketů.

Protokol IEEE 1588 byl dále implementován do různých zařízení a operačních systémů a byly porovnány vlastnosti jednotlivých implementací. Byly zvoleny operační systémy Windows a Linux a dále byl testován systém RTX, což je real-time rozšíření pro Windows.

Výsledky měření byly použity pro návrh distribuovaného systému pro sběr dat, který využívá časovou synchronizaci a dále používá časový multiplex pro vzájemnou komunikaci, aby se neprojevovalo zatížení sítě do zpoždění paketů pro synchronizaci.

Abstract

The aim of this work is to find a way of measuring various sources of clock synchronization errors and a way to differentiate among these sources. This work deals mainly with the clock synchronization over packet networks using a protocol defined in the IEEE 1588. A significant source of errors in such networks is a packet delay variation.

A method for precise measuring of the packet delay was designed and implemented. It is based on a precise time protocol synchronization according to the IEEE 1588. A mere comparison of e.g. 1 PPS outputs is not sufficient enough to distinguish between network parameters and e.g. characteristics of the filtering of packet delays by the slave clock.

The method consists of implementation of the protocol on the communication medium so that the packets are used only for the delay measurement whereas the communicating counterparts are synchronized in a different way. A device for a local measurement with a resolution of 8 ns was developed to internally synchronize the communication ports. Furthermore another device for a distributed measurement was created using GPS receivers for synchronization of both ends.

The device was calibrated on networks with a known delay. First calibration consisted of measuring the delay on the cable itself, another measurement was performed on the packet delay simulator. The device was used to measure the properties of various network elements. For example the measured delay of a conventional network switch showed a packet delay variance of hundreds of ns without a load. An Ethernet switch with a support of the IEEE 1588 protocol wrote a delay information to the packets so that the compensation of the delay variations was possible.

Another important source of synchronization errors can be the protocol implementation itself. A communication monitoring tool, which automatically detects errors in the mutual exchanges of the packets was designed and implemented.

The IEEE 1588 protocol was implemented into a variety of devices and operating systems. The precision of the time synchronization using these implementations was compared. Windows and Linux were selected as general purpose systems and RTX, a real-time extension for Windows, was selected as a real-time operating system.

The results were used for designing a distributed data acquisition system that uses the time synchronization plus the time-division multiplex for communication so that the network traffic does not affect the delay of packets for synchronization.

Obsah

1 Úvod....1
2 Současný stav problematiky....3
2.1 Synchronizace obecně....3
2.2 Synchronizace v bezdrátových senzorových sítích....6
2.3 Synchronizace v počítačových sítích....7
2.4 Precise Time Protocol....10
3 Stanovení cílů práce....18
4 Definice základních pojmů....19
4.1 Definice hodin zařízení....19
4.2 Synchronizace hodin....21
4.3 Požadavky na synchronizaci....21
4.4 Časová zpoždění v synchronizaci....22
4.5 Parametry hodnocení synchronizace....23
4.6 Metriky pro hodnocení synchronizace....24
5 Měření zpoždění paketů....29
5.1 Popis problému....29
5.2 Návrh měřicího zařízení....30
5.3 Naměřené výsledky....32
5.4 Zhodnocení výsledků....44
6 Testování a validace PTP protokolu....45
6.1 Struktura aplikace....46
6.2 Pravidla pro zpracování dat....47
6.3 Praktické testování a výsledky....51
6.4 Zhodnocení výsledků....52
7 Eliminace vlivu datového zatížení přenosového média....54
7.1 Časový multiplex....54
7.2 Návrh časového multiplexu pro měřicí aplikace....55
8 Implementace a funkční vzorky....58
8.1 Podpora protokolu IEEE 1588 u mikrokontrolérů STM32....59
8.2 Zařízení pro synchronizaci s podporou protokolu IEEE 1588....68
8.3 Měřicí ústředna s protokolem IEEE 1588....75
8.4 Implementace protokolu IEEE 1588 v OS Linux....80
8.5 Implementace protokolu IEEE 1588 v OS Microsoft Windows....82
8.6 Implementace protokolu IEEE 1588 v OS IntervalZero RTX....85
8.7 Zhodnocení výsledků....90
9 Závěr....91
9.1 Porovnání výsledků práce se stanovenými cíli....91

9.2 Originální výsledky dosažené v této práci....92
9.3 Doporučení pro další rozvoj a realizace v praxi....93
10 Literatura....94
11 Seznam vlastních publikací....99
12 Seznam symbolů a zkratek....102
13 Přílohy....103
13.1 SCPI Parser....103
13.2 Sleepy Cat IDE....106
13.3 Sleepy Cat KIT....107
13.4 Odhad efektivního počtu bitů heterogenního systému pro sběr dat 110

1 Úvod

Synchronizace hodin se v měřicích, řídicích a telekomunikačních systémech využívá stále častěji. Používá se k synchronizaci času nebo frekvence, a to buď pouze jako prostředek pro synchronizaci běhu sítě, a nebo přímo k synchronizaci hodin sloužících např. k zachytávání časových značek událostí nebo ke generování událostí v přesném čase.

V distribuovaných měřicích aplikacích lze synchronizaci uplatnit k porovnání nebo k fúzi dat z více nezávislých senzorů. Tento způsob využití je znám hlavně z bezdrátových senzorových sítí (Wireless sensor network – WSN). V laboratorních měřicích systémech lze nahradit komunikační sběrnice, jako je například GPIB, za ethernetové připojení a využít tak například stávající síťovou infrastrukturu nebo vysokou přenosovou rychlost. Při požadavku na přesné časování se pouze zavede synchronizace času a nadále lze ethernetové připojení používat bez dalších doprovodných synchronizačních sběrnic.

V průmyslových řídicích systémech se pro komunikaci používají sběrnice, které zaručují dobu doručení zpráv. Běžné paketové sítě jako např. Ethernet jsou nedeterministické, protože není tato doba zaručena. Pomocí synchronizace lze plánovat komunikaci na síti a tím přidat determinismus k jinak nedeterministické síti a zaručit dobu doručení zpráv. Tímto způsobem lze například nahradit specializované průmyslové sběrnice za univerzální, např. pomocí Ethernetu. Náhrada za Ethernet má výhodu ve zvýšení přenosové rychlosti a v možnosti použití běžně dostupných síťových prvků.

V telekomunikačních systémech se synchronizace používá také a to jednak ve smyslu synchronizace času přepínání síťových prvků (Time-division multiplexing TDM) a jednak ve smyslu synchronizace frekvence například vysílačů v mobilních sítích. Běžně používané synchronní přenosy řešily synchronizaci již v návrhu. Při nahrazení těchto přenosů za univerzální paketové sítě je nutné řešit synchronizaci dodatečným mechanizmem.

Pro zajištění synchronizace hodin v distribuovaných systémech je zapotřebí vybrat prostředek pro vlastní synchronizaci. Pro prostorově rozsáhlé sítě je například možné využít Global Positioning System (GPS). V některých případech to ale není možné, protože jsou GPS přijímače příliš drahé pro použití na velkém množství modulů a nebo není dostupný GPS signál, např. pod zemí, v budovách, pod vodou.

Další možností synchronizace je využití speciálních synchronizačních sběrnic, na kterých bude přesný synchronizační signál např. Pulse Per Second (PPS) nebo 10MHz sinusový signál a nebo budou navíc obsahovat i informaci o čase pomocí časového kódu. Pokud se jedná o rozsáhlejší síť,

je nutné u koncových modulů kompenzovat zpoždění vzniklá na přenosové cestě.

V některých případech jsou stávající specializované komunikační sběrnice omezující např. z důvodu přenosové kapacity. Tyto sběrnice ale často řeší synchronizaci již ve vlastním návrhu. Při nahrazení těchto typů specializovaných sběrnic pomocí univerzálních je tedy nutné dodat explicitní synchronizaci. Tento problém nastává například při přechodu na paketové sítě, jako je třeba Ethernet.

V Internetu je pro takové účely nejrozšířenější protokol Network Time Protocol (NTP). S NTP protokolem je možné v lokální síti dosáhnout synchronizační chyby menší než 100 µs. Pro průmyslové aplikace je vhodnější protokol IEEE 1588 známý jako Precise Time Protocol (PTP). Pomocí tohoto protokolu je možné dosáhnout synchronizační chyby menší než 1 µs a zároveň automaticky kompenzovat délku přenosové cesty.

Tyto protokoly mohou pracovat na existující síti bez jakýchkoli hardwarových změn, jsou ale ovlivněny provozem v síti a různými proměnnými zpožděními síťových prvků. Pro získání ještě přesnější a robustnější synchronizace je zapotřebí použít speciální síťové prvky, tedy především síťové přepínače (switch) a síťové karty jednotlivých koncových zařízení. Díky specializovaným síťovým prvkům lze pak použít synchronizaci i na nižší úrovni, například pomocí synchronního Ethernetu SyncE.

Většina prací o přesné časové synchronizaci po sítích pro obecné použití (např. Ethernet) předpokládá modifikaci fyzické nebo linkové vrstvy. Taková modifikace vede ke zpřesnění synchronizace, ale za cenu zvýšení nákladů na takovou síť a k potřebě specializovaného hardware. Takové požadavky jsou pak protichůdné k tomu, co původně vedlo k použití těchto obecných sítí. Menší množství příspěvků se pak věnuje statistickému zpracování komunikace při synchronizaci a odhadu parametrů a návrhu algoritmů pro takové odhady.

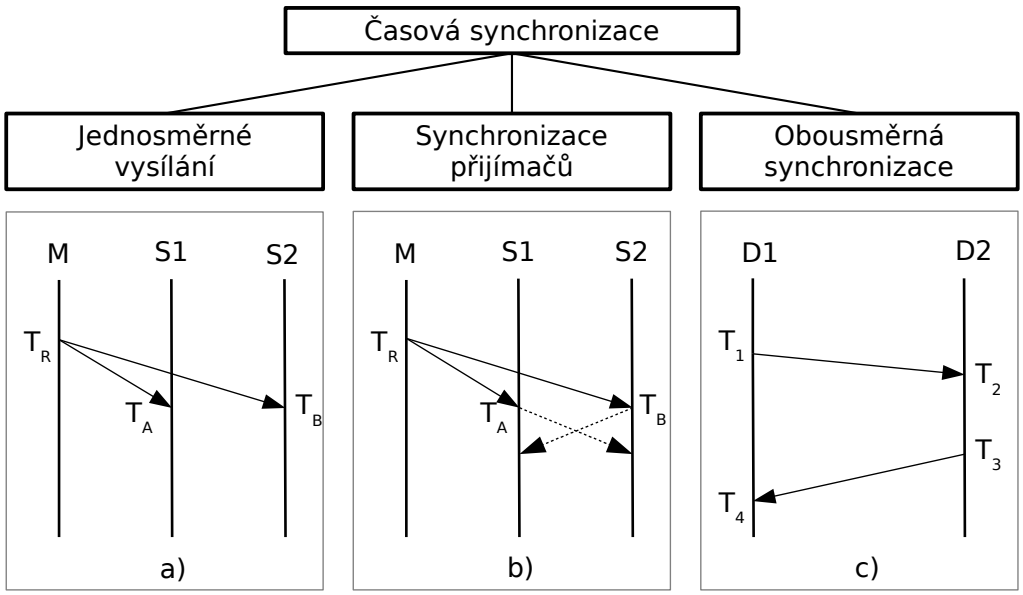
2 Současný stav problematiky

Pro synchronizaci hodin zařízení, jak v podobě informace o čase, tak v podobě informace o frekvenci, je využíváno několik základních principů. Volba principu synchronizace závisí na požadované přesnosti synchronizace vzhledem k šířce přenosového pásma a k energii, která je na synchronizaci potřeba.

2.1 Synchronizace obecně

Existuje několik základních způsobů synchronizace. V některých případech se skutečný čas události získává až zpětně. Hlavní rozdíl je především v tom, jestli se nějak kompenzuje zpoždění na přenosové cestě a jestli je tato kompenzace automatická nebo manuální.

Na obr. 2.1 jsou znázorněny základní způsoby synchronizace. M je časová osa hlavních hodin, S1 a S2 je časová osa podružných hodin. D1 a D2 jsou časové osy obecných zařízení. Může jít o komunikaci, kterou používá NTP, kdy se D1 zeptá hlavních hodin D2 na čas a hodiny odpoví. Zároveň se ještě získají další časové značky, ze kterých se vypočte zpoždění na přenosové cestě. Může jít ale i o princip, jako je v protokolu PTP, kde D1 představuje hlavní hodiny, které vyšlou synchronizační zprávu podružným hodinám D2, a ty se zeptají na dodatečné informace o zpoždění přenosové cesty. V tomto případě by ještě následovala zpráva z D1 obsahující T_4.



Obr. 2.1: Typy synchronizací hodin a) jednosměrná, b) jednosměrná s výměnou informace u přijímačů, c) vzájemná obousměrná.

2.1.1 Nárazová synchronizace způsobená událostí

Při nárazové synchronizaci se neprovádí pravidelné dotazování a korekce času. Metody se specializují na konkrétní události, nikoli na kontinuální časovou stupnici. Někdy pak stačí události pouze seřadit a absolutní čas není vůbec potřeba.

Trigger bus

Jeden ze způsobů synchronizace je vyslání spouštěcího impulzu. Tento impulz slouží k jednorázové synchronizaci a spuštění události na protistraně. Nevýhoda tohoto způsobu spočívá v nemožnosti automatické korekce zpoždění přenosové cesty a hodí se tedy v situacích, kde zpoždění přenosové cesty je zanedbatelné vzhledem k požadované přesnosti spouštění, nebo je zpoždění změřeno a kompenzováno.

Post-facto synchronizace

Post-facto synchronizace [1] je zajímavý koncept popsaný pro použití v senzorové síti, kde mají senzory omezené množství energie. V takovém prostředí není vhodné provádět kontinuální synchronizaci. Způsob post-facto synchronizace spočívá v tom, že senzory nejsou synchronizovány, ale mohou být kdykoli synchronizovány, když bude potřeba. Časová značka události se pak získá pomocí extrapolace aktuálního času do minulosti.

Synchronizace bez synchronizace hodin

V některých případech lze problém časové synchronizace řešit bez vlastní synchronizace. Významný mezník v této oblasti byla práce L. Lamporta, který vytvořil koncept virtuálních hodin [2]. Autor se v této práci zaměřil na získání pořadí událostí místo získání absolutního času událostí. V takovém systému má každý stroj vlastní hodiny, které jsou inkrementovány vždy, když je detekována událost. Při odesílání zprávy dalšímu stroji, odesílá i svoji časovou značku. Pokud nějaký stroj přijme zprávu, nastaví si svoji časovou značku větší, než byla časová značka v příchozí zprávě. Toto jednoduché schéma dostačuje k tomu, aby bylo možné zpětně rekonstruovat pořadí událostí z časových značek zpráv. V tomto systému také časové značky neukládají přímo čas, ale pouze pořadí událostí.

2.1.2 Nepřetržitá jednosměrná synchronizace

Jednosměrná synchronizace spočívá v generování signálu s danými parametry pouze jedním směrem. Při šíření signálu se projeví zpoždění přenosové cesty. Bez znalosti doby šíření signálu od zdroje k cíli není způsob jak toto zpoždění kompenzovat. Čas v cílovém zařízení bude o toto zpoždění posunutý.

Princip jednosměrné synchronizace lze rozdělit na tři základní mechanizmy – pouze syntonizace, pouze synchronizace a synchronizace i syntonizace dohromady.

Pouze syntonizace

V prvním případě vlastně nejde o synchronizaci času, protože se přenáší pouze signál s určitou frekvencí nebo se v cílovém zařízení frekvence obnovuje z nosného signálu. Bez znalosti zpoždění přenosové cesty nelze kompenzovat fázi signálu. Vzhledem k tomu, že ve většině případů můžeme zpoždění považovat za konstantní, je zdrojová frekvence stejná jako cílová. Z tohoto důvodu je metoda pojmenována pouze jako syntonizace, kdy se přenáší pouze „tón“ – frekvence a nikoli synchronizace, kdy se přenáší i informace o čase – „chronos“.

Do této kategorie patří různé signály s definovanou frekvencí a fází (např. 10 MHz, 1 PPS) nebo signály na přenosové cestě, ze kterých lze obnovit frekvenci nosné, jako synchronní Ethernet SyncE [3], synchronní telekomunikační přenosy apod.

Pouze synchronizace

Synchronizaci lze provádět po blíže nespecifikovaném synchronizačním kanálu, po kterém se posílají zprávy s informací o čase. Komunikační rychlost ani žádné jiné parametry tohoto kanálu pak nemají vazbu na přenášenou informaci o čase nebo frekvenci.

Synchronizace i syntonizace dohromady

Do této kategorie spadají metody přenosu informace o frekvenci a času zakódované do komunikačního kanálu, jehož parametry jsou odvozeny od přenášené informace o frekvenci a fázi. Jde například o signály s časovým kódem IRIG [4] nebo DCF77 [5]. U těchto signálů má každý bit přesně definovaný okamžik náběžné hrany každého pulzu a šířka pulzu určuje bitovou hodnotu. Určitý počet bitů pak nese časový kód.

Z těchto signálů tak jde obnovit informaci o frekvenci například sledováním náběžných hran. Šířka pulzu pak může například určovat informační hodnotu a z té je dekódována informace o čase. Tyto signály mohou být navíc modulovány pomocí AM modulace na nosném signálu, který je také odvozen od přesného zdroje hodin. Lze tak syntonizaci provést ještě před vlastním demodulováním signálu.

2.1.3 Speciální případy jednosměrné synchronizace

Společnou vlastností všech výše jmenovaných jednosměrných principů synchronizace je, že nelze automaticky kompenzovat zpoždění přenosové cesty. Existují ale i mechanizmy, jak v jednosměrném přenosu tuto kompenzaci provést.

Jedna z možností je získávat dodatečné informace a zároveň je získávat z více zdrojů. Touto cestou jdou všechny globální družicové polohové systémy, jako je třeba GPS nebo Galileo. Při příjmu signálu z více zdrojů lze spočítat pozici přijímače v prostoru vůči jednotlivým satelitům a tím kompenzovat časové zpoždění přenosové cesty. [6]

Druhým přístupem je Reference Broadcast Synchronization (RBS) [7]. Tato metoda byla navržena pro senzorové sítě a staví na požadavku, že nepotřebujeme absolutní přesnost časové synchronizace u jednotlivých zařízení, ale potřebujeme mít tato zařízení vzájemně synchronizovaná. Referenční node vysílá událost, ale již dále nekomunikuje. Ostatní nody se vzájemně doptají, kdy to bylo. Chyba synchronizace oproti referenci může být veliká, ale vzájemná chyba jednotlivých nodů je tímto minimalizována.

2.1.4 Obousměrná synchronizace

Nevýhoda jednosměrných metod spočívá v nemožnosti kompenzovat zpoždění přenosové cesty a tím získat přesný čas, a nebo v nutnosti zavést další veličiny do celého procesu synchronizace, jako je přesná poloha.

Obousměrná synchronizace spočívá ve výměně informací dvou zařízení, která takto nejen zjistí informaci o čase, ale odhadnou i zpoždění přenosové cesty. Toto zpoždění lze pak použít pro kompenzaci synchronizovaného času.

Existují dva základní principy. V prvním případě hlavní hodiny aktivně vysílají informaci o čase všem najednou a ostatní zařízení se pak doptávají na zpoždění přenosové cesty, např. PTP protokol v režimu multicast. V druhém případě se zařízení aktivně doptávají hlavních hodin, které bez dotazu neposílají žádná data, např. NTP protokol v režimu unicast.

2.1.5 Kombinované metody

V některých případech lze použít technologie původně oddělené a spojit jejich výhody. Lze tak například využít dohromady synchronní Ethernet SyncE s protokolem PTP. SyncE pak zajišťuje přesnou frekvenci a PTP přesný čas.

2.2 Synchronizace v bezdrátových senzorových sítích

Synchronizace hodin v bezdrátových sítích upřednostňuje jiná kritéria než synchronizace v drátových sítích. V bezdrátových sítích je navíc problém topologie jednotlivých senzorů, která je závislá na poloze senzoru v prostoru. Díky nativnímu broadcastu jsou zprávy jednotlivých senzorů přenášeny automaticky všem senzorům v dosahu. Přenos zprávy mezi dvěma senzory, které nejsou v dosahu, je realizován dalším senzorem. Tento senzor se chová jako prostředník a zprávu přepošle dál. Pokud by takové přeposílání zpráv bylo řešeno hrubou silou, byla by síť přeplněna, je tedy třeba implementovat algoritmy pro hledání cesty.

Kvůli omezenému množství energie a omezené šířce pásma musí komunikační protokoly řešit minimalizaci zpráv. Senzorové sítě mohou být jednoduché a mohou předpokládat přímou vzájemnou dosažitelnost

senzorů. U rozsáhlých sítí toto předpokládat nelze a je třeba řešit i inteligentní přeposílání zpráv. Všechny tyto aspekty znesnadňují synchronizaci hodin a každý senzor, který přeposílá zprávu přidává další zpoždění do přenosové cesty.

Pro bezdrátové senzorové sítě byly vyvinuty synchronizační algoritmy jako RBS, TPSN, Post-facto synchronizace a mnoho další. Vybrané metody budou popsány dále. I když se metody používané v bezdrátových sítích nedají přímo aplikovat na sítě drátové, jsou zajímavou inspirací pro další vývoj.

2.2.1 Timing-Sync Protocol for Sensor Networks (TPSN)

TPSN [8] popisuje protokol, který pracuje ve dvou fázích: fáze nalezení úrovní a fáze synchronizace. Cíl první fáze je vytvoření hierarchické topologie v síti, kde každému senzoru je přiřazena úroveň. V druhé fázi se senzory z úrovně *i* synchronizují se senzory z úrovně *i-1*. Na konci synchronizační fáze jsou všechny senzory synchronizovány ke kořenovému senzoru a je dosažena celková synchronizace sítě. Vlastní synchronizace je pak realizována klasickou dvoucestnou synchronizací.

2.2.2 Reference Broadcast Synchronization (RBS)

Protokol RBS [7] navrhuje synchronizační schéma pro senzorovou síť, které má vyloučit vliv zpoždění při vysílání u kořenového zařízení. Pro synchronizaci je tedy použito speciální zařízení, které pouze vysílá synchronizační zprávy, ale již žádné zprávy nepřijímá. Každý senzor v dosahu si pouze uloží čas přijetí této referenční zprávy. Senzory se pak synchronizují na základě výměny času přijetí této referenční zprávy. Vzhledem k tomu, že nedeterminismus vysílače synchronizační zprávy nemá žádný vliv na celkovou synchronizaci, je synchronizace přesnější než klasické dvoucestné synchronizace.

2.3 Synchronizace v počítačových sítích

Pro synchronizaci počítačových sítí bylo navrženo velké množství protokolů. Většina z nich používá bez-spojovou komunikaci založenou na výměně zpráv mezi klienty a jedním nebo více servery. Protokoly se liší především v tom, jestli je pro synchronizaci použit externí frekvenční normál apod. Další práce řeší základní modely synchronizace a navrhují statistické a heuristické modely pro redukci nedeterministického chování [9].

- Cristian vypozoroval, že při provedení velkého množství žádostí a odpovědí vznikne pravděpodobně jedna výměna, která nebude zatížena náhodnými zpožděními [10]. Tuto cestu lze identifikovat jednoduše tím, že její průchod trval nejkratší dobu.

- Marzullo popsal algoritmus, jak vypočítat odhad aktuálního času za použití *n* časových serverů za předpokladu, že méně než polovina z těchto serverů selhala [11]
- Mills vytvořil řídicí smyčku pro řízení frekvence lokálního oscilátoru tak, že se přibližuje referenci jak fází tak frekvencí [12].

2.3.1 Network Time Protocol

Z těchto poznatků například vychází Network Time Protocol (NTP) [13], který se v internetu stal de facto standardem. Protokol v sobě obsahuje škálovatelnost, robustnost a automatickou konfiguraci. Protokol umožňuje vytváření hierarchií hodin.

NTP je protokol pro synchronizaci hodin počítačových systémů v přepínané paketové síti. Protokol využívá Marzullův algoritmus pro výpočet nejpřesnějšího odhadu časové stupnice z více zdrojů. Poskytuje časovou stupnici UTC. Nejsou vysílány informace o časových zónách a o přepínání letního a zimního času. Tyto informace jsou nad rámec protokolu NTP. Pro zjednodušení byl vytvořen i protokol Simple Network Time Protocol (SNTP), který nevyžaduje uchovávání stavu systému po delší dobu. Tento protokol se používá především v embedded systémech.

2.3.2 Precise Time Protocol

Standard IEEE 1588, neboli Precise Time Protocol (PTP) [14], je protokol pro přesnou časovou synchronizaci v počítačových sítích. Synchronizace je založena na komunikaci master/slave. Master periodicky vysílá zprávy s časovou značkou a slave zařízení se podle těchto časových značek synchronizují. Pro zvýšení přesnosti není přesná časová značka odeslána v synchronizační zprávě, ale ve zprávě, která ji okamžitě následuje. Pokud existuje podpora HW, je možné přesnou časovou značku vložit do právě odesílaného paketu. Pro korekci časového zpoždění přenosové cesty vysílá slave zařízení dotaz na délku přenosové cesty. Pro srovnání je zde protokol pouze zmíněn. V kapitole 2.4 následuje podrobnější popis celého chování.

2.3.3 IEEE 802.1AS

IEEE 802.1AS [15] je standard popisující protokol pro přesnou časovou synchronizaci. Protokol je součástí větší rodiny protokolů pro Audio Video Bridging, IEEE 802.1BA.

Vychází ze standardu IEEE 1588, ale má zjednodušené zprávy. Navíc se tento standard snaží být univerzálnější vzhledem k linkové vrstvě. Zatímco IEEE 1588 je primárně určen pro Ethernet (IEEE 802.3), tento protokol lze bez problémů provozovat na jiných linkových vrstvách.

Z protokolu IEEE 1588 si zachovává možnost autokonfigurace, takže libovolné zařízení lze za běhu připojit nebo odpojit a všechna ostatní se

v případě potřeby překonfigurují na jiné hlavní hodiny. Ve svém důsledku lze IEEE 802.1AS brát jako podmnožinu IEEE 1588 a byl dokonce schválen jako profil pro IEEE 1588.

Typické použití tohoto protokolu je především v multimédiích. Lze jej například použít v chytrých Wi-Fi reproduktorech, kde je potřeba synchronizace přesnější než 10 µs, nebo v jiných systémech pro šíření multimediálního obsahu, kde je potřeba synchronizovat více kanálů např. podle standardu AES11 [16].

2.3.4 White Rabbit

Projekt White Rabbit vznikl především pro účely časové synchronizace na částicovém urychlovači v CERN. V tomto projektu je navržen celý koncept časové synchronizace, který se snaží využít stávajících standardů a dosáhnout co největší přesnosti a spolehlivosti. Cílem projektu je dosáhnout přesnosti lepší než 1 ns. Pro časovou synchronizaci využívá protokol IEEE 1588. Všechny síťové prvky mají redundantní spojení, ale na rozdíl od klasického PTP protokolu jsou aktivní všechny redundantní spoje. V případě výpadku tak není třeba čekat na autokonfiguraci, protože se okamžitě začne používat jiný aktivní datový spoj. Projekt kombinuje synchronizaci se SyncE, díky které je možné přesné navázání frekvence. U redundantních linek je definováno chování odvození z více zdrojů.

Provoz je zajištěn po optických kabelech a je v něm numericky korigována asymetrie způsobená různou rychlostí šíření světla při různých vlnových délkách. Součástí návrhu je i přesné měření fázové odchylky technikou Dual Mixer Time Difference, DMTD, díky které je možné měřit fázový rozdíl v jednotkách ps. Toto měření je prováděno automaticky a tím je možné celou komunikace korigovat.

Díky koncepci protokolu a použitým korekcím lze například na 2 km dlouhém optickém vedení dosáhnout jitter menší než 80 ps a to i v případě, že je optické vedení podrobeno tepelnému namáhání, které by se jinak projevilo v nanosekundových odchylkách [17].

2.3.5 Time-Triggered Protocol a TTEthernet

Time-Triggered Protocol (TTP) [18] je jeden z představitelů průmyslové sběrnice reálného času. Pro zajištění funkce využívá principu Time Division Multiple Access (TDMA). K funkci je potřeba časová synchronizace zařízení, aby byly jednotlivé sloty vysílány ve vyhrazený čas. Časová synchronizace je navržena tak, aby se celý systém vzájemně synchronizoval bez nutnosti centrální autority, která poskytuje informaci o čase. TTP dále obsahuje zprávy, které informují každé korektně fungující zařízení o případných chybách v síti.

TTEthernet je rozšíření tohoto protokolu do sítě Ethernet. TTEthernet je jeden z příkladů průmyslové sběrnice založené na Ethernetu. Protokol

sjednocuje real-time komunikaci a obecnou komunikaci do jednoho komunikačního protokolu. TTEthernet je plně kompatibilní s původním konceptem Ethernetu a libovolné zařízení může využívat tuto sběrnici pro obecné účely bez úpravy. Pro zajištění funkce jsou potřeba speciální síťové prvky, které dodržují časové sloty a do zbylého prostoru mapují stávající tok dat na síti. Pro zajištění funkčního TDMA je třeba celý systém synchronizovat. Vzhledem k tomu, že průchod zprávy tímto systémem je vždy konstantní, není třeba používat složité protokoly typu PTP, ale stačí jednodušší. Není totiž nutné neustále měřit zpoždění přenosové cesty. Pokud je potřeba celý systém synchronizovat na externí hodiny s absolutní časovou stupnicí, je pro takovou synchronizaci použit PTP protokol.

2.4 Precise Time Protocol

Precise Time Protocol (PTP) [14] je protokol pro přesnou časovou synchronizaci. Synchronizace je založena na komunikaci master/slave. Hlavní hodiny (master) periodicky vysílají zprávy s časovou značkou a podružné hodiny (slave) se podle těchto časových značek synchronizují. Pro korekci časového zpoždění přenosové cesty vysílají podružné hodiny dotaz na délku přenosové cesty. Každé zařízení může mít více fyzických portů a každý port může být v jiném režimu. Na některých portech může zařízení komunikovat jako master nebo být pasivní a na jednom se může chovat jako slave.

Režimy jednotlivých portů určují, jestli jsou nebo nejsou vysílány synchronizační zprávy. Režim master znamená, že na daném portu jsou pravidelně vysílány synchronizační zprávy. Režim slave znamená, že dané zařízení se synchronizuje s pomocí synchronizačních zpráv a dále se doptává na zpoždění přenosové cesty. Pasivní režim je pouze na takových portech, kde dané zařízení je synchronizováno z jiného zdroje. Nemá tedy smysl, aby se synchronizovalo ještě z PTP. Tato zařízení mohou být například náhradní hlavní hodiny. Pokud ty hlavní přestanou fungovat, přejde zařízení z pasivního režimu do režimu master. V pasivním režimu nevysílá zařízení žádné synchronizační pakety kromě paketů pro zjištění zpoždění linky v režimu P2P, který bude popsán dále.

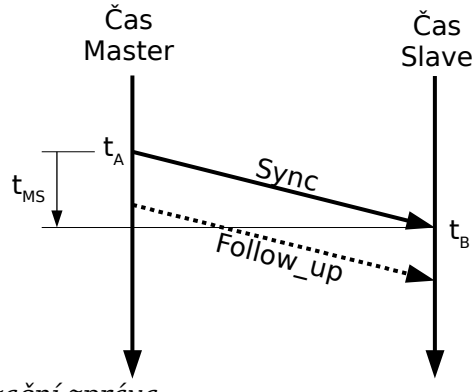
PTP protokol je určen pro použití především pro lokální sítě. Existuje několik standardizovaných profilů pro různé účely. Pro měřicí aplikace se využívá nejčastěji multicast, pro telekomunikační účely naopak unicast. Protokol se stále vyvíjí a v roce 2008 vyšla jeho aktualizace, která implementuje nové poznatky.

Protokol obsahuje autokonfigurační vlastnosti tzv. Best Master Clock Algorithm (BMCA), který slouží pro výběr nejlepších hlavních hodin, na které se potom všichni sesynchronizují. Tento mechanizmus lze využít i pro zabezpečení redundance hlavních hodin, kdy si při výpadku automaticky ostatní zařízení vyberou náhradní hlavní hodiny.

2.4.1 Princip synchronizace

Synchronizace spočívá ve dvou hlavních procesech, které běží paralelně. První je aktivní vysílání synchronizačních zpráv a druhý je měření zpoždění přenosové cesty.

První proces je periodické zasílání synchronizačních zpráv z hlavních hodin do podružných hodin, viz obr. 2.2. Podružné hodiny si mohou podle těchto zpráv nastavit časovou stupnici a při dlouhodobém sledování zkalibrovat frekvenci svých vnitřních hodin. Tímto způsobem ale nelze přesně zjistit rozdíl časových základen, protože je zde obsaženo časové zpoždění potřebné na přenesení a zpracování zprávy. Hlavní hodiny v synchronizační zprávě zasílají časovou značku, kdy byla tato zpráva odeslána. Bez speciální hardwarové podpory to nelze provést přesně, protože je nutné měnit právě odesílanou zprávu. Proto je v protokolu definována i možnost, kdy může být součástí zprávy jen přibližná časová značka. Ta přesná bude odeslána až po skutečném odeslání synchronizační zprávy pomocí doprovodné zprávy.



Obr. 2.2: Synchronizační zpráva

Poměr frekvencí hlavních hodin a podružných hodin pak lze určit ze dvou synchronizačních zpráv vzdálených $N>0$ synchronizačních intervalů.

$$\frac{f_{\mathrm{M}}}{f_{\mathrm{S}}}=\frac{t_{\mathrm{B}_N}-t_{\mathrm{B}_0}}{t_{\mathrm{A}_N}-t_{\mathrm{A}_0}}\,, \tag{2.1}$$

kde f_{M} je frekvence hlavních hodin, f_{S} je frekvence podružných hodin, t_{A} je čas odeslání zprávy a t_{B} je čas přijetí. Tento poměr lze použít pro kalibraci časové základny na straně podružných hodin.

Hodnota zpoždění master-slave t_{MS} se vypočte ze vztahu

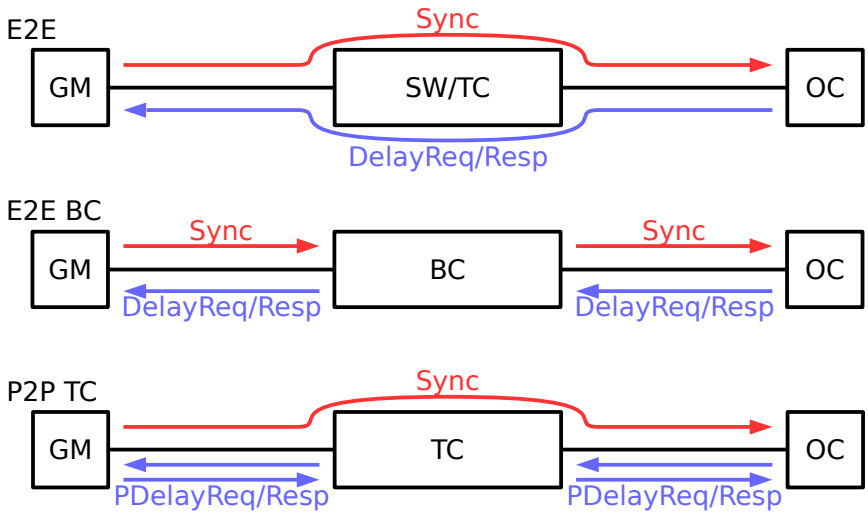
$$t_{\mathrm{MS}}=t_{\mathrm{B}}-t_{\mathrm{A}}\,. \tag{2.2}$$

Druhý proces slouží ke zjištění zpoždění přenosové cesty. V protokolu jsou popsány dva základní principy. První princip je měření od konce do

konce, označovaný jako End to End, nebo zkráceně E2E a druhý způsob od souseda k sousedu označovaný jako Peer to Peer, nebo zkráceně P2P.

První způsob, E2E, nevyžaduje na cestě žádné speciální síťové prvky. Dotaz na zpoždění je vždy posílán hlavním hodinám. Hlavní hodiny se na zpoždění přenosové cesty neptají, protože těch cest je několik – ke každým podružným hodinám ideálně jedna.

Druhý způsob vyžaduje speciální síťové prvky zvané Transparent Clock (TC), které si měří zpoždění mezi sebou. Tento způsob měření probíhá v obou směrech, tedy i hlavní hodiny si měří zpoždění. Neměří zpoždění k podružným hodinám, ale k nejbližšímu síťovému prvku. Vše je znázorněno na obr. 2.3.



Obr. 2.3: Měření zpoždění přenosové cesty v různých režimech E2E, E2E BC a P2P TC. GM – Grand Master Clock – hlavní hodiny, BC – Boundary Clock – hraniční hodiny, TC – Transparent Clock – transparentní hodiny, OC – Ordinary Clock – obecné hodiny, SW – Switch – síťový přepínač

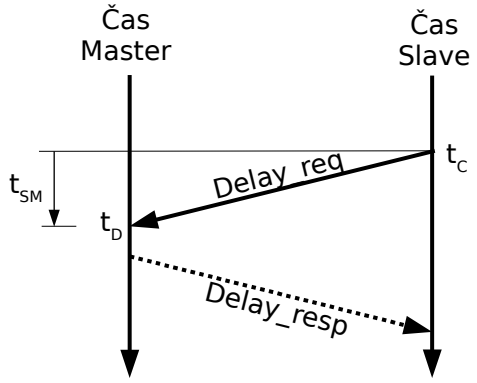
V režimu E2E s použitím běžného síťového přepínače nebo transparentních hodin vychází synchronizační zpráva z hlavních hodin a je doručena přes všechny prvky až do podružných hodin. Všechny podružné hodiny se ptají na délku přenosové cesty a ptají se opět přímo hlavních hodin. Tento mechanizmus má výhodu v univerzálním použití a nevyžaduje speciální síťové prvky na cestě. Speciální síťové prvky navíc mohou vkládat do komunikace korekční informace, kde předají informaci o zpoždění, které sami způsobily. Pokud je ale velký počet podružných hodin, tak může být počet dotazů na zpoždění přenosové cesty neúměrně veliký. Další nevýhoda celého mechanizmu je, že pokud vypadnou hlavní hodiny a jsou vybrány náhradní, je třeba znovu změřit zpoždění všech cest k novým hlavním hodinám. Odhad zpoždění nelze přesně získat z první naměřené hodnoty

a je třeba provést statistické vyhodnocení a filtrování. K tomuto vyhodnocení je potřeba více vzorků a jejich získání může trvat dlouho.

Pokud je použit koncept hraničních hodin, označovaných jako Boundary Clock (BC), situace se změní. Hlavní hodiny synchronizují pouze první BC, to pak déle distribuuje čas ostatním zařízením. Jejich dotazy na délku přenosové cesty směřují opět jen k BC, které přímo odpovídá. BC si samo měří zpoždění přenosové cesty k hlavním hodinám. V tomto režimu nemůžou být hlavní hodiny přetíženy, protože jediný, kdo s nimi komunikuje, je BC. Rekonfigurace při výpadku BC je také snadnější. Největší problém celého konceptu spočívá ve snížení přesnosti synchronizace. Každé BC se nejprve musí synchronizovat na své hlavní hodiny. BC mohou být zřetězeny v libovolné hierarchii. Synchronizační algoritmus a schopnost BC udržet stabilní časovou stupnici pak ovlivňují každé další návazné hodiny.

Poslední hlavní způsob předávání informace o zpoždění přenosové cesty je režim P2P. Tento režim vyžaduje speciální síťové prvky, protože každý port každého zařízení musí být zapojen tak, že může přistupovat k maximálně jednomu portu jiného zařízení. Je tedy potřeba provozovat například transparentní hodiny (TC), které se pro všechny pakety, kromě PTP, chovají jako obyčejný síťový přepínač. Tento režim má především výhodu v možnosti rychlé rekonfigurace při výpadku libovolného prvku. Celková doba zpoždění se vypočte jako součet dílčích zpoždění a pokud vypadne některý spoj, tak se použije jiný, který je ale již změřený, takže se provede pouze jiný součet již známých hodnot. Zásadní výhoda tohoto režimu je také zvýšená přesnost oproti použití BC, protože synchronizační zprávy se pouze přeposílají a neaplikuje se na ně žádné vyhodnocení a filtrování. Přesnost synchronizace tak není ovlivněna filtračním algoritmem, ani přesností synchronizace vlastních TC. Nevýhodou tohoto režimu je nutnost použít speciální síťové prvky. Problematická je také sama funkčnost transparentních hodin, protože pozměňují vnitřní obsah zprávy, ale stále ponechávají původního původce zprávy, což ze síťového pohledu není korektní chování.

Režim měření zpoždění E2E je znázorněn na obr. 2.4. Podružné hodiny se doptávají na délku přenosové cesty a z přijatých dat ji mohou spočítat. Vše funguje přesně pouze za předpokladu, že jsou zpoždění v obou směrech symetrická.



Obr. 2.4: Měření zpoždění přenosové cesty

Hodnota zpoždění slave-master t_{SM} se vypočte ze vztahu

$$t_{\mathrm{SM}} = t_{\mathrm{D}} - t_{\mathrm{C}} \,. \tag{2.3}$$

Je třeba brát v potaz, že hodnoty t_{MS} a t_{SM} nemají sami o sobě vypovídající význam. Regulační algoritmus podružných hodin by se měl snažit korigovat časovou stupnici tak, aby tyto hodnoty vycházely stejné, protože předpokládáme stejné časy při komunikaci oběma směry. Z tohoto důvodu definujeme další hodnotu t_{MPD} jako průměrnou hodnotu zpoždění označovanou ve standardu jako meanPathDelay, která se vypočte ze vztahu

$$t_{\mathrm{MPD}} = \frac{t_{\mathrm{SM}} + t_{\mathrm{MS}}}{2} = \frac{(t_{\mathrm{B}} - t_{\mathrm{A}}) + (t_{\mathrm{D}} - t_{\mathrm{C}})}{2} = \frac{(t_{\mathrm{B}} - t_{\mathrm{C}}) + (t_{\mathrm{D}} - t_{\mathrm{A}})}{2} \,. \tag{2.4}$$

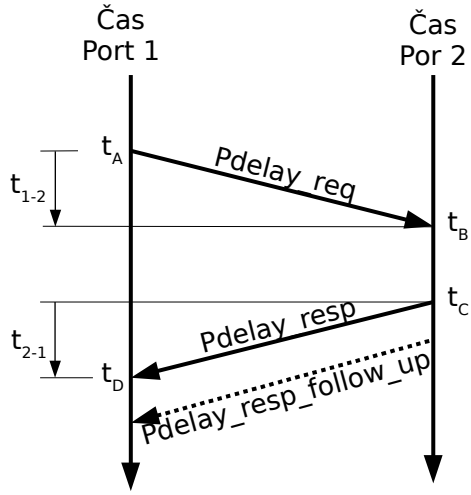
Rozdíl časů hlavních a podružných hodin t_{OFM} označovaný jako offsetFromMaster se vypočte ze vztahu

$$t_{\mathrm{OFM}} = t_{\mathrm{B}} - t_{\mathrm{A}} - t_{\mathrm{MPD}} \,. \tag{2.5}$$

Tento údaj slouží k určení přesnosti synchronizace. Regulační algoritmus v podružných hodinách by se měl snažit tuto hodnotu minimalizovat.

Režim měření zpoždění P2P probíhá podobně, ale proto, že ho může vyslat kdokoli, obsahuje všechny čtyři časové značky. Stejné časové značky získalo zařízení slave ze synchronizační zprávy a z E2E zprávy pro zjištění zpoždění.

Časové značky mají tedy stejný význam jako v předchozím případě a zpoždění přenosové cesty se vypočte podle vztahu 2.4. Hlavní rozdíl je v tom, že se na zpoždění přenosové cesty ptají všechna zařízení, tedy i hlavní hodiny. Dokonce se na něj ptají i jinak pasivní zařízení, která čekají v podobě zálohy.

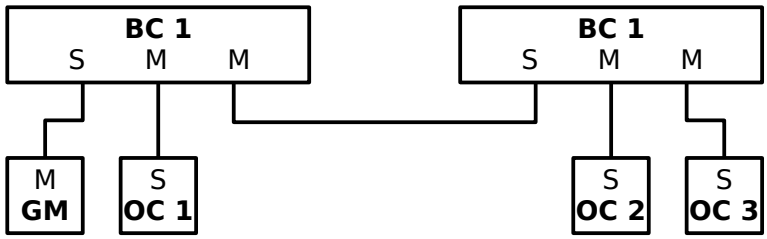


Obr. 2.5: Měření zpoždění linky

2.4.2 Síťové topologie

Pokud je požadována rozsáhlejší síť, je možné vložit další síťové prvky do komunikačního kanálu. Každý síťový prvek s sebou ale nese dodatečné zpoždění přenosové cesty a hlavně se negativně podílí na jitteru celkového zpoždění.

Standard IEEE 1588 popisuje hierarchickou master-slave architekturu pro distribuci hodin. Systém pro distribuci hodin sestává z komunikačního média a jednoho nebo více hodin. Základním typem hodin je Ordinary Clock (OC). Jde o zařízení s jedním síťovým připojením, které se podle požadavků může chovat jako zdroj (master) nebo příjemce (slave) hodin.



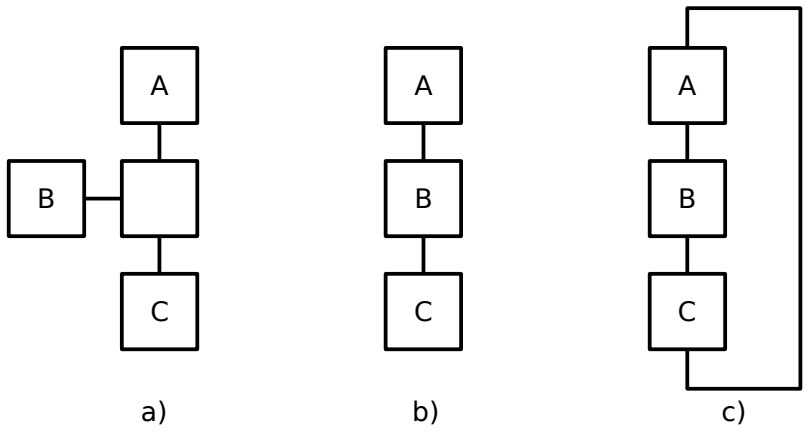
Obr. 2.6: Hierarchie za použití boundary clock [14] BC – Boundary Clock, OC – Ordinary Clock, M – Master, S - Slave

U složitějších sítí počítá původní koncepce pouze s tzv. Boundary Clock (BC), které obsahují více komunikačních portů a které se chovají na jednom portu jako slave a na ostatních jako master (obdoba z hierarchie NTP protokolu). Pokud je PTP protokol používán spíše v hvězdicové topologii, není v cestě od zdroje k cíli velké množství takových hodin. V průmyslových systémech se ale často používá lineární nebo kruhová topologie viz obr. 2.7. Takováto hierarchie zbytečně přidává při použití každého BC jeho chybu

synchronizace. Tato chyba pak neúměrně narůstá s počtem použitých BC v cestě.

V nové koncepci bylo proto navrženo zařízení tzv. Transparent Clock (TC). TC k přesné časové značce přidává informaci o tom, jak dlouho trval průchod tímto TC. Každé další TC udělá totéž a koncové zařízení má tedy k dispozici tři časové údaje. První údaj je čas odeslání paketu, druhý údaj je čas přijetí paketu a třetí údaj je čas strávený ve všech TC dohromady. Protože zbytek komunikace se odehrává například po metalickém vedení, je čas potřebný pro překonání takového vedení konstantní. Z těchto údajů lze spočítat zpoždění přenosové cesty s minimalizací vlivu nedeterministického chování spínacích prvků. Nevýhoda tohoto způsobu spočívá v porušování zavedeného principu, kdy paket procházející sítí již není měněn. Tento paket je sice změněn, ale stále se tváří jako od původního zdroje.

V dalších navazujících normách [15] je proto doporučován postup, kdy se opět používají pouze BC, ale v podobném režimu jako TC. BC tedy pouze přeposílá relevantní synchronizační pakety, stejně jako TC, ale mění zdrojovou adresu na sebe, stejně jako klasický BC. Díky tomu, že se BC nesnaží nejprve synchronizovat a pak nezávisle distribuovat informaci o čase, neovlivňuje svým synchronizačním algoritmem přesnost návazných hodin.



Obr. 2.7: Používané topologie v ethernetových sítích. a) Hvězdicová topologie b) Lineární topologie c) Kruhová topologie

2.4.3 Obecné vlastnosti protokolu

Protokol PTP může pracovat jak s relativní časovou stupnicí, tak i přímo s absolutní stupnicí UTC. Podle potřeb synchronizace lze synchronizační interval volit v rozmezí od desítek ms do desítek s.

Zprávy protokolu PTP mohou být odesílány jak v režimu multicast, tak v režimu unicast. Pro komunikaci mohou využívat například nespojový protokol UDP přes IPv4 nebo IPv6. Protokol dále rozděluje dva typy zpráv –

události a obecné zprávy. Ke každé zprávě typu událost se ukládá časová značka na přijímací i vysílací straně. Následně se pomocí obecných zpráv tyto časové značky přenesou. Aby se tyto typy zpráv rozdělily, jsou při použití UDP vysílány pakety událostí na portu 319 a obecné zprávy na portu 320.

Zprávy pro synchronizaci času by měly být co nejkratší, aby procházely přes síťové přepínače co nejrychleji. Ve standardu IEEE 1588 je tedy definován i způsob přenosu zpráv přímo ve druhé vrstvě IEEE 802.3 Ethernet.

2.4.4 Určení nejlepších hlavních hodin

Dostatečnou robustnost protokolu zajišťuje automatický algoritmus pro výběr nejlepších hodin – BMCA. Každé hodiny v systému mají definovány jednotlivé kvalitativní parametry.

- *Identifikátor* – univerzální a jedinečný identifikátor hodin. Ve většině případů je odvozen od MAC adresy zařízení.
- *Kvalita* – parametr kvalita v sobě zahrnuje informace o typu hodin a o zdroji, ze kterého získává informaci o čase. Pokud je nějaké zařízení synchronizováno přes GPS, má lepší kvalitu než volně běžící krystalový oscilátor.
- *Priorita* – ručně nastavená hodnota priority hodin pro zvýhodnění konkrétních hodin v systému
- *Rozptyl* – hodnota, která by měla ukazovat naměřený rozptyl parametrů vzhledem k referenci.

BMCA pak podle důležitosti jednotlivých parametrů srovnává, které hodiny jsou lepší, a ty se stanou masterem.

Funkce BMCA je distribuovaná a celý systém se shodne na jediném master zařízení. Pokud například aktuální master přestane vysílat, začnou nějaké hodiny po vypršení timeoutu vysílat své časové značky jako master. Každé další hodiny pak hodnotí tyto příchozí zprávy. Pokud zpráva pochází z horších hodin (podle BMCA), začnou tyto hodiny vysílat své přesnější časové značky. Pokud zpráva pochází z lepších hodin, zařízení přestane vysílat svoje časové značky a na tyto hodiny se synchronizuje. Celý proces pokračuje tak dlouho, až v systému zbude pouze jeden master.

Při použití BC v síti se složitější topologií je potřeba řešit i nejlepší cestu. V síti může být kruh, nebo může být jinak složitěji zapojená. BMCA řeší i tuto situaci a vyhodnocuje i cestu, po které paket prošel. Nakonec se tedy systém ustálí s výběrem hlavních hodin a každé hodiny v systému vědí i nejlepší cestu k nim.

3 Stanovení cílů práce

Hlavním cílem této práce je nalezení způsobu měření různých zdrojů chyb synchronizace. Práce se věnuje především protokolu IEEE 1588, který je popsán v kapitole 2.4. Tento protokol se snaží některé zdroje chyb potlačit využitím speciálního hardwarového vybavení, což není vždy možné. Cílem této práce je především přesné měření zpoždění přenosové cesty. Pro splnění hlavního cíle práce bylo nezbytné splnit následující dílčí fáze:

a) *Navrhnout a realizovat metodu měření zpoždění přenosové cesty.*

   Změny zpoždění přenosové cesty se po způsobu získávání časových značek paketů nejvíce projevují na chybách synchronizace. Bez vyloučení vlivů algoritmů synchronizace a bez přesného měření není možné odhadnout chování synchronizace v dané síti.

b) *Navrhnout a implementovat způsob ověření správnosti protokolu.*

   Získávání časových značek může být přesné a přenosová cesta může mít konzistentní zpoždění, ale pokud jsou chyby v implementaci některého zařízení, které používá protokol pro přesnou časovou synchronizaci, může se narušit synchronizace celého systému. Zjištění takových chyb vyžaduje zevrubnou kontrolu výměn paketů, která se těžko provádí ručně. V době tvorby práce neexistoval nástroj, který by to provedl automatizovaně a přímo za běhu systému.

c) *Navrhnout a realizovat metodu pro eliminaci změn zpoždění přenosové cesty při různých zátěžích.*

   Na změny zpoždění paketů má velký vliv zatížení sítě. Pro přenos velkého objemu dat a zároveň pro zajištění synchronizace je třeba definovat způsob komunikace po sdíleném médiu.

d) *Implementovat protokol IEEE 1588 do různých systémů, ověřit chování a dosažitelnou přesnost.*

   Vytvoření implementace pro různé systémy je důležité pro získání přehledu o chování protokolu a o vlastnostech jednotlivých systémů z hlediska udržení přesné časové stupnice.

4 Definice základních pojmů

Pro potřeby časové synchronizace zavedeme pojem hodiny zařízení. Hodiny zařízení jsou zde myšleny jako zástupný výraz za časovou základnu, ze které lze odvodit aktuální čas a frekvenci. Tento nadřazený pojem je zaveden proto, že v některých synchronizačních algoritmech je jedna z těchto dvou složek hodin zanedbána a udržuje se pouze frekvence, nebo pouze čas.

4.1 Definice hodin zařízení

Každé zařízení udržuje své vlastní hodiny. Tyto hodiny jsou soubor softwarových a hardwarových komponent. Teoreticky jde o funkční blok, který generuje periodický časový signál. Prakticky je ale tento signál ovlivněn šumem, např. kolísáním fáze oscilátoru, a proto není časový signál zcela periodický – je pseudo-periodický. Hodiny se skládají ze tří základních částí: oscilátor, čítač a prostředky pro zobrazení nebo záznam výsledků [19].

4.1.1 Časový signál

Obecná rovnice popisující pseudo-periodický průběh, který modeluje časový signál $s(t)$ na výstupu hodin je dán

$$s(t)=A\sin\phi(t)\,, \tag{4.1}$$

kde A je konstantní koeficient amplitudy (předpoklad, že změny amplitudy jsou zanedbatelné) a $\Phi(t)$ je celková okamžitá fáze popisující ideální lineární fázi zvětšující se s t a libovolný frekvenční drift nebo náhodné fázové fluktuace. [20]

Ideální model $\Phi_{id}(t)$ lze zapsat jako

$$\Phi(t)=2\pi\nu_{\mathrm{nom}}t\,, \tag{4.2}$$

kde ν_{nom} nazýváme nominální frekvence.

Aktuální časové signály $\Phi(t)$ jsou modelovány jako:

$$\Phi(t)=\Phi_0+2\pi\nu_{\mathrm{nom}}(1+y_0)t+\pi D\nu_{\mathrm{nom}}t^2+\varphi(t)\,, \tag{4.3}$$

kde Φ_0 je počáteční fázový offset, y_0 je dílčí frekvenční offset od nominální frekvence ν_{nom}, D je lineární dílčí frekvenční drift (reprezentující stárnutí oscilátoru), $\varphi(t)$ je složka náhodných fázových odchylek.

Aktuální frekvence je pak dána jako

$$\nu(t)=\frac{1}{2\pi}\frac{d\Phi(t)}{dt}\,, \tag{4.4}$$

kde $\Phi(t)$ je celková okamžitá fáze.

Častý model pro charakterizaci $\nu(t)$ je dán vztahem

$$\nu(t)=\nu_{\mathrm{nom}}+\Delta\nu+D\,\nu_{\mathrm{nom}}t+\frac{1}{2\pi}\frac{d\varphi(t)}{dt}\,, \tag{4.5}$$

kde $\Delta\nu$ reprezentuje frekvenční offset od nominální hodnoty ν_{nom}, D je dílčí lineární frekvenční drift, který nejvíce popisuje stárnutí oscilátoru, $\varphi(t)$ je náhodná fázová odchylka modelující fázový šum oscilátoru.

4.1.2 Časová funkce

Obecná časová funkce $T(t)$ hodin je definována ve smyslu celkové okamžité fáze jako

$$T(t)=\frac{\Phi(t)}{2\pi\,\nu_{\mathrm{nom}}}\,, \tag{4.6}$$

kde ν_{nom} reprezentuje nominální frekvenci oscilátoru. Pro ideální hodiny pak vychází očekávaný vztah

$$T_{id}(t)=t\,. \tag{4.7}$$

4.1.3 Funkce časové chyby

Offset hodin od reference můžeme nazvat chybou synchronizace. Chybu synchronizace lze také vyjádřit jako závislost počátečního časového offsetu, frekvenčního offsetu, frekvenčního driftu a náhodného šumu podle následujícího vzorce [21], [22]. Pro dané hodiny je chyba času $TE(t)$ (někdy udávaná jako $x(t)$) mezi vlastním časem $T(t)$ a referenčním časem $T_{ref}(t)$ dána jako

$$TE(t)=x(t)=T(t)-T_{ref}(t)\,. \tag{4.8}$$

Z definice časové chyby (4.8) a z modelu $\Phi(t)$ (4.3) lze odvodit model reprezentující časovou chybu jako

$$TE(t)=x_0+y_0t+\frac{D}{2}t^2+\frac{\varphi(t)}{2\pi\nu_{\mathrm{nom}}}\,. \tag{4.9}$$

kde $TE(t)$ je celková chyba synchronizace, x_0 je počáteční chyba synchronizace, y_0 je počáteční chyba syntonizace od nominální frekvence ν_{nom}, D je frekvenční drift a $\varphi(t)$ jsou náhodné fázové odchylky.

Chyba počáteční synchronizace x_0 je konstantní časový offset k referenci. Chyba je závislá na konečné přesnosti měření a na šumu [21].

Chyba počáteční syntonizace y_0 je chyba závislá lineárně na čase. Bez resyntonizace by tato chyba nejvíce ovlivňovala celkovou chybu synchronizace. Kvůli tomu je zapotřebí periodicky syntonizovat hodiny s referencí [21].

Citlivost na prostředí je další největší zdroj chyby synchronizace. Je dobré umět rozlišit jak vlastnosti zařízení (např. stárnutí), tak vlivy okolí (např. teplota).

4.2 Synchronizace hodin

Synchronizace je problém nastavení N hodin tak, aby v čase $t>t_0$ byl vzájemný fázový offset všech hodin menší než předem určené K. Toho lze dosáhnout kompenzací hodnot fázového offsetu Φ_0 nebo frekvenčního offsetu y_0 dle rovnice (4.3). Synchronizace je tedy minimalizace vzájemného fázového offsetu hodin pod určenou mez.

Pro dosažení synchronizace lze kompenzovat fázový offset hodin. Kompenzace fázového offsetu má ale několik nevýhod. Hodiny stále běží s jinou nominální frekvencí, protože není kompenzován frekvenční offset. Tento způsob synchronizace navíc způsobuje, že fáze $\Phi(t)$ není pouze rostoucí a její hodnota se může znovu opakovat.

Synchronizaci je možné provést i korekcí frekvenčního offsetu jednotlivých hodin. Pokud bude korigovaný frekvenční offset vždy větší než 0, neboli dílčí frekvenční offset $y_0>-1$ dle rovnice (4.3), bude výsledná časová osa vždy rostoucí. Navíc se díky kompenzaci frekvenčního offsetu může prodloužit perioda jednotlivých cyklů synchronizace pro zachování stejné chyby.

4.3 Požadavky na synchronizaci

V praxi nemusí být synchronizace hodin tak striktní. Je možné použít jednodušší prostředky pro dosažení potřebného cíle.

V nejstriktnějším případě synchronizace poskytuje aproximaci absolutní časové stupnice (např. TAI nebo UTC). To klade vysoké nároky nejen na vlastní proces synchronizace, ale i na vlastnosti hlavních hodin, od kterých se absolutní časová stupnice odvozuje.

Absolutní časová stupnice není vždy potřeba a je možné využít stupnici relativní. Relativní stupnici je možné využít například při sledování akustické nebo seismické vlny, kdy potřebujeme toto vlnění přesně monitorovat, ale již není třeba přesně znát čas, kdy daná událost nastala.

Největší zjednodušení při požadavcích na synchronizaci nastane ve chvíli, kdy nás zajímá pouze pořadí událostí, ale již ne čas, kdy událost nastala.

Vzhledem k těmto požadavkům je třeba volit prostředky synchronizace. Pro průmyslové a laboratorní účely je v mnoha případech plně dostačující relativní časová stupnice a není třeba celý systém dále synchronizovat s absolutní stupnicí.

4.4 Časová zpoždění v synchronizaci

Nedeterministická zpoždění v šíření zpráv mohou být mnohem větší než požadovaná přesnost. Z tohoto důvodu musejí být tato zpoždění analyzována a kompenzována. Zpoždění, která mohou vzniknout, lze rozdělit do následujících kategorií [23].

- *vytvoření zprávy* – čas potřebný pro vytvoření zprávy a její předání podřazené Media Access Controller (MAC) vrstvě na vysílací straně. Tato doba závisí na vytížení procesoru a na režii použitého systému
- *přístup k médiu* – přístup k médiu je doba strávená v MAC vrstvě, po kterou zpráva čeká na své odeslání. Může se jednat o čekání na vysílací rámec nebo na volný komunikační kanál
- *vysílání zprávy* – čas potřebný pro vlastní vyslání zprávy. Tato doba je závislá na přenosové rychlosti a na délce zprávy
- *čas šíření* – čas potřebný pro překonání vzdálenosti mezi komunikujícími stranami v přenosovém médiu
- *přijímání zprávy* – čas potřebný pro příjem celé zprávy. Jde o stejný čas jako o dobu vysílání, pouze na přijímací straně
- *zpracování zprávy* – čas potřebný pro zpracování přijaté zprávy. Jeho charakteristiky jsou stejné jako pro čas potřebný pro vytvoření zprávy

Jednotlivá časová zpoždění lze odhadnout pomocí synchronizačního protokolu. Některé typy zpoždění lze úplně vyřadit pomocí specializovaného hardware, který získává časové značky na úrovni bližší vlastnímu fyzickému přenosu zprávy. Časové značky jsou tak například získány v okamžiku skutečného vyslání prvního slova zprávy a v okamžiku přijmutí prvního slova zprávy. Čas šíření zprávy lze změřit a kompenzovat, ostatní zdroje nejistot jsou vyřazeny.

Přesnější časové synchronizace narážejí na vlastní minimální rozlišovací schopnost (granularitu) hodin a přesnost určení časové značky. V takovém případě je granularita limitující faktor synchronizace.

4.5 Parametry hodnocení synchronizace

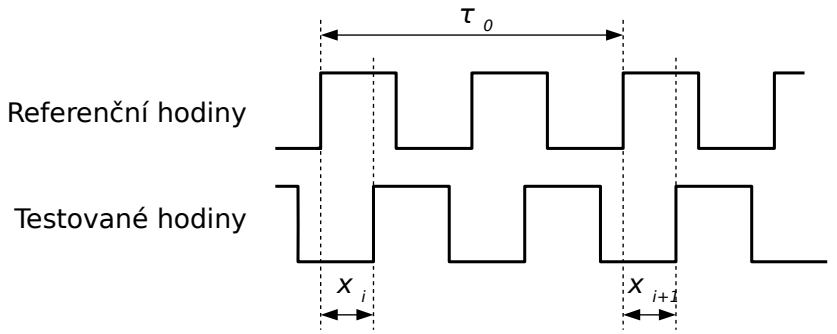
Potřeby časové synchronizace je možné popsat několika parametry. Vytvoření kombinace kritérií se stává výzvou pro další vývoj, protože není možné vytvořit protokol, který by byl dokonalý ve všech směrech. Je tedy nutné parametry kombinovat tak, aby bylo vytvořeno efektivní řešení [13].

- *Přesnost* – potřeba přesnosti časové synchronizace se může velmi lišit s ohledem na účel výsledné sítě. Někdy může postačit pouhé řazení událostí a jindy je potřeba vysoká přesnost v řádu ns.
- *Spotřeba energie* – v některých případech může být žádoucí minimalizovat spotřebu jednotlivých zařízení. Nízká spotřeba jde ale většinou proti přesnosti synchronizace, protože pro vyšší přesnost je třeba například vyměňovat více zpráv nebo použít termostatované oscilátory. Tyto způsoby jsou ale energeticky náročné. Spotřeba energie hraje velkou roli v bezdrátových senzorových sítích, kde jsou například jednotlivé senzory bateriově napájené.
- *Škálovatelnost* – škálovatelnost je důležitá vlastnost pro sítě, u kterých není předem znám počet zařízení. Synchronizační protokol, který byl navržen jako dobře škálovatelný, funguje pro malý i velmi vysoký počet zařízení.
- *Robustnost* – robustnost je významná hlavně pro průmyslové využití. Tímto pojmem rozumíme například to, že pokud v síti vypadnou některá komunikační spojení, případně některá zařízení přestanou fungovat, celý systém bude fungovat dále a zůstane zasynchronizovaný.
- *Životnost synchronizace* – životnost synchronizace může být rozdílná. Požadavkem může být, aby synchronizace fungovala po celou dobu běhu systému a v jakémkoli okamžiku byl systém synchronizován. Na druhou stranu u některých systémů může postačovat jen chvilková synchronizace v čase nějaké události.
- *Rozsah synchronizace* – synchronizace může poskytovat globální synchronizaci pro celou síť, a nebo jen lokální synchronizaci pro zařízení, která jsou v blízkém okolí. Vzhledem ke škálovatelnosti sítě je někdy obtížné dosáhnout globální synchronizaci pro všechny části sítě.
- *Velikost a cena* – velikost a cena výsledného zařízení je důležitá především v senzorových sítích. Použité senzory mají být co nejmenší a co nejlevnější. Ve většině případů není možné kvůli přesné časové synchronizaci použít na všech zařízeních přesný oscilátor nebo GPS přijímač.
- *Bezprostřednost* – bezprostřednost určuje, zdali z procesu synchronizace známe nebo kompenzujeme odchylku od hlavních

hodin a aplikujeme ji okamžitě. V každém časovém okamžiku tedy máme k dispozici odhad přesného času zdroje. Opakem může být pouze sledování a porovnávání časové stupnice se společným prvkem a následné zpětné porovnání naměření dat. Nemusíme mít tedy k dispozici aktuální informaci o čase, ale zpětně jsme schopni se k ní dopočítat.

4.6 Metriky pro hodnocení synchronizace

Pro hodnocení výsledné synchronizace lze použít několik metrik. Nejstriktnější je Maximum Time Interval Error (MTIE), která poukazuje na maximální chybu synchronizace v daném intervalu τ. Dále lze použít například TIErms pro určení efektivní hodnoty chyby synchronizace. Pro sledování vlastností jak hlavních hodin, tak podružných hodin, lze použít Allanovu varianci.



Obr. 4.1: Fázová data

Pro potřeby výpočtů je potřeba naměřit fázová data. Způsob získání vzorků je znázorněn na obr. 4.1.

Sekvence $\{x_i\}$ vzorků TE získávaných po dobu T je definována jako

$$x_i = x\left(t_0 + (i-1)\tau_0\right), \qquad i = 1, 2, 3, \ldots, \tag{4.10}$$

kde t_0 je počáteční pozorovací čas a τ_0 je vzorkovací frekvence. Dále definujeme celkový počet vzorků N_T získávaných po dobu T, jako

$$N_T = T/\tau_0 + 1 \tag{4.11}$$

a počet vzorků n_τ v okně τ jako

$$n_\tau = \tau/\tau_0 + 1\ . \tag{4.12}$$

4.6.1 Time Interval Error

Chyba synchronizace $TE(t)$ podle (4.8) definuje aktuální chybu synchronizace. Time Interval Error, TIE, definuje chybu synchronizace v daném intervalu τ a začínající v čase t_0 lze popsat jako

$$TIE_{t0}(\tau)=TE(t_0+\tau)-TE(t_0)=x_{n_\tau}-x_0\,. \tag{4.13}$$

TIE tedy vyhodnocuje pouze rozdíly jednotlivých chyb synchronizace mezi sebou a ne absolutní hodnotu chyby synchronizace $TE(t)$. Pokud je tedy celá synchronizace stabilní, ale nepřesná, TIE vyjde malé v závislosti pouze na vzájemných rozdílech $TE(t)$. Tímto problémem trpí všechny metriky odvozené od výpočtu TIE a je třeba na to pamatovat a případně udávat i absolutní offset.

Efektivní hodnotu TIE pro dané τ pak můžeme spočítat jako

$$TIE_{rms}(\tau,T)=\sqrt{\frac{1}{N_T-n_\tau}\sum_{i=1}^{N_T-n_\tau}(x_{i+n_\tau}-x_i)^2}\,, \tag{4.14}$$

kde N_T a n_τ jsou definovány podle (4.11) a (4.12).

4.6.2 Maximum Time Interval Error

Maximum Time Interval Error, MTIE, je maximální chyba hodin během určitého intervalu. Tato statistika se používá převážně v telekomunikačním odvětví. Je vypočtena jako pohybující se n-bodové okno přes fázová data, kde nachází rozdíl nejmenší a největší hodnoty pro každou polohu okna. MTIE je pak maximální velikost tohoto rozdílu ze všech vypočtených oken.

$MTIE(\tau,T)$ je pak špičková variance $TE(t)$ ve všech možných pozorovacích intervalech τ při měřené periodě T a je definována jako [24]

$$MTIE(\tau,T)=\max_{0\le t_0\le T-\tau}\left\{\max_{t_0\le t\le t_0+\tau}[TE(t)]-\min_{t_0\le t\le t_0+\tau}[TE(t)]\right\}. \tag{4.15}$$

Aktuální standardy definují MTIE jako funkci τ a tedy implicitně předpokládají

$$MTIE(\tau)=\lim_{T\to\infty}MTIE(\tau,T)\,. \tag{4.16}$$

Z naměřených fázových dat lze vypočítat aproximaci MTIE podle vztahu

$$MTIE(\tau,T)=\max_{j=1}^{N_T-n_\tau+1}\left[\max_{i=1}^{n_\tau+j-1}(x_i)-\min_{i=j}^{n_\tau+j-1}(x_i)\right], \tag{4.17}$$

kde N_T a n_τ jsou definovány podle (4.11) a (4.12).

MTIE je metrika citlivá na špičkové hodnoty časových změn a reaguje tedy citlivě na jednu extrémní hodnotu.

Výpočet MTIE je časově náročný a výpočet podle definice má složitost $O(N^2)$. Existuje ale rychlejší výpočet založený na binární dekompozici popsaný v [24], který má složitost pouze $O(N \log_2 N)$.

4.6.3 Packet Delay Variation

Změna zpoždění paketů, anglicky Packet Delay Variation (PDV), je metrika popisující schopnost dané cesty přenést pakety s konzistentním zpožděním [25]. Tato metrika je důležitá například pro streamované přenosy obrazu a zvuku nebo například pro časovou synchronizaci. Při časové synchronizaci nezáleží na absolutní hodnotě zpoždění přenosové cesty, ale na jejích změnách. Jsou definovány dvě základní metriky.

První metrika z ITU-T Y.1540 [26], označovaná jako PDV, je definována následovně:

$$\begin{aligned} x_k &= a_{2,k} - a_{1,k} \\ v_k &= x_k - d_{1,2} \end{aligned}, \tag{4.18}$$

kde k je index paketu, $a_{1,k}$ je čas vyslání paketu, $a_{2,k}$ je čas příjmu paketu, x_k je absolutní čas strávený na cestě, $d_{1,2}$ je referenční zpoždění paketů, v_k je výsledná dvoubodová změna zpoždění paketu. Distribuční funkce v má pak stejný tvar jako distribuční funkce x, pouze je o $d_{1,2}$ posunutá.

Jako referenční hodnota $d_{1,2}$ může být použita minimální hodnota zpoždění, tato hodnota je doporučována v RFC5481 [27] a ITU-T Y.1451 [28]. Použití minimální hodnoty má výhodu v tom, že všechny hodnoty změny zpoždění paketů jsou kladné a maximální hodnota změny určuje rozsah. Ve starších verzích ITU-T Y.1540 bylo možné používat jako referenci průměrnou hodnotu, ale v aktuální verzi se to již nedoporučuje.

Druhá metrika je popsána v RFC 3393, je označovaná jako IPDV a je definována jako

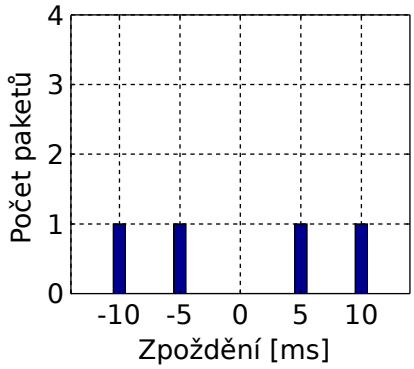
$$\begin{aligned} x_k &= a_{2,k} - a_{1,k} \\ v_k &= x_k - x_{k-1} \end{aligned}, \tag{4.19}$$

kde k je index paketu, $a_{1,k}$ je čas vyslání paketu, $a_{2,k}$ je čas příjmu paketu, x_k je absolutní čas strávený na cestě, x_{k-1} je absolutní čas strávený na cestě předchozího paketu, v_k je výsledná dvoubodová změna zpoždění paketu. Tvar distribuční funkce IPDV není shodný jako PDV a je závislý na pořadí naměřených zpoždění.

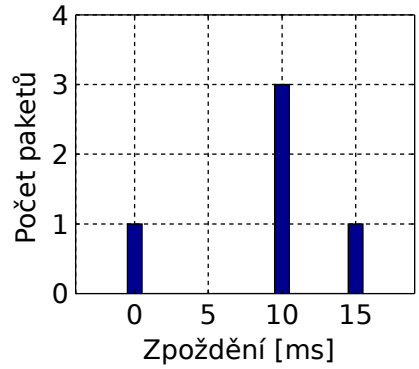
Například pro sekvenci pěti paketů byly změřeny hodnoty v tab. 4.1 a výsledné histogramy jsou na obr. 4.2 a 4.3. [27]

Tab 4.1: Příklad zpoždění paketů a hodnot PDV a IPDV

Paket	**1**	**2**	**3**	**4**	**5**
Zpoždění [ms]	20	10	20	25	20
PDV	10	0	10	15	10
IPDV	-	10	10	5	5



Obr. 4.2: Histogram IPDV



Obr. 4.3: Histogram PDV

4.6.4 Allanova odchylka

Allanova odchylka je metrika pro určení stability oscilátoru a hodin. Na rozdíl od standardní statistické směrodatné odchylky konverguje a je díky ní možné poznat různé druhy šumu. Allanova odchylka [29] $\sigma_y(\tau)$ je definována jako

$$\sigma_y(\tau)=\left\langle \frac{1}{2}\left(y(t+\tau)-y(t)\right)^2 \right\rangle^{\frac{1}{2}}, \tag{4.20}$$

kde $\langle\rangle$ reprezentuje průměr přes nekonečný čas, $y(t)$ je frekvenční odchylka v čase t a τ je časový interval průměrování.

Pro konečnou množinu hodnot lze vytvořit odhad

$$\sigma_y(\tau)=\left[\frac{1}{2(M-1)}\sum_{i=1}^{M-1}\left(y_{i+1}+y_i\right)^2\right]^{\frac{1}{2}}, \tag{4.21}$$

kde M je počet časových intervalů τ, y_i je průměrná diskrétní frekvenční odchylka daná vztahem

$$y_i=\frac{x_{i+1}-x_i}{\tau}, \tag{4.22}$$

kde x_i jsou fázová data dle (4.10). Z (4.21) a (4.22) lze odvodit vztah přímo pro fázová data jako

$$\sigma_y(\tau)=\left[\frac{1}{2(N-2)\tau^2}\sum_{i=1}^{N-2}(x_{i+2}-2x_{x+1}+x_i)^2\right]^{\frac{1}{2}}, \tag{4.23}$$

kde N je počet fázových vzorků.

Popisovaný odhad je jen základní, existuje několik dalších odhadů, které lépe vystihují různé druhy šumu. Odhad lze také spočítat s překrýváním a tím získat lepší interval spolehlivosti vypočtených dat.

5 Měření zpoždění paketů

Synchronizace zařízení může být implementována v ethernetové síti s použitím časových protokolů, např. IEEE 1588. Aktivní síťové prvky jako routery a síťové přepínače mohou ovlivňovat přesnost synchronizace, protože mění zpoždění paketů v síti. Z tohoto důvodu není zpoždění konstantní a také zpoždění v jednom směru není shodné se zpožděním v opačném směru.

V této kapitole je popsán návrh a realizace speciálního zařízení pro přesné měření síťových parametrů, které se skládá ze dvou koncových prvků. Tyto prvky jsou navrženy tak, aby mohly odesílat a přijímat pakety, poskytovat jejich časové značky a vyhodnocovat zpoždění paketů. Zpoždění paketů je následně vyhodnoceno a použito k výpočtu změny zpoždění paketů (PDV), asymetrie přenosové cesty a závislosti zpoždění v jednom směru na zpoždění v opačném směru. Měřicí proces může být snadno rozšířen o výpočet Maximum Time Interval Error (MTIE) [20], Time Deviation (TDEV) a dalších metrik používaných v hodnocení časové synchronizace. Některé metody jsou popsány v [21].

Naměřené hodnoty mohou být použity k určení použitelnosti dané sítě pro implementaci protokolu IEEE 1588 a mohou sloužit k předpovědi kvality synchronizace v této síti. Tato měření jsou důležitá při zavádění přesné synchronizace v nízkonákladových síťových infrastrukturách.

5.1 Popis problému

Kvalita synchronizace v ethernetové síti závisí na přesnosti získávání časových značek na koncových uzlech, stabilitě časových základen koncových uzlů a změně zpoždění paketů způsobené vlastní sítí. Přesné značkování paketů může být prováděno ve speciální ethernetové fyzické vrstvě (PHY). Stabilní časová základna může být zajištěna jak tepelně kompenzovaným oscilátorem (TCXO), tak termostatovaným oscilátorem (OCXO). Změny zpoždění paketů mohou být kompenzovány speciálními síťovými prvky.

Protokol IEEE 1588 je založen na hierarchické Master-Slave struktuře. Hierarchie je vytvářena automaticky pomocí BMCA. Protokol je založen na multicastové komunikaci, ale pro potřeby měření jej lze rozšířit i o unicastovou výměnu paketů.

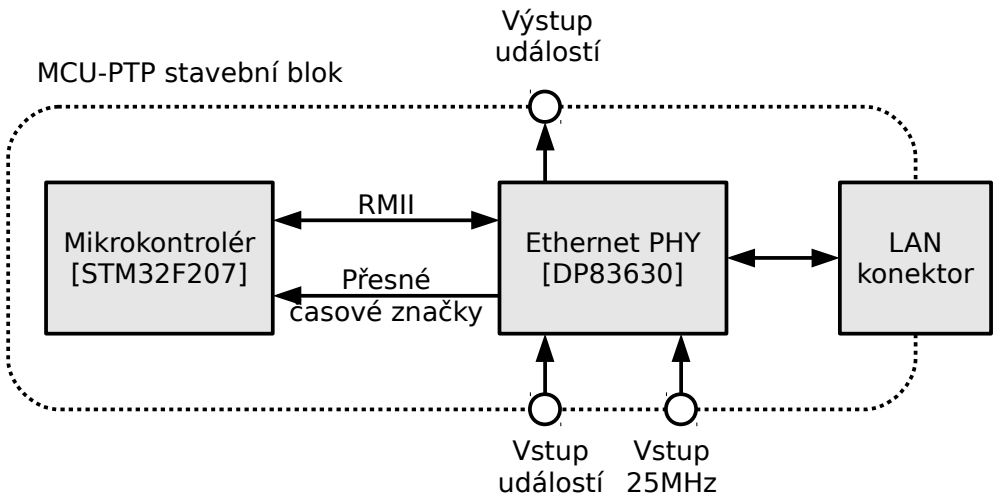
Nejdůležitějším parametrem při synchronizaci je zpoždění paketu od jednoho koncového uzlu k druhému. Především jsou důležité změny těchto zpoždění pro jednotlivé pakety. Paket může projít přes několik aktivních prvků. Další důležitý aspekt k pozorování je měření symetrie přenosové cesty. Pomocí těchto dvou parametrů můžeme zjistit, zdali je síť vhodná k přesné synchronizaci pomocí protokolu IEEE 1588.

5.2 Návrh měřicího zařízení

Hlavní komponentou měřicího zařízení je integrovaný obvod DP83630 od Texas Instruments (viz [30]). Tento obvod umožňuje přidělování časových značek k příchozím a odchozím paketům s rozlišením 8 ns.

Zařízení se skládá ze dvou procesorových bloků, které musí mít vzájemně synchronizovanou časovou stupnici. Není ale nutné použít absolutní stupnici jako UTC. Pro distribuovaná měření je ale definovaná absolutní časová stupnice velmi užitečná [31].

Naměřená data jsou odesílána do nadřazeného počítače pomocí sítě nebo RS-232 nebo mohou být ukládána na vestavěnou SD kartu. Díky použitému obvodu PHY, který obsahuje všechny důležité funkce, lze vybrat libovolný mikroprocesor. Jedinou podmínkou je přítomnost MII/RMII sběrnice. Při návrhu byl využit STM32F207 od firmy STMicroelectronics [32]. Na procesoru běží FreeRTOS [33] a upravená verze PTPd [34]. První verze zařízení byla vyvinuta s použitím vývojového přípravku SleepyCat vytvořeném v rámci řešení práce jako testovací platforma [35].

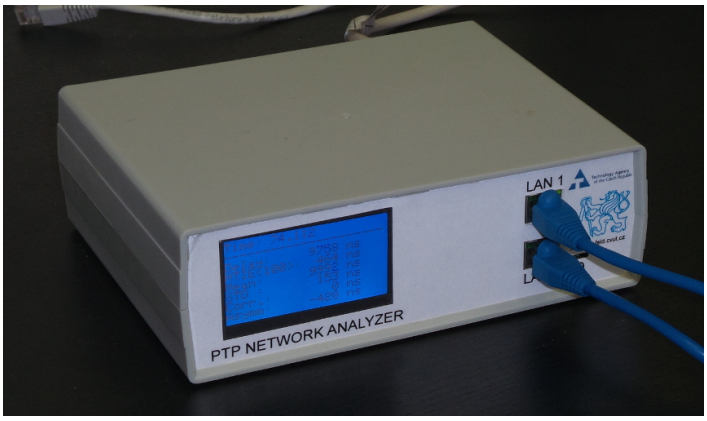


Obr. 5.1: Základní stavební blok zařízení

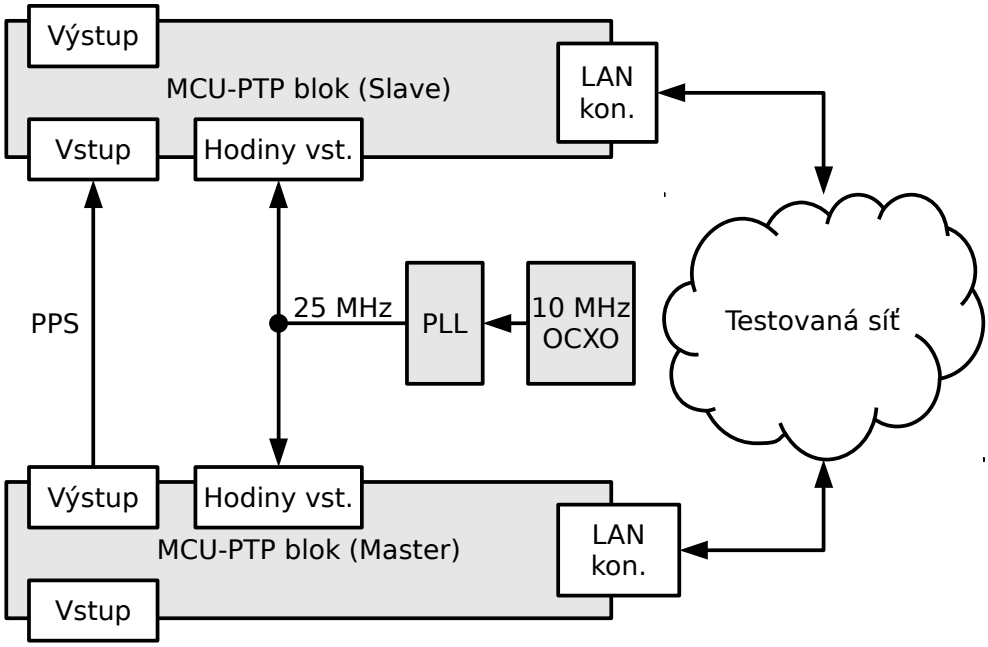
5.2.1 Zařízení pro lokální měření

Zařízení se skládá ze dvou procesorových bloků. Je velice důležité mít tyto bloky vzájemně synchronizované jinou cestou než přes IEEE 1588. Pro lokální měření je jednoduché tento požadavek splnit. Oba procesorové moduly využívají stejnou časovou základnu tvořenou termostatovaným oscilátorem 10 MHz OCXO MTI-210 od firmy MTI-Milliren Technologies, Inc. [36]. Čas je synchronizován s použitím signálu 1 PPS, který je jedním modulem vysílán do druhého. Vzhledem k tomu, že moduly mají stejnou

časovou základnu, stačí pouze jedno měření PPS signálu k synchronizaci časové stupnice. Po automatické kalibraci po startu zařízení mají oba procesorové moduly synchronizovanou časovou stupnici.



Obr. 5.2: Fotografie lokálního měřicího přístroje



Obr. 5.3: Lokální měření

Jeden procesorový modul se z pohledu PTP protokolu chová jako Master a druhý jako Slave. Hlavní rozdíl oproti standardnímu zařízení v režimu Slave ale je, že nesynchronizuje svoje hodiny, ty jsou totiž

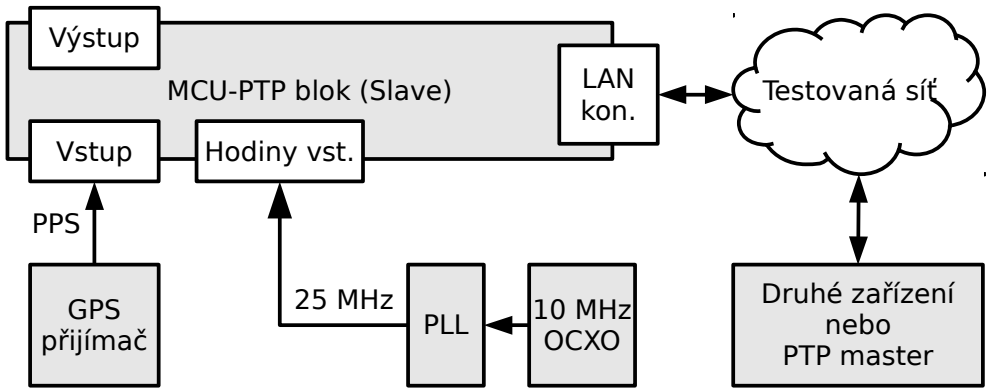
synchronizovány jinou cestou. Zařízení tak může přesně měřit zpoždění jednotlivých paketů.

Zařízení pro lokální měření je první krok k distribuovanému měření. Zařízení je plně funkční pro lokální měření a jeho rozlišovací schopnost je 8 ns. Díky lokální časové stupnici jsou jednotlivé bloky plně synchronizovány a zařízení může měřit i asymetrii přenosové cesty.

5.2.2 Distribuovaný měřicí systém

Koncept měření vychází z lokálního měření, pouze se dva úzce spojené bloky rozdělí na samostatné zařízení a každé zařízení má svou vlastní časovou základnu. Aby celý systém fungoval, je nutné synchronizovat časové základny jinak než s použitím protokolu IEEE 1588. Pro tento účel je každé zařízení vybaveno vlastním termostatovaným oscilátorem a GPS přijímačem [6]. GPS přijímač zajišťuje dlouhodobou a OCXO krátkodobou stabilitu.

Distribuovaný měřicí systém funguje podobně jako lokální. Jeden blok může být dokonce nahrazen standardním zařízením implementujícím hlavní hodiny protokolu IEEE 1588 v režimu Master. Klíčové je zařízení, které se chová jako Slave. To je synchronizováno pomocí GPS na stejnou časovou stupnici jako Master a může tedy nezávisle měřit parametry přenosové cesty v obou směrech.



Obr. 5.4: Vzdálené měření

5.3 Naměřené výsledky

Základní měřená veličina těchto zařízení je zpoždění přenosové cesty. Navržené měřicí systémy byly nejprve otestovány a zkalibrovány na sítích se známým nebo vypočitatelným zpožděním.

Po otestování byly měřeny sítě s neznámými parametry. V měřených sítích byly použity jednak běžné síťové prvky a jednak průmyslové síťové přepínače s podporou protokolu IEEE 1588.

5.3.1 Kalibrace pomocí UTP kabelu

Pro zkalibrování zařízení bylo potřeba změřit systém se známými parametry. Byly použity kabely UTP v různých délkách. Bylo změřeno přenosové zpoždění pomocí čítače SR620 a tyto hodnoty jsou v tab. 5.1.

Tab. 5.1: Naměřená zpoždění na UTP kabelech

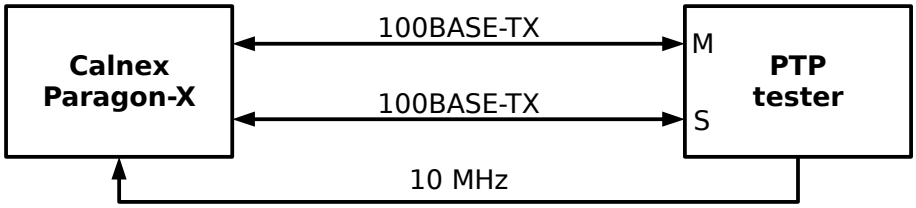
Cat	Cat 5	Cat 5	Cat 5e	Cat 5e	Cat 5e
Typ	UTP	UTP	S/UTP	S/UTP	S/UTP
Délka	1 m	10 m	1 m	5 m	50 m
Referenční zpoždění	5,0 ns	44,3 ns	5,6 ns	22,4 ns	236,1 ns
Naměřené zpoždění	6 ns	46 ns	6 ns	22 ns	238 ns
Odchylka od reference	1 ns	1,7 ns	0,4 ns	0,4 ns	1,9 ns

Podle katalogového listu DP83630 je interní zpoždění značkování paketů 210 ns. To bylo odečteno před vlastním měřením.

Podle tab. 5.1 vycházela odchylka naměřených hodnot od referenčních menší než 2 ns, přičemž granularita určování časových značek zařízení byla 8 ns.

5.3.2 Kalibrace pomocí generátoru zpoždění paketů

První kalibrace byla vytvořena pomocí měření UTP kabelů. Zpoždění lze měřit pomocí čítače. Maximální délka zpoždění je dána maximální délkou kabelu, což pro Ethernet znamená 100 m.



Obr. 5.5: Zapojení simulátoru zpoždění paketů

Pro kalibrování delších zpoždění bylo třeba použít simulátor zpoždění paketů. V tomto případě byl použit Calnex Paragon-X vyrobený firmou Calnex Solutions. Paragon-X může simulovat zpoždění paketů v ethernetové síti a umožňuje nastavit i dynamické změny. Měření bylo provedeno ve Skotsku v sídle společnosti Calnex Solutions, kde nám byl simulátor k dispozici.

Celý test byl proveden v síti založené na 100BASE-TX Ethernetu. Obě zařízení byla napojena na stejné referenční hodiny 10 MHz, čímž byla zaručena stejná časová stupnice simulátoru i testeru. Celkové zapojení používané při testech je na obr. 5.5.

Před testováním bylo změřeno zpoždění připojených kabelů. Zpoždění každého z nich bylo $T_c = 14\,ns$. Pro Paragon-X byla ještě nastavena kompenzační konstanta fixního zpoždění $T_p = 3{,}5\,\mu s$.

Zpoždění přenosové cesty T_a bylo měřeno pro několik předdefinovaných zpoždění T_s. Hodnoty v tab. 5.2 jsou vypočteny podle vztahu

$$T_d = T_a - T_s - T_p - 2T_c, \tag{5.1}$$

kde T_d představuje rozdíl od očekávané hodnoty, T_a je naměřené zpoždění, T_s je předdefinované zpoždění, T_p je fixní zpoždění Paragon-X a T_c je zpoždění na propojovacím kabelu.

Asymetrie T_{as} přenosové cesty je spočítána jako rozdíl naměřeného zpoždění v jednom a v druhém směru podle vztahu

$$T_{as} = T_{aM \to S} - T_{aS \to M}. \tag{5.2}$$

Podle tab. 5.2 vycházela asymetrie zhruba 56 ns, což byla očekávaná hodnota zařízení Paragon-X při použití 100BASE-TX Ethernet. Zpoždění bylo měřeno v obou směrech. Hodnota byla měřena zhruba čtyřikrát za sekundu. V tabulce je uveden průměr z 1000 měření.

Další experiment spočíval v nastavení proměnlivého zpoždění na simulátoru a měření změn tohoto zpoždění. V grafu je použito zobrazení hodnoty $T_r(t)$, což je kompenzovaná naměřená hodnota zpoždění o známé offsety systému, a referenční hodnoty $T_s(t)$.

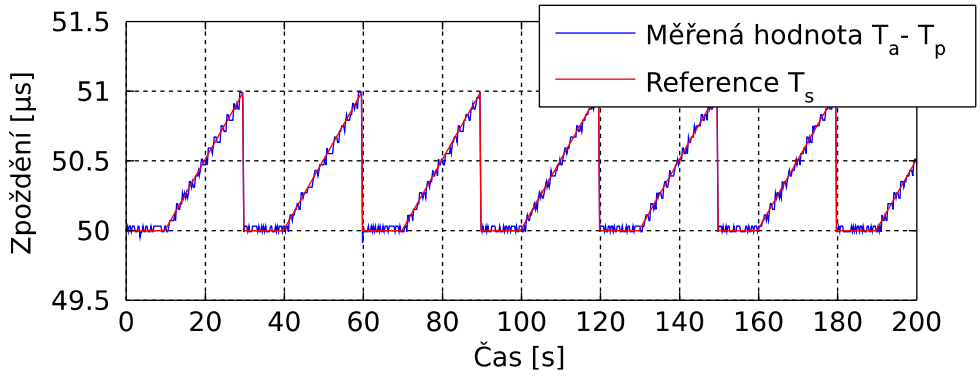
$$T_r(t) = T_a(t) - T_p - 2T_c \tag{5.3}$$

Na obr. 5.6 je znázorněn pilový průběh zpoždění s parametry základního zpoždění 50 µs a amplitudy pily 1 µs. Referenční i naměřený průběh spolu korespondují, na obr. 5.7 je tedy znázorněn rozdíl od reference $T_d(t)$. Průběh chyby je informativně rozšířen o průběh referenční hodnoty zpoždění, který ale není v měřítku.

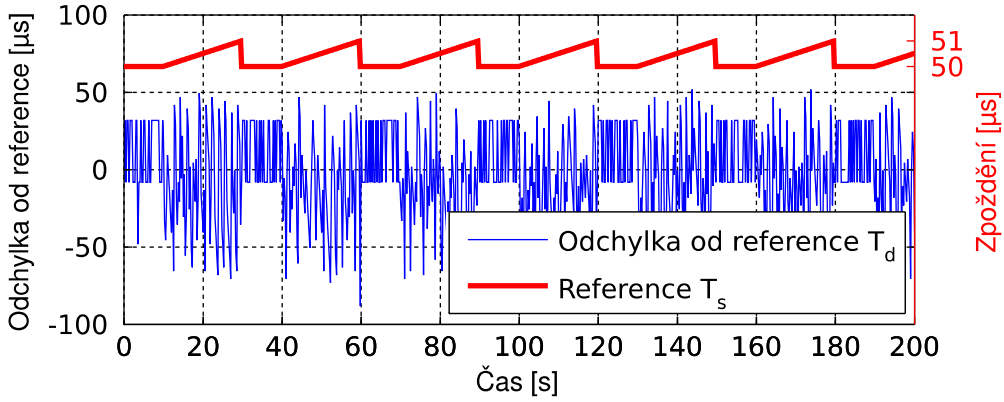
Poslední kalibrační měření bylo provedeno s pilovým průběhem zpoždění 10 µs a je vyobrazen již jen rozdíl od reference na obr. 5.8.

Tab. 5.2: Naměřené odchylky zpoždění paketů přes Paragon-X

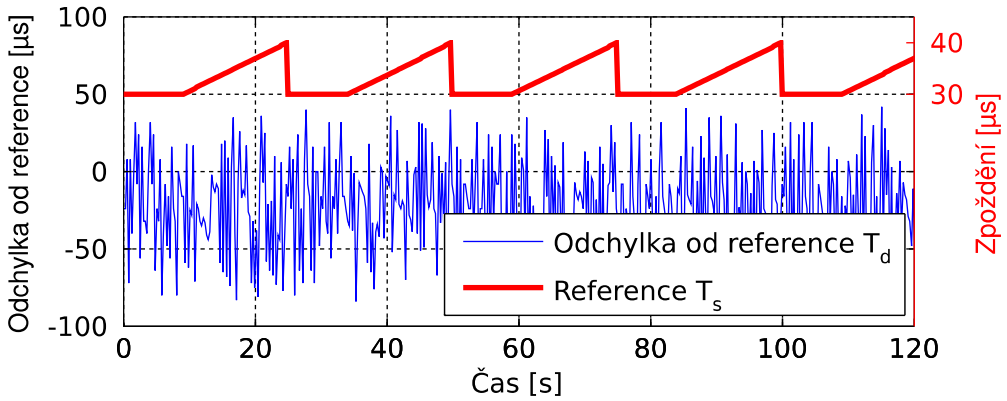
Konfigurované zpoždění T_s	**Průměr** T_d	**Směrodatná odchylka** T_d	**Asymetrie** T_{as}
0 ms	1975 ns	20 ns	56 ns
0,5 ms	36 ns	20 ns	57 ns
1,5 ms	38 ns	20 ns	57 ns
10 ms	40 ns	20 ns	56 ns



Obr. 5.6: Naměřené a referenční zpoždění - pila 1 µs



Obr. 5.7: Rozdíl naměřené hodnoty zpoždění paketu od reference – pila 1 µs



Obr. 5.8: Rozdíl naměřené hodnoty paketu od reference - pila 10 µs

5.3.3 Testování obyčejného síťového přepínače

Pro testování měření zpoždění paketů byly vybrány dva síťové přepínače. První je obyčejný bez speciální podpory protokolu IEEE 1588, druhý je průmyslový s podporou pro transparentní hodiny protokolu IEEE 1588.

Obyčejný síťový přepínač byl typ TP-LINK TL-SF1008P obsahující čip RTL8309G, který se osazuje i do zařízení jiných výrobců v podobné cenové hladině.

Tab. 5.3: Zpoždění paketů obyčejného síťového přepínače

Směr	Průměr	Směrodatná odchylka	Rozsah PDV
1 → 2	9851 ns	126 ns	504 ns
2 → 1	9764 ns	126 ns	504 ns

Výsledky byly naměřeny z přibližně 78 000 výměn paketů s rychlostí odesílání 1 paket/s. Z výsledků v tab. 5.3 je vidět, že zpoždění v jednom směru je průměrně o 100 ns kratší než v tom druhém.

5.3.4 Testování průmyslového síťového přepínače

Jako průmyslový síťový přepínač byl vybrán Hirschmann MS20. Testování probíhalo jak se započítáváním korekce z transparentních hodin, tak bez ní. Měření proběhlo rychlostí 3,884 paketů/s a bylo provedeno 332 000 výměn paketů.

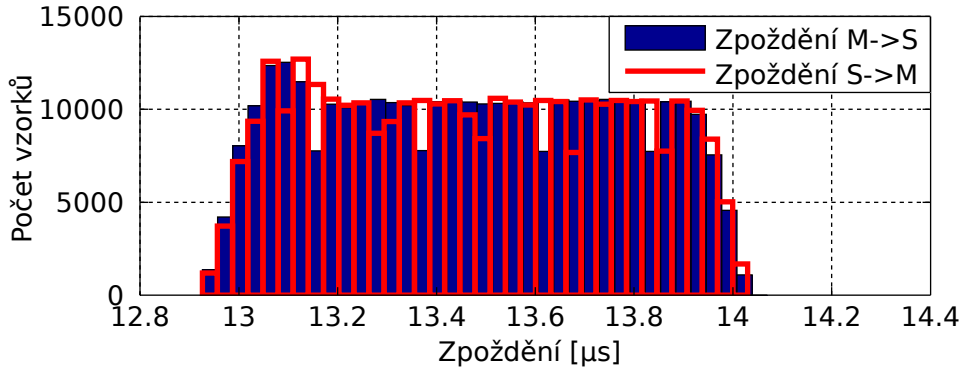
Takto vysoký počet výměn byl volen proto, aby bylo možné zjistit, že se parametry síťového přepínače v čase nemění. Měření ukázala, že se parametry mění rovnoměrně v omezeném rozsahu. Bez započtení korekcí kolísá zpoždění paketů v rozmezí jedné µs přibližně v rovnoměrném rozložení. Při započtení korekcí se rozsah zmenší na pouhých 25 ns a rozložení se blíží normálnímu, takže jsou častější hodnoty okolo průměrných hodnot, než okrajové.

Tab. 5.4: Zpoždění paketů průmyslového síťového přepínače bez korekcí

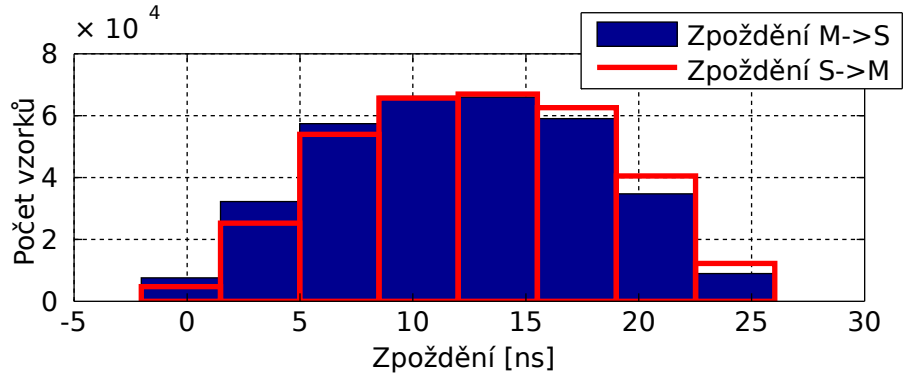
Směr	Průměr	Směrodatná odchylka	Rozsah PDV
1 → 2	13475 ns	294 ns	1144 ns
2 → 1	13475 ns	293 ns	1104 ns

Tab. 5.5: Zpoždění paketů průmyslového síťového přepínače s korekcemi

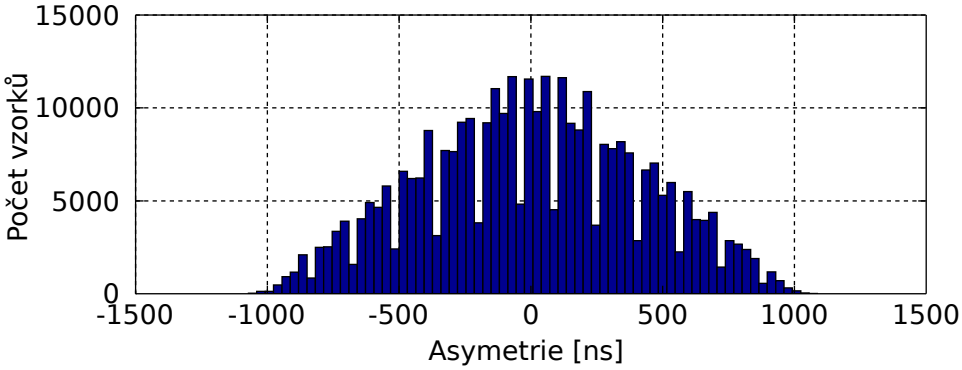
Směr	Průměr	Směrodatná odchylka	Rozsah PDV
1 → 2	12 ns	7 ns	28 ns
2 → 1	12 ns	7 ns	28 ns



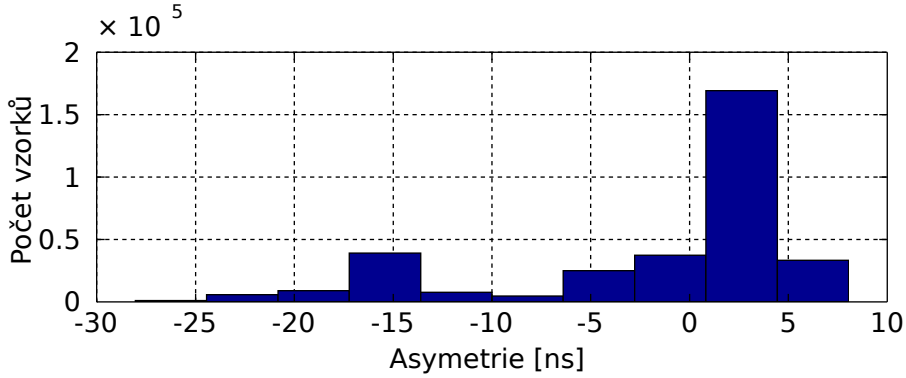
Obr. 5.9: Histogram zpoždění paketů bez korekcí



Obr. 5.10: Histogram zpoždění paketů s korekcemi



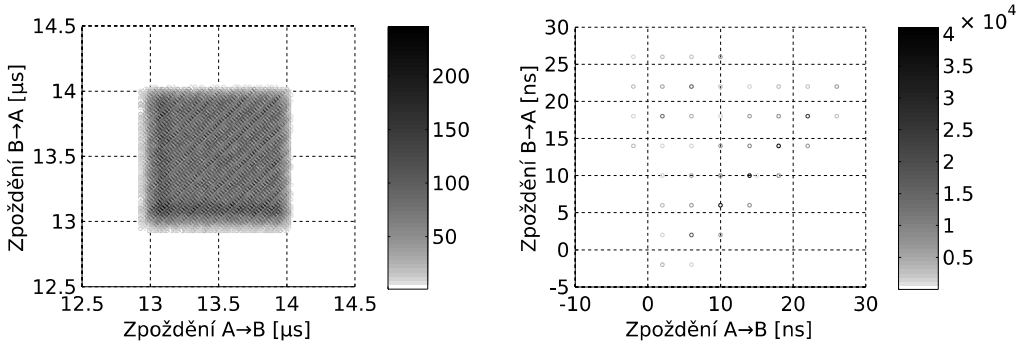
Obr. 5.11: Histogram asymetrie bez korekcí



Obr. 5.12: Histogram asymetrie s korekcemi

Síťový přepínač umí poskytnout zpoždění paketu jako korekční hodnotu přímo v paketu. Bez korekcí vychází ve všech parametrech hůře než obyčejný síťový přepínač, ale při započítání těchto korekcí již vše vychází o několik řádů přesněji.

Výsledky měření zpoždění průmyslového síťového přepínače byly ještě dále zpracovány. Vzhledem k tomu, že zařízení vysílají měřicí paket ve stejný čas, je možné vyhodnotit aktuální asymetrii zpoždění přenosové cesty pro každou výměnu paketů. Vynesením těchto odpovídajících měření z obou směrů do grafu závislosti zpoždění v jednom směru na zpoždění v druhém směru lze vypozorovat vlastnosti asymetrie. Pokud by vynesené body tvořily přímku, nebo by byly v okolí přímky, znamenalo by to, že asymetrie zpoždění je konstantní. V našem měření se ale ukázalo, že zpoždění v jednom směru není nijak závislé na zpoždění v druhém směru. Zpoždění jsou v obou směrech náhodné a přibližně rovnoměrně rozdělené v celém intervalu od 13 µs do 14 µs, viz obr. 5.13.



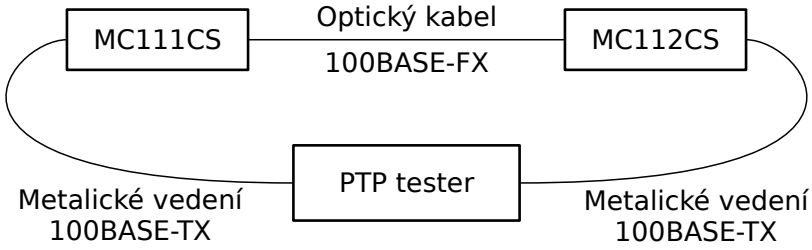
a) *b)*

Obr. 5.13: Závislost zpoždění paketů v jednom směru na druhém směru – a) bez korekce, b) s korekcí

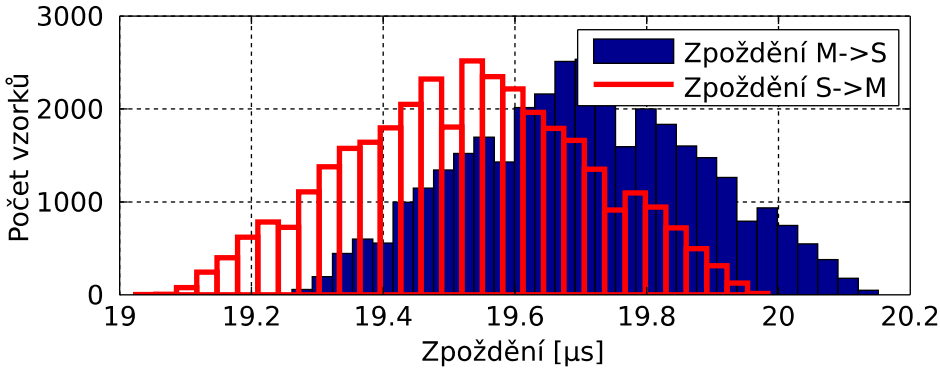
5.3.5 Testování převodníku z metalického na optické vedení

Ethernet po metalickém vedení má nevýhodu v omezení maximální délky segmentu, která je zhruba 100 m. Na větší vzdálenost je nutné použít další aktivní prvky, které ale mohou zhoršovat parametry synchronizace.

Řešením pro tuto situaci může být použití optického vlákna, jehož jeden segment může být několikanásobně delší. Pro testování byl vybrán převodník metalického vedení na optické TP-LINK MC111CS v páru s MC112CS. Blokové schéma testovaného zapojení je na obr. 5.14. Vybrané převodníky deklarují použití až na vzdálenost 20 km.



Obr. 5.14: Schéma zapojení media konvertorů



Obr. 5.15: Zpoždění paketů na media konvertoru

Podle naměřených údajů se převodník v tomto podání chová jako obyčejný síťový přepínač. Dva převodníky se chovají jako dva síťové přepínače za sebou. Přináší tedy dvojnásobné PDV a dvojnásobnou asymetrii. Na obr. 5.15 je zobrazen histogram zpoždění paketů v obou směrech. Z histogramu je patrná asymetrie, která v tomto případě činí 175 ns. Ostatní naměřené výsledky jsou v tab. 5.6. Rozsah změn zpoždění paketů vychází zhruba 1 µs. Při použití podružných hodin bez teplotně kompenzovaného oscilátoru by nebylo jednoduché takto velké změny kompenzovat.

Při požadavku na delší délky segmentu lze principiálně použít optické vedení, ale použití testovaných převodníků je pro účely časové synchronizace nevhodné. Převodníky se chovají jako síťový přepínač a každý z nich zanáší velkou a proměnnou asymetrii do přenosové cesty.

Tab. 5.6: Naměřená data na převodnících

Směr	Průměr	Směrodatná odchylka	Rozsah PDV
1 → 2	19,70 µs	178 ns	952 ns
2 → 1	19,53 µs	178 ns	960 ns

Velikost PDV při použití převodníků vycházela okolo 960 ns, takže dosažení přesnosti synchronizace pod 1 µs by nebylo snadné. Levné koncové prvky bez stabilních oscilátorů by nebylo možné provozovat při požadavku na takovou přesnost synchronizace.

5.3.6 Kalibrace distribuovaného testeru

Kalibrace distribuovaného testeru byla provedena v laboratorních podmínkách a byla měřena síť známých parametrů, podobně jako v kapitole 5.3.1. Pro testování byl použit UTP kabel délky 30 m. Každé měřicí zařízení mělo vlastní GPS přijímač pro synchronizaci. Bylo provedeno 231 tis.

výměn paketů s rychlostí 1 výměna/s. Zpoždění paketů dané délkou kabelu je 140 ns.

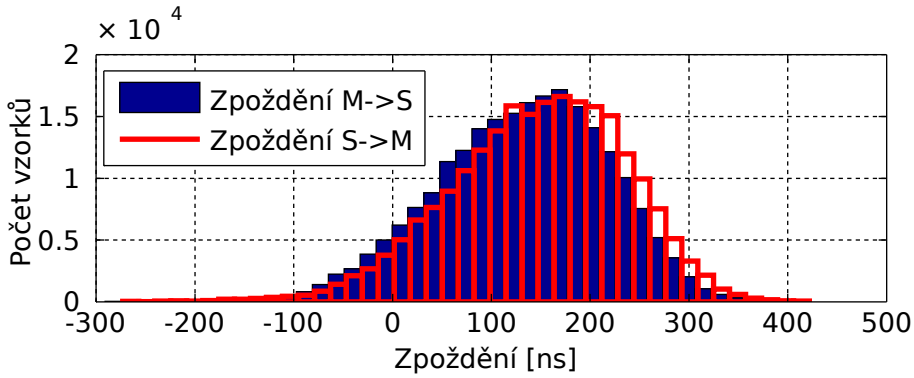
V tab. 5.7 jsou uvedeny průměrné hodnoty naměřeného zpoždění a směrodatná odchylka. Je také uvedena hodnota PDV, která reprezentuje vlastnosti použitého GPS přijímače, protože PDV u lokálního měření pomocí UTP kabelů byla nulová.

Na obr. 5.16 je znázorněn histogram naměřených zpoždění. Z něj je patrné, že některá zpoždění vycházela záporně, což na první pohled vypadá nesmyslně. Záporné hodnoty jsou dány nepřesností referenčních hodin a stabilitou časové základny poskytované GPS přijímači.

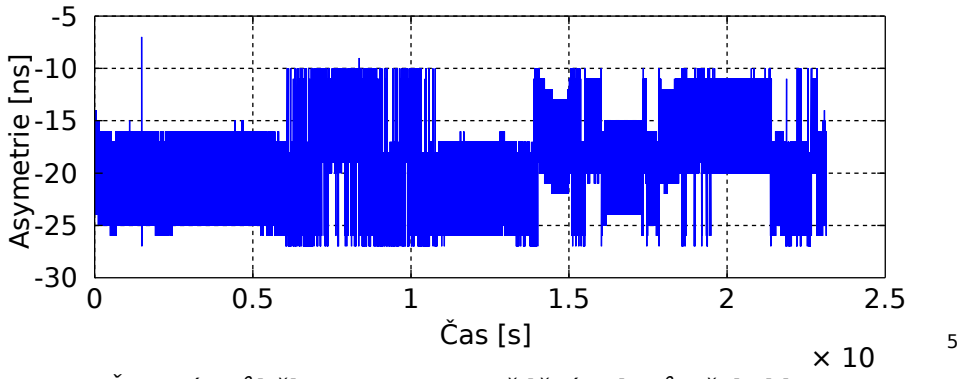
Na obr. 5.17 je znázorněn časový průběh asymetrie. Tyto hodnoty jsou ovlivněny stabilitou jednotlivých GPS přijímačů, které slouží jako reference a dále granularitou hodin a od nich odvozených časových značek. Hodnoty kolísají s krokem zhruba 8 ns, což je rozlišovací schopnost testeru.

Tab. 5.7: Zpoždění paketů při kalibraci distribuovaného testu

Směr	**Průměr**	**Směrodatná odchylka**	**Rozsah PDV**
1 → 2	129 ns	88 ns	694 ns
2 → 1	147 ns	88 ns	695 ns



Obr. 5.16: Histogram zpoždění paketů při kalibraci distribuovaného testeru



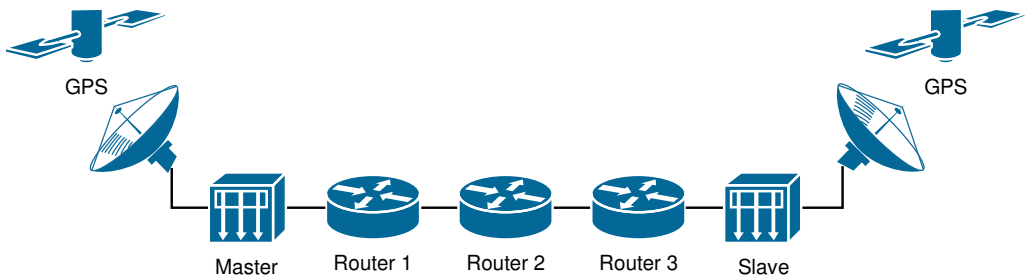
Obr. 5.17: Časový průběh asymetrie zpoždění paketů při kalibraci distribuovaného testeru

5.3.7 Měření zpoždění paketů v akademické síti

Distribuovaný PTP tester byl testován i v akademické síti. Byla použita dvě zařízení. Podle dostupné topologie byly v cestě 3 routery. Měření bylo prováděno rychlostí 1 paket/s a bylo zachyceno více než 60 tis. paketů.

Na obr. 5.18 je znázorněna zjednodušená topologie měřené sítě. Na obou koncích sítě je jeden modul distribuovaného testeru. Jeden modul je v režimu master a druhý v režimu slave. Oba moduly jsou nezávisle synchronizovány s pomocí GPS přijímače. Moduly si vzájemně vyměňují zprávy PTP protokolu, pouze se podle nich nesynchronizují, jen měří zpoždění přenosové cesty.

Z naměřených dat je zřejmá asymetrie přenosové cesty, která činí 4,8 µs. Dále byly určeny základní statistiky naměřených dat, které jsou uvedeny v tab. 5.8. Změny zpoždění paketů vycházejí v jednom směru téměř dvojnásobné oproti směru opačnému.

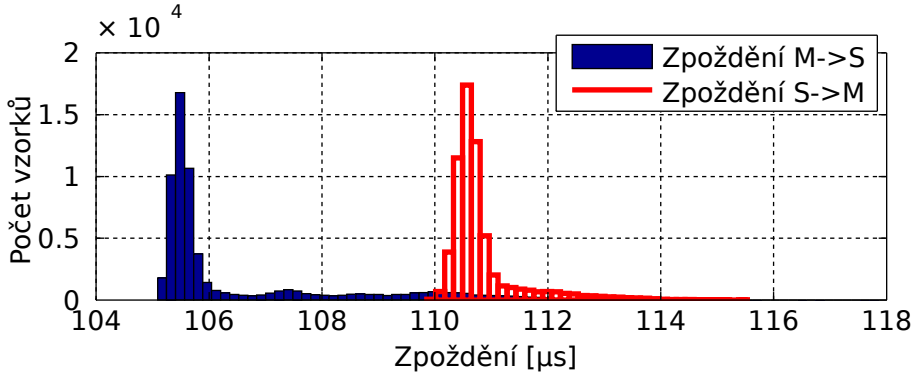


Obr. 5.18: Schéma měření distribuovaného testeru

Tab. 5.8: Zpoždění paketů v akademické síti

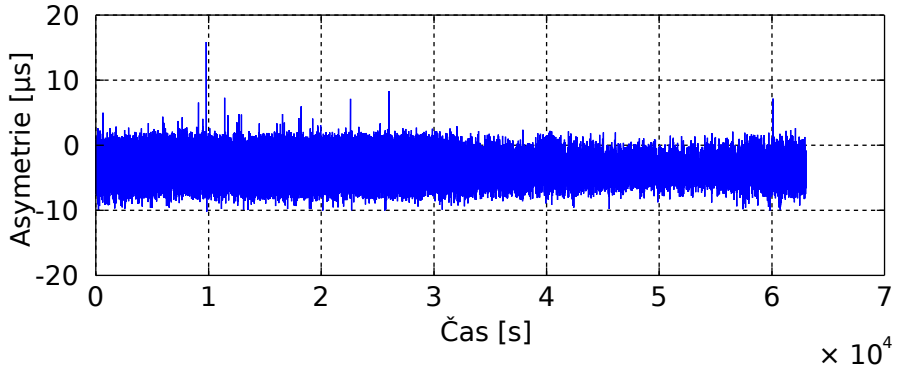
Směr	Průměr	Směrodatná odchylka	Rozsah PDV
1 → 2	106,71 µs	17,26 µs	2445,5 µs
2 → 1	111,01 µs	27,91 µs	5746,5 µs

Na obr. 5.19 je znázorněn histogram zpoždění paketů v obou směrech. Na histogramu je vidět typické rozložení zpoždění paketů na rozsáhlejších sítích. Většina paketů vykazuje podobné zpoždění, které je jen o několik stovek ns větší než minimální délka zpoždění přenosové cesty. Existuje však malé procento paketů, jejichž zpoždění je delší, někdy až v řádu milisekund.



Obr. 5.19: Zpoždění paketů v akademické síti

Na obr. 5.20 je znázorněn průběh naměřené asymetrie. Průměrná asymetrie sítě vycházela 4,8 µs. Pokud se v síti vyskytuje asymetrie, ale není kompenzována, bude působit chybu synchronizace o velikosti poloviny své hodnoty, tedy v tomto případě 2,4 µs. Pokud do koncových zařízení tuto asymetrii zadáme, můžeme ji kompenzovat a dosáhnout lepší přesnosti synchronizace. V tomto měření vycházela asymetrie konstantní, ale bohužel ji nemůžeme považovat za konstantní i v budoucnu. Libovolná změna konfigurace routerů může změnit zpoždění v libovolném směru.



Obr. 5.20: Asymetrie zpoždění paketů v akademické síti

5.4 Zhodnocení výsledků

Bylo navrženo a realizováno zařízení pro přesné měření zpoždění přenosové cesty v ethernetové síti popsané v kapitole 5.2. Zařízení je založeno na zasílání zpráv protokolu IEEE 1588, které se v tomto případě nepoužívají k synchronizaci, ale pouze k měření. Byly navrženy dvě varianty, jedna slouží pro lokální měření a druhá pro vzdálené distribuované měření.

Zařízení bylo zkalibrováno a otestováno na sítích se známým zpožděním (kapitoly 5.3.1 a 5.3.2). Byly použity jednak UTP kabely různých délek a jednak simulátor zpoždění přenosové cesty Calnex Paragon-X. Granularita měření časových značek zařízení je 8 ns. Při měření zpoždění na UTP kabelech byl naměřen rozdíl od referenčního měření lepší než 2 ns. Při měření na simulátoru byl rozdíl od referenčního zpoždění lepší než ±60 ns, což bylo výrobcem potvrzeno jako vlastnost simulátoru.

Pomocí navrženého zařízení byla změřena zpoždění přenosové cesty neznámých sítí (kapitoly 5.3.3, 5.3.4 a 5.3.5). Byly měřeny vlastnosti obyčejného a průmyslového síťového přepínače a dále byly měřeny vlastnosti převodníků z metalického vedení na optické. Podle naměřených výsledků se bez korekcí chovají všechna zařízení podobně a vykazují zpoždění přenosové cesty zhruba 10 µs a PDV 0,5 µs až 1 µs. U průmyslového síťového přepínače je ale výhoda v podpoře protokolu IEEE 1588. Síťový přepínač přidává do přenášené zprávy informaci, jak dlouho tato zpráva v zařízení setrvala. Tím je možné kompenzovat PDV, které pak vycházelo pouze 28 ns.

V kapitole 5.3.6 je popsána kalibrace distribuovaného měření a v kapitole 5.3.7 je popsáno měření zpoždění přenosové cesty v akademické síti. Díky přesnému měření byla odhalena asymetrie zpoždění měřené sítě o velikosti 4,8 µs.

6 Testování a validace PTP protokolu

Testování lze rozdělit do dvou hlavních skupin. První skupina testů je založena na správnosti implementace protokolu v různých situacích a dodržování standardu. Druhá skupina testů vyhodnocuje vlastní přesnost synchronizace při různých podmínkách na přenosové cestě.

Pro druhou skupinu existuje několik komerčních nástrojů pro provedení testu [37] a dále existují například telekomunikační normy, definující požadavky na synchronizaci v různých situacích [38].

Vlastní protokol PTP je podrobně popsán v příslušné normě IEEE 1588:2008 [14], ale již není specifikováno, jak implementaci protokolu testovat. Na testování protokolu PTP neexistují žádné standardní nástroje. Lze použít pouze obecné zachytávání paketů a jejich zkoumání pro základní testy a následně již pouze připojení do sítě s dalšími prvky podporujícími protokol PTP. Lze použít jak reálné, tak simulované prvky [39].

Částečně má tento problém odstranit setkání výrobců zařízení s protokolem PTP každý půlrok na akci s názvem PlugFest při konferenci ISPCS [40]. Zde se testují různé scénáře a všichni účastníci jsou připojeni do sítě a sledují chování svých zařízení. Tato setkání mají několik problémů:

- testování probíhá pouze jednou za půl roku,
- testování je negativně ovlivněno účastníky, kteří mají chybu, v implementaci protokolu,
- testování trvá dlouho a je těžko opakovatelné,
- výsledky testování nelze jednoduše vyhodnotit a po projití všech scénářů nelze jednoznačně říct, že všechno funguje správně.

Některé chyby implementace protokolu nejsou na první pohled patrné a zařízení zdánlivě funguje, porušuje ale některé striktní předpisy o povolené četnosti dotazování nebo dodržení synchronizačního intervalu.

Pro potřeby testování a validace jednotlivých implementací používající PTP protokol bylo zapotřebí vytvořit nástroj, který to provede automatizovaně a který může provést dlouhodobý a opakovatelný test celé konfigurace [41].

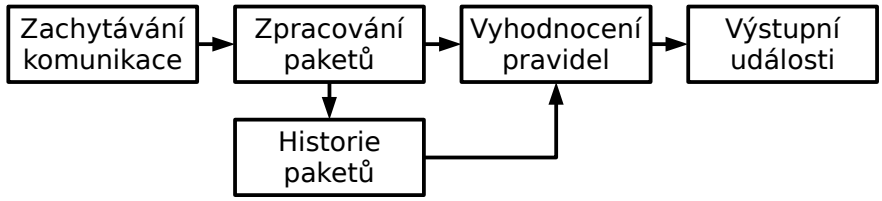
Základní požadavky na takový nástroj jsou:

- Analýza posloupnosti paketů
- Analýza deklarovaných a skutečných rychlostí odesílání paketů
- Analýza správnosti obsahu paketu
- Analýza správnosti BMCA
- Analýza správného zdroje a cíle každého paketu

Nástroj, který splňuje tyto podmínky, byl navržen, vytvořen a ověřen na testovací síti a na záznamu komunikace z Plugfestu [42].

6.1 Struktura aplikace

Aplikace (viz obr. 6.1) je rozdělena do několika hlavních funkčních bloků, které jsou spolu vzájemně propojeny. Celá aplikace je napsána v jazyce Java a má pouze textové rozhraní.



Obr. 6.1: Struktura aplikace

6.1.1 Záznam paketů

Záznam paketů je realizován pomocí knihovny PCAP, která se stará o nízkoúrovňový přístup k síťovému rozhraní. Tyto pakety jsou zpracovány a filtrovány a relevantní pakety protokolu PTP jsou předány vyšší vrstvě aplikace. Tato knihovna je multiplatformní a celá aplikace je tedy přenositelná a spustitelná jak na operačním systému Windows, tak na operačním systému Linux.

Knihovna PCAP existuje pro operační systém Linux pod názvem libpcap a pro operační systém Windows pod názvem winpcap. Tyto nízkoúrovňové knihovny pracují na úrovni ovladače síťové karty a dokáží zaznamenávat nebo modifikovat síťový provoz. Aby bylo možné použít tyto nízkoúrovňové knihovny v Javě, je použita knihovna JNetPcap, která zapouzdřuje volání nativní knihovny do jazyka Java.

Knihovna PCAP umožňuje zaznamenávat všechny pakety, nebo použít rozsáhlé možnosti filtrování. Pro filtrování PTP paketů odesílaných přímo na druhé vrstvě lze použít pravidlo „`ether proto 0x88F7`“, Pro zaznamenávání PTP paketů v IPv4 nebo IPv6 lze použít pravidlo „`udp port 319 or udp port 320`“.

6.1.2 Zpracování paketů

Další vrstvou aplikace jsou pravidla. Pravidla se jednak spouštějí buď při příchodu nějakého paketu, nebo po uplynutí určené doby. Pokud jsou data zpracovávána z uloženého záznamu komunikace, je aktuální čas interpolován z přijatých paketů.

U každého paketu je detekováno, o jaký typ paketu se jedná, paket je otestován a načten do datové struktury, ve které je snadné přistupovat k jednotlivým položkám.

Každý takto zpracovaný paket je následně předán všem pravidlům a pravidla provádějí rozbor a testování. Pokud je nalezen problém, pravidlo předá informaci o chybě výstupnímu subsystému.

Program si dále vede evidenci o všech zařízeních, které se účastní komunikace. Do paměťové struktury se ukládá použitá verze protokolu, komunikační vrstva, počet odeslaných paketů apod.

6.2 Pravidla pro zpracování dat

Každý paket projde aplikací a je vyhodnocen. Nejprve je vyhodnocen a klasifikován obecným způsobem a pak je předán konkrétním pravidlům. Ze zaznamenaných paketů jsou vyhodnocena komunikující zařízení a je i odhadnut jejich vnitřní stav. Podle těchto informací se pak aplikace rozhoduje, jestli došlo k porušení pravidel nebo ne.

6.2.1 Typy zpráv

V protokolu IEEE 1588 je definováno několik typů zpráv. Každá zpráva slouží k jinému účelu. Některé zprávy musí přijímací strana označit časovou značkou a některé nemusí. Existují tyto typy zpráv:

Annouce – slouží k vyslání informací o zařízení Master. Tato zpráva slouží k vyhodnocení nejlepších hodin v síti. K této zprávě se neukládá časová značka.

Sync – slouží k synchronizaci a zařízení Slave. Na příjmu se ukládá časová značka. Pokud používá Master speciální hardware, může tato zpráva nést informaci o odeslání a nebo může být následována zprávou FollowUp.

FollowUp – slouží k předání času odeslání Sync zprávy v případě, že tuto informaci nelze přenést přímo v Sync.

DelayReq, DelayResp – jsou zprávy pro měření zpoždění přenosové cesty pomocí Slave. Slave se pomocí zprávy DelayReq zeptá na zpoždění a pomocí zprávy DelayResp získá odpověď od Master. Tyto zprávy slouží pro měření zpoždění v režimu E2E. Slave si uloží časovou značku odeslání DelayReq a Master mu zpět ve zprávě DelayResp zdělí čas přijetí.

PdelayReq, PdelayResp, PdelayRespFollowUp – jsou zprávy pro měření zpoždění v režimu P2P. Vzhledem k tomu, že tyto zprávy může použít jakékoli zařízení, tedy i master, musí obsahovat plnou sadu časových značek, tedy jak čas odeslání a čas přijetí dotazu, tak i čas odeslání a čas přijetí odpovědi.

Management a Signalling – jsou zprávy pro vyčítání a nastavování parametrů hodin. Na tyto zprávy může zařízení odpovídat v libovolném režimu.

6.2.2 Stavy portů

Každé zařízení má jeden a více portů. Každý port má svou adresu a je v nějakém stavu. U jednoho zařízení s více porty může být každý port v jiném stavu. Jednotlivé stavy mají omezení vzhledem k protokolu, nemohou například odesílat některé typy zpráv. Popis jednotlivých stavů vychází ze standardu IEEE 1588 a je následující:

Master – v každé uzavřené síti pro danou PTP doménu by měl být maximálně jeden port, který se chová jako Master. Při použití Boundary Clock, se síť rozdělí na několik částí. Každá část má pak vlastní zařízení Master. V takové topologii nelze tento analyzátor jednoduše provozovat bez dalších sond. Zařízení Master může vysílat zprávy Announce, Sync a FollowUp. V případě použití režimu P2P také zprávy pro měření tohoto zpoždění.

Slave – v každé síti může být několik zařízení ve stavu Slave. V tomto stavu jsou hodiny ve chvíli, kdy mají jednoznačně vybrané hlavní hodiny a synchronizují se z nich.

Listening – v tomto stavu zařízení setrvá, dokud není vybrán vhodný Master. Pokud do určitého timeoutu nenavrhne žádné zařízení, že se stane master, tak může toto zařízení samo požádat pomocí Announce zprávy. V tomto režimu může opět používat P2P zprávy, ale jinak by mělo mlčet a neodpovídat na dotazy režimu E2E.

Initializing – v tomto stavu je zařízení po startu. Tento stav nelze jednoduše detekovat externím pozorováním pouze PTP protokolu.

Faulty – v tomto stavu se zařízení ocitne, pokud nastane nějaká závažná vnitřní chyba. Zařízení se může pokusit tuto chybu opravit a restartovat, nebo počkat na zásah zvenčí.

Disabled – zařízení má daný port úplně vypnutý a nemělo by na něm komunikovat.

Pre-master – tento stav není nutné implementovat. Zařízení se chová jako zařízení Master, pouze ještě nezasílá synchronizační zprávy.

Uncalibrated – tento stav není nutné implementovat a slouží jako mezistupeň při nalezení nejlepších hodin. V tomto stavu se zařízení chová jako slave a může v něm zůstat po takovou dobu, dokud nebude mít dostatečně přesně synchronizovanou časovou základnu. Pak přejde do režimu master.

Passive – v tomto režimu je port, pokud je zařízení synchronizováno z jiného zdroje než PTP. V takovém případě nemá smysl se synchronizovat na zařízení Master. Zařízení tedy vyčkává a v případě výpadku hlavních hodin převezme jejich funkci. V tomto režimu může odpovídat na P2P dotazy a může je i samo iniciovat.

6.2.3 Pravidlo BMCA

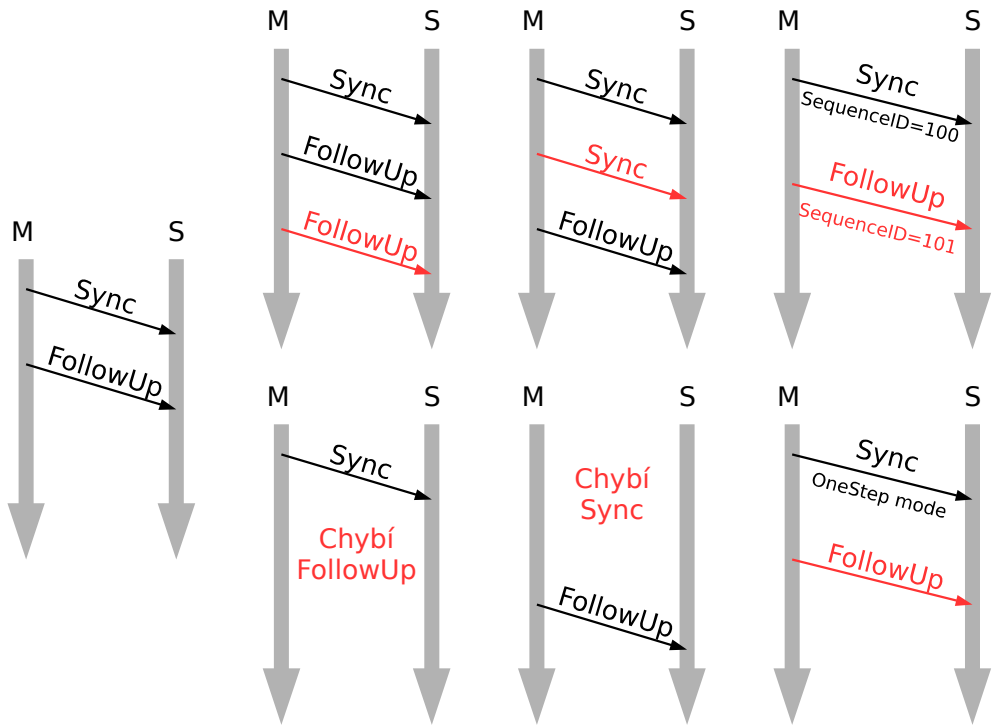
Best Master Clock Algorithm (BMCA) je jedna z důležitých vlastností PTP protokolu. Pokud nefunguje správně kvůli chybě implementace, některé zařízení může nechtěně převzít kontrolu nad synchronizací času.

Pravidlo vyhodnocuje Announce zprávy stejně jako standardní BMCA a detekuje nejlepší hodiny v daném čase. V programu jsou detekována všechna zařízení, která vyslala alespoň jednu PTP zprávu. Zařízení lze také detekovat pomocí management zpráv.

Každé nalezené zařízení je pak identifikováno a je mu přiřazen stav. Podle tohoto stavu jsou pak vykonávána další pravidla. Toto pravidlo kontroluje, jestli nevysílá někdo jiný Announce zprávy i přesto, že je evidentně horší zdroj hodin podle BMCA. Dále kontroluje aktuálně zvolené zařízení Master, jestli stále vysílá Announce zprávy a jestli je vysílá ve správné četnosti podle specifikace. Ve standardu je definována četnost a její rozmezí a při nedodržení těchto pravidel program nahlásí chybu.

6.2.4 Pravidlo Sync

Pravidlo Sync implementuje testování Sync a FollowUp zpráv. Na obr. 6.2 je znázorněn správný průběh Sync a FollowUp a příklady chyb, které mohou nastat.



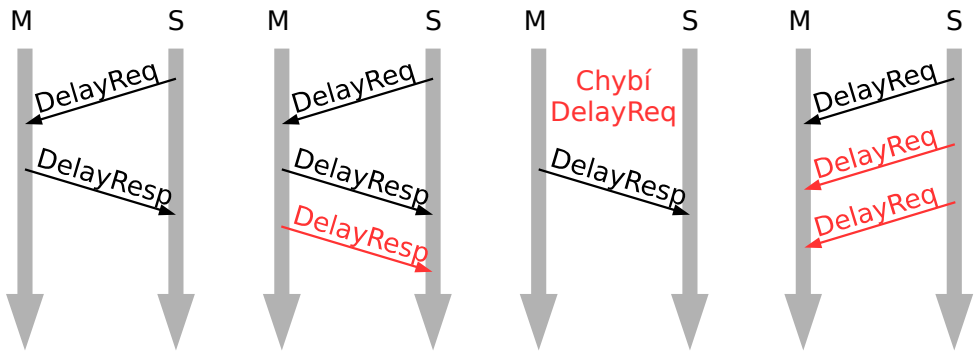
Obr. 6.2: Pravidlo Sync

Pravidlo detekuje chyby, kdy dochází k porušení standardu. Sync zprávy mohou být odeslány ve dvou režimech, tzv. One step a Two step. Ve One step režimu obsahuje Sync zpráva i čas svého odeslání a není potřeba posílat FollowUp, naopak v Two step režimu je nutnost poslat ještě FollowUp. Synchronizační zprávy musí být odesílány pravidelně s definovanou odchylkou. Pravidlo detekuje následující chyby:

- Byla detekována zpráva FollowUp, ale není k ní odpovídající zpráva Sync.
- V komunikaci je další Sync, aniž byl předchozí dokončen pomocí FollowUp.
- V komunikaci je duplicitní FollowUp nebo duplicitní Sync, který byl již přijat.
- FollowUp sice existuje, ale má špatné sekvenční číslo, takže patří k jinému Sync, nebo je chyba v sekvenčních číslech.
- V komunikaci je FollowUp, ale Sync je One step, již byl FollowUp vyslán, nebo nebyl vyslán žádný Sync.
- Byl přijat Sync nebo FollowUp, ale ještě nebyl zvolen nejlepší Master, nebo byly odeslány ze zařízení, které nebylo vybráno jako nejlepší Master.
- Četnost zasílání Sync zpráv neodpovídá deklarované hodnotě. Master zprávy odesílá příliš často nebo s příliš velkou prodlevou. Chyba je nahlášena i pokud je vysílání nepravidelné.

6.2.5 Pravidlo Delay

Pravidlo Delay sleduje režim E2E měření zpoždění přenosové cesty. Testují se pakety DelayReq a DelayResp. Na obr. 6.3 jsou znázorněny vzorová komunikace a detekovatelné chyby. Master v Announce zprávě oznámí střední hodnotu, jak často se smějí zařízení Slave doptávat. Tato hodnota se může v průběhu měnit a Master tak může korigovat počet zpráv při připojení většího množství Slave zařízení.



Obr. 6.3: Pravidlo Delay

Toto pravidlo je komplikovanější, protože se testuje pro každé zařízení zvlášť. Lze detekovat jednak neočekávané zprávy, chyby v sekvenci a střední dobu mezi DelayReq. Lze tak detekovat odpojení nebo selhání jednotky Slave a jednotky, které přetěžují Master častým doptáváním na délku přenosové cesty. Pravidlo detekuje následující případy:

- DelayReq je bez odpovědi DelayResp.
- Odpověď DelayResp je bez dotazu DelayReq.
- V datech se objeví duplicitní DelayResp nebo duplicitní DelayReq, který byl již zpracován.
- Četnost zasílání DelayReq zpráv neodpovídá požadované střední hodnotě. Master je přetěžován nebo se Slave doptává příliš málo.
- Pravidlo dále testuje, kým byly zprávy odeslány, a pokud je odeslalo špatné zařízení, je opět nahlášena chyba. Zařízení, které smí odeslat DelayReq, musí být ve stavu Slave nebo Uncalibrated. Zařízení, které smí odeslat DelayResp, musí být ve stavu Master nebo Pre-master.

6.2.6 Ostatní pravidla

Testování bylo implementováno pouze pro základní režim se zjišťováním zpoždění přenosové cesty v režimu E2E. Netestují se zde tedy P2P zprávy, protože je není možné efektivně sledovat bez dalších sond v systému. Analyzátor byl vytvořen především pro ověření konceptu testování protokolu.

Nejsou testovány ani zprávy Management a Signalling. Pokud zařízení implementuje Management zprávy, bylo by možné zjišťovat vnitřní stav přímo a nepokoušet se jej jen odhadovat ze zachycené komunikace. Mnoho zařízení ale Management zprávy neimplementuje, takže by tam nezávislé zjišťování stejně muselo být.

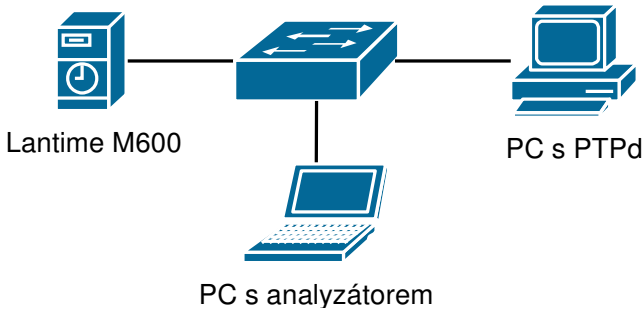
6.3 Praktické testování a výsledky

6.3.1 Podmínky pro testování

Analyzátor komunikace byl testován v laboratoři za použití zařízení Meinberg Lantime M600, počítače s PTPd a dalšího počítače s analyzátorem. Vše bylo propojeno do jednoho síťového přepínače podle obr. 6.4. Dále byla provedena řada testů nad rozsáhlejším záznamem dat, který obsahoval mnoho chyb a byl tedy vhodný pro testování celé aplikace.

6.3.2 Provedené testy

Bylo provedeno několik testů, kdy se ověřovala reakční schopnost celého analyzátoru. Tím se ověřovalo, jestli analyzátor reaguje správně na chyby způsobené rekonfigurací sítě.



Obr. 6.4: Propojení při testování

Testování maximální doby čekání – z testovacího zapojení bylo odpojeno zařízení Master a sledovalo se, jestli aplikace zahlásí problém. Chyba byla detekována a nahlášena pravidlem BMCA. Další test naopak probíhal při odpojení zařízení Slave. To bylo detekováno pravidlem Delay, protože zařízení dlouho neodeslalo žádný paket.

Testování rychlosti odesílání zpráv – na testovaném zařízení byly nastaveny různé časy pro odesílání jednotlivých zpráv. Při sledování zpráv Announce a Sync je situace jednoduchá, zprávy musejí být odesílány pravidelně s definovanou maximální odchylkou. Problematičtější je sledování četnosti zpráv DelayReq. Zprávy mohou být odeslány hned za sebou a je to v pořádku, musí být ale dodržena střední hodnota intervalu dotazování.

Testování BMCA – na testovacích zařízeních byly různě nastavovány parametry ovlivňující BMCA, jako jsou priority a deklarované stability oscilátorů. Pravidlo detekovalo změny v systému.

Testování správnosti PTP paketů – analyzátor vyhodnocuje některé příznaky a parametry zpráv a kontroluje je na přípustné hodnoty. Pokud jsou hodnoty mimo rozsah, jsou nahlášeny.

6.3.3 Výsledky testování

Všechny testy proběhly úspěšně a analyzátor detekoval všechny předpokládané problémy. Při delším testování byla objevena chyba v zařízení Lantime M600, které pravidelně zasílalo některé Sync pakety s periodou delší, než povoluje standard.

Aplikace byla testována i na záznamech komunikace z PlugFestu [40]. Nad záznamem komunikace byl proveden test a díky různorodým chybám byla opravena pravidla testování a byla tak zvýšena jejich robustnost.

6.4 Zhodnocení výsledků

Byla navržena a realizována aplikace pro testování PTP protokolu, která může sloužit jak pro on-line sledování komunikace tak pro off-line analýzu zaznamenaných dat. Její návrh je popsán v kapitole 6.1.

V on-line režimu je aplikace určena pro testování pouze multicastové komunikace. Její rozšíření na unicastovou by vyžadovalo implementaci sond v síti a rozšíření některých pravidel.

Aplikace může být dále rozšířena a použita v několika režimech. První režim je jako ověřovací nástroj nové implementace protokolu, nebo otestování změn před vydáním nové verze. V tomto režimu byla aplikace testována a s tímto účelem byla hlavně vyvíjena.

Druhý režim vychází z prvního a jde jen o aplikování stejného algoritmu na záznam dat. Lze tak zpětně ze záznamu zjistit nepatřičné chování. V tomto režimu by bylo možné aplikaci implementovat do analyzátorů síťového provozu, jako je třeba Wireshark a tím by se rozšířily možnosti sledování protokolu.

Další režim vychází z univerzálnosti aplikace a z vlastností protokolu. Pokud se použije protokol v režimu multicast a použije se režim E2E zjišťování zpoždění přenosové cesty, lze veškerou komunikaci sledovat z jednoho místa. Tento analyzátor tak může sloužit jako doplněk dohledových nástrojů celého protokolu, kdy aplikace kontroluje veškerý provoz a může hlásit výpadky komunikace, případně jiné nesrovnalosti.

Byla provedena řada testů popsaných v kapitole 6.3, které byly provedeny v předem připravených sítích. Byly detekovány komunikující strany a byla ověřována komunikace vzhledem ke standardu IEEE 1588. V tomto režimu byla odhalena chyba v komerčních hlavních hodinách Meinberg Lantime M600, které při zasílání synchronizačních zpráv nedodržovaly limity dané standardem.

V off-line režimu byla aplikace testována s daty z PlugFestu, ve kterých byla detekována řada chyb. Vzhledem k povaze PlugFestu, který slouží především pro testování, to byl očekávaný výsledek. Tato data tedy posloužila spíše pro zdokonalení vyhodnocovacích pravidel.

7 Eliminace vlivu datového zatížení přenosového média

Pokud potřebujeme časovou synchronizaci a zároveň potřebujeme mezi zařízeními přenášet nějaká data, můžeme zvolit dva základní přístupy.

První způsob je vytvoření speciálního fyzického kanálu pro přenos dat a speciálního fyzického kanálu pro vlastní synchronizaci. Tato metoda může být velice přesná ale zásadní problém v reálném nasazení jsou dvě fyzické cesty. K jejich realizaci je potřeba speciální kabeláž, dodatečná zařízení apod.

Druhý způsob využívá existující přenosovou cestu jak pro přenos dat, tak pro synchronizaci. Problém tohoto přístupu je možné zahlcení komunikačního kanálu a možné zvětšení zpoždění synchronizačních zpráv, které čekají na volný komunikační kanál.

Výhoda časové synchronizace po datovém přenosovém médiu spočívá v tom, že je možné využít stávající kabeláž a není třeba budovat separátní synchronizační kanál. Sdílení přenosové cesty má ale úskalí v podobě změny zpoždění přenosové cesty při různých zatíženích. Tomuto problému se snaží předcházet různé specifikace průmyslových Ethernetů.

Existuje několik konceptů průmyslového Ethernetu, které se snaží předcházet kolizi paketů a tím zajistit deterministické chování. Tyto implementace ale často potřebují speciální hardware pro zpracování (EtherCat [43]), nebo kladou Real Time požadavky na operační systém (RTnet [44]) kvůli příliš přesnému časování, kterého nelze v běžném systému dosáhnout.

Řešení jsou určena především pro řídicí aplikace. Pokud potřebujeme pouze měřicí aplikace a sběr dat, můžeme celý systém zjednodušit. Nejsou na něj totiž kladeny nároky na rychlou odezvu, ale pouze na přenesení velkého objemu dat při zachování přesnosti synchronizace.

Jako řešení se nabízí použít časový multiplex a pro každý typ komunikace vyhradit určitou část přenosového pásma. Tím se zaručí, že synchronizace nebude ovlivněna velkým množstvím dat a zároveň bude přesně dáno, jak velký objem lze bez problémů přenést.

7.1 Časový multiplex

Časový multiplex může převzít výhody z obou přístupů tedy separátní synchronizační kanál, nebo sdílený. Díky časovému multiplexu se komunikace rozdělí na více virtuálních kanálů, které se při komunikaci vzájemně neovlivňují. [45] [46]

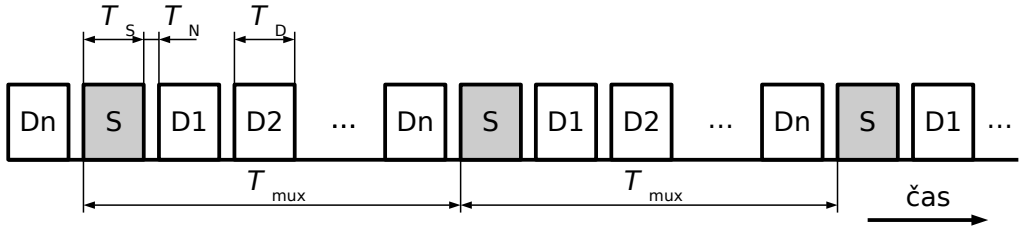
Při realizaci je třeba dbát na správnou volbu parametrů multiplexu. Pokud budou jednotlivá okna příliš malá, nebude možné dostatečně přesně

naplánovat událost vysílání dat na systémech s běžným operačním systémem. Pokud budou naopak okna příliš velká, bude se neúměrně prodlužovat celková doba k další možné komunikaci. Pokud bude implementován například systém pro sběr dat, zařízení musí mít tak velkou vnitřní paměť, aby mohlo zaznamenat veškerá data do příštího cyklu vysílání.

Aby časový multiplex správně fungoval, musí být k dispozici přesná časová synchronizace. Tato synchronizace je tedy nedílnou součástí implementace multiplexu. Koncová zařízení ji využijí jednak pro značkování naměřených dat a jednak pro realizaci vlastní komunikace.

7.2 Návrh časového multiplexu pro měřicí aplikace

Časový multiplex je rozdělen do několika fází, které se neustále pravidelně opakují. V jedné periodě multiplexu musí být zaručena možnost komunikace pro všechna připojená zařízení a zároveň možnost přenosu synchronizační zprávy.



Obr. 7.1: Časový multiplex

Na obr. 7.1 je znázorněna část multiplexu. T je perioda multiplexu, T_S je velikost synchronizačního okna, T_D je velikost datového okna a T_N je velikost bezpečnostní mezery. Okna S jsou vyhrazena pro synchronizaci a okna Dx jsou vyhrazena pro jednotlivé datové kanály. Velikost jednotlivých oken se určí podle počtu modulů, které mají komunikovat, a podle velikosti vnitřní paměti, kterou mají k dispozici. Výsledná perioda multiplexu se zvolí taková, aby odpovídala synchronizačním zprávám, nebo aby perioda synchronizace byla celočíselný násobek periody multiplexu.

Celý koncept předpokládá systém pro sběr dat, kde jednotlivé moduly sbírají a odesílají data do koncentrátoru a data obráceným směrem tečou jen minimálně.

Pro toto řešení je třeba upravit jednotlivé prvky. Hlavní hodiny musí zasílat synchronizační zprávy pravidelně a ve zvoleném okně. Pro synchronizační zprávy bylo vyhrazeno nulté okno. V dalších oknech vysílají data jednotlivé moduly. Každé okno má svůj vypočítatelný začátek a konec.

Za každým oknem je bezpečnostní mezera, která slouží k pokrytí chyb synchronizace.

V synchronizačním okně vysílají hlavní hodiny zprávu sync a podružné hodiny toto okno použijí ke zjištění délky přenosové cesty. Každé podružné hodiny se ptají na délku přenosové cesty náhodně tak, aby střední doba vycházela dle stanovených podmínek hlavních hodin. Náhodnost spočívá v tom, že si podružné hodiny vyberou synchronizační zprávu, na kterou bezprostředně odpoví dotazem na zpoždění přenosové cesty. Při velkém počtu podružných hodin by bylo možné upravit jejich software tak, aby dotaz na zpoždění nebyl bezprostřední, ale aby se i u náhodně vybrané synchronizační zprávy aplikovalo krátké náhodné zpoždění a pak by se teprve odeslala žádost o zpoždění. V tomto režimu by ale náhodná složka nesměla překročit vyhrazené okno pro synchronizace.

Pokud je potřeba jen sbírat data, nejsou kladeny žádné speciální nároky na zařízení pro sběr dat. Komunikace tohoto zařízení bude minimální a nenaruší tak ani komunikační okna synchronizace, ani okna dat. Zařízení pro sběr dat z modulů tak může být implementováno v libovolném programovacím jazyce na libovolném operačním systému. Konfigurace předpokládá, že systém pro sběr dat má k dispozici externí hlavní hodiny, které se starají o synchronizaci času.

Synchronizační zprávy se dle standardu vysílají s periodou v násobcích nebo zlomcích sekund dané vztahem

$$T_{\text{sync}} = 2^{K}, \tag{7.1}$$

kde T_{sync} je synchronizační interval a K je celé číslo, které může nabývat i záporných hodnot. Tím jsou dosaženy synchronizační intervaly kratší než 1 s. Pro periody synchronizace menší než 1 s je zřejmé, že pokud byla první synchronizační zpráva v celou sekundu, pak každá 2^{-K}-tá zpráva začne také v celou sekundu. Pokud je tedy zvolena perioda multiplexu jako

$$T_{\text{mux}} = 2^{M}, \tag{7.2}$$

kde T_{mux} je perioda multiplexu a M je celé číslo a $M \leq K$, pak vždy po 2^{K-M} synchronizačních intervalech je okno v multiplexu využito pro synchronizaci. Pro optimální využití multiplexu je pak vhodné volit $K = M$.

Pro praktické využití je vhodné volit $M \leq 0$ a tím docílit periodu multiplexu stejnou nebo kratší než 1 s. Každé připojené zařízení tak potřebuje k provozu pouze číslo přiděleného časového okna a přesný začátek sekund a podle toho si dokáže spočítat pozici všech přidělených oken v každé sekundě.

Délku periody multiplexu T_{mux} lze také zapsat jako

$$T_{\mathrm{mux}}=T_{\mathrm{s}}+N\cdot T_{\mathrm{D}}+(N+1)T_{\mathrm{N}}\,, \tag{7.3}$$

kde N je počet datových oken multiplexu a tedy maximální počet komunikujících modulů.

Velikost paměti, kterou potřebuje vzorkovací systém, vychází z jeho vzorkovací frekvence f_{vz} a počtu bajtů na jeden vzorek P_{vz}. Výsledná paměť potřebná pro pokrytí celé periody multiplexu je tak dána vztahem

$$P_{\mathrm{min}}=T_{\mathrm{mux}}f_{\mathrm{vz}}P_{\mathrm{vz}}\,, \tag{7.4}$$

kde P_{min} je minimální paměť, kterou musí zařízení mít pro ukládání vzorků. Pokud by byl použit například vzorkovací systém se vzorkovací frekvencí 50 kHz a s 8-bitovým vzorkováním, bylo by potřeba 50 kB (48,8 kiB) dat pro uložení vzorků pro odeslání v okně multiplexu.

Za předpokladu $M\leq 0$ a začátku multiplexu v celou sekundu lze jednoduše spočítat začátek každého synchronizačního okna v rámci jedné sekundy jako

$$T_{\mathrm{DZ}}(n,i)=i\cdot T_{\mathrm{mux}}+n\cdot T_{\mathrm{s}}+(n-1)\cdot T_{\mathrm{d}}\,, \tag{7.5}$$

kde $T_{\mathrm{DZ}}(n,i)$ je začátek časového okna v rámci sekundy, $n=1..N$ je číslo časového okna, $i=0..(2^{-M}-1)$ je index bloku časového multiplexu v rámci sekundy. V každé sekundě je pak 2^{-M} bloků multiplexu a v každém bloku je N datových oken a jedno okno pro synchronizaci.

Lze také určit přidělenou délku okna z parametrů multiplexu a z počtu komunikujících zařízení. Délku okna T_{D} lze vyjádřit jako

$$T_{\mathrm{D}}=\frac{T_{\mathrm{mux}}-T_{\mathrm{s}}-(N+1)T_{\mathrm{N}}}{N}\,. \tag{7.6}$$

Pokud zvolíme délku okna pro synchronizaci stejnou jako délku okna pro data, tedy $T_{\mathrm{S}}=T_{\mathrm{D}}$, lze celý vztah zjednodušit na

$$T_{\mathrm{D}}=\frac{T_{\mathrm{mux}}}{N+1}-T_{\mathrm{N}}\,. \tag{7.7}$$

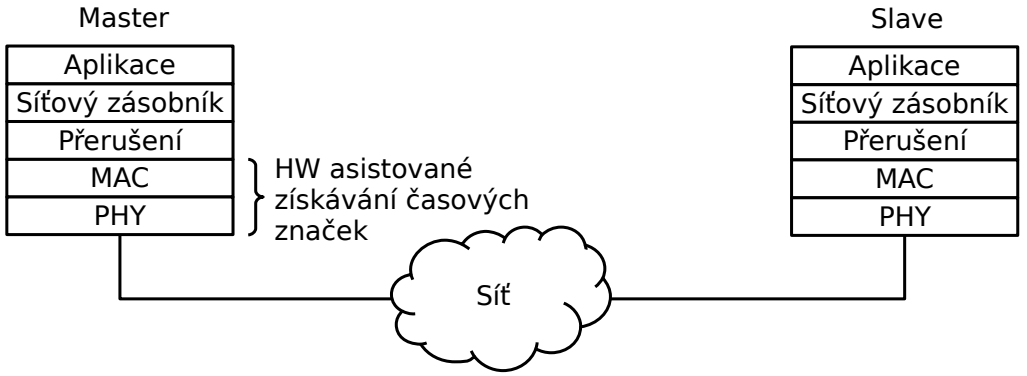
kde N je počet datových oken, T_{N} je bezpečnostní mezera a T_{mux} je délka jedné periody multiplexu.

8 Implementace a funkční vzorky

V rámci řešení projektu TAČR TA01010988 [58] bylo vyvinuto několik funkčních vzorků. Úsilí bylo věnováno především implementaci měřicích a synchronizačních zařízení pomocí mikrokontrolérů. Byly ale realizovány i čistě softwarové implementace v různých operačních systémech pro srovnání schopností systémové časové stupnice a možností synchronizace.

Pro tyto účely byly vybrány mikrokontroléry od firmy STMicroelectronics z řady STM32 především pro svoji dostupnost a podporu protokolu IEEE 1588 na hardwarové úrovni. Dále byla testována speciální ethernetová PHY.

Protokol IEEE 1588 byl také implementován na systémy Linux, Windows a RTX pro porovnání vlastností. Na Linuxu byla využita stávající implementace PTPd, ale na Windows ani RTX žádná softwarová implementace neexistovala a bylo nutné vytvořit vlastní. Pro Windows existovaly pouze implementace využívající speciální hardware, např. NI PCI-1588 [47]. Pro systém RTX neexistovaly žádné ani komerční ani otevřené implementace využívající protokol IEEE 1588.



Obr. 8.1: Znázornění získávání časových značek

Na obr. 8.1 jsou znázorněny možnosti získávání časových značek paketů. Čím nižší úroveň je k dispozici, tím přesnější časové značky lze získat. Bez hardwarové podpory lze získávat časové značky v obsluze přerušení od ethernetového bloku nebo v síťovém zásobníku.

U procesorů řady STM32 je k dispozici hardwarová podpora získávání časových značek v Media Access Control (MAC), což je hardwarová periferie přímo obsažená na čipu. U jiných typů mikrokontrolérů se musí časové značky získávat až v obsluze přerušení.

U běžných počítačů není většinou k dispozici přesné získávání časových značek přímo v síťové kartě, a tak musí být využita možnost softwarového získávání časových značek. U operačního systému Linux je

podpora zabudována přímo v jádře systému a časové značky jsou získávány na úrovni ovladače síťové karty. U operačního systému Windows ale tato podpora není a jediný způsob, jak získávat časové značky, je až na úrovni aplikace. Tím se podstatně zhorší přesnost jejich získávání.

Pro praktické použití synchronizace v systému pro sběr dat byl vytvořen odhad dosažitelné přesnosti založený na odhadu efektivního počtu bitů popsaný v kapitole 13.4.

8.1 Podpora protokolu IEEE 1588 u mikrokontrolérů STM32

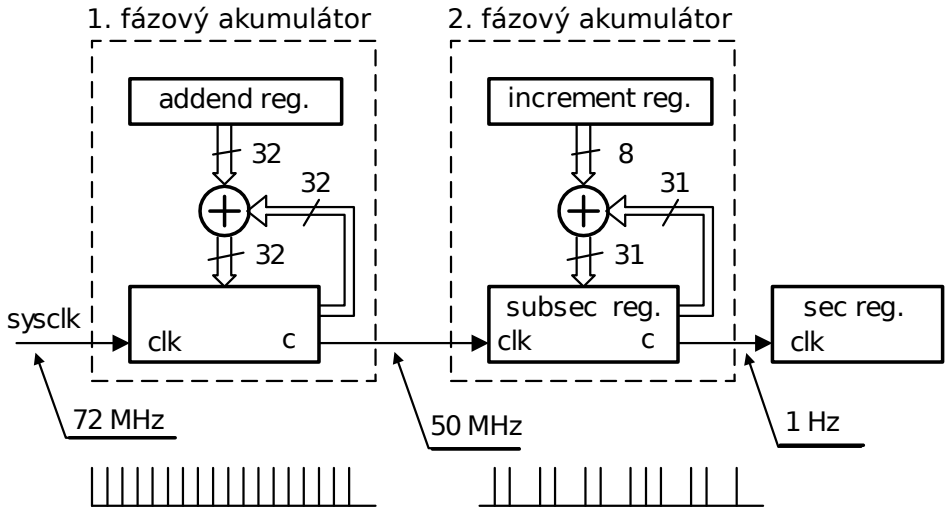
Počáteční implementace byly vytvořeny na běžně dostupných vývojových přípravcích s použitím komerčních vývojových prostředí. Na počátku vývoje byl použit mikrokontrolér STM32F107, který byl postupně nahrazen generačně novějším STM32F207 a nakonec STM32F407.

S narůstající složitostí se ale tento způsob vývoje ukázal jako nedostatečný. Byl vytvořen vlastní vývojový přípravek s procesorem STM32F107 s ohledem na možnost testování hardwarové podpory protokolu IEEE 1588 jak přímo na procesoru tak pomocí přídavných modulů. Na procesoru lze získávat časové značky v ethernetové MAC. Jako přídavný modul lze použít externí ethernetovou PHY s podporou protokolu IEEE 1588 [35].

Pro potřeby vývoje bylo také nutné vybrat vhodné vývojové prostředí. Po vyzkoušení všech v době řešení dostupných vývojových prostředí (např. Raisonance RIDE7, Keil uVision, Atollic TrueSTUDIO) bylo přistoupeno k vytvoření vlastního vývojového prostředí. Všechna komerční prostředí sice disponovala podporou daného procesoru, takže prvotní nastavení projektu bylo snadné, ale další práce na rozsáhlejších projektech byla komplikovaná kvůli nelogické správě projektů (TrueSTUDIO), nefunkční nebo neexistující podpoře automatického doplňování a podobných vyspělých vlastností, které jsou zcela běžné u vývojových prostředí pro programy na PC (RIDE7, uVision), nepodpoře verzovacích systémů, nebo vlastní nestabilitě vývojového prostředí.

Proto bylo vyvinuto zcela nové vývojové prostředí SC-IDE [48] odvozené od platformy NetBeans [49]. Díky dobrým základům v NetBeans pro editaci kódu byla pouze přidána podpora pro kompilátor jazyka C z projektu GCC ARM Embedded [50] a podpora nahrávání a ladění v mikrokontroléru pomocí OpenOCD [51].

Byla implementována podpora protokolu IEEE 1588-2002 i 1588-2008 pro mikrokontrolér STM32. Vlastní protokol byl řešen otevřenou implementací unixové služby PTPd [34], která byla upravena pro potřeby nasazení v mikrokontroléru. Implementace byla rozšířena o hardwarovou podporu tohoto protokolu v mikrokontroléru [52].



Obr. 8.2: Princip digitálního nastavení hodin v STM32 [59]

V procesoru STM32 je pro podporu generování přesných časových značek implementován dvojitý fázový akumulátor. První fázový akumulátor slouží k jemnému nastavení výstupní frekvence, která vstupuje do druhého fázového akumulátoru, který již slouží přímo k inkrementaci sub-sekundového a sekundového registru. Nejprve je třeba spočítat výchozí hodnoty *Addend* a *Increment* registru pro danou frekvenci procesoru a pro zvolený čas, o který se bude zvyšovat sub-sekundová část τ.

Systém na obr. 8.2 lze popsat pomocí rovnic

$$\tau = \frac{Increment}{2^{31}}, \tag{8.1}$$

$$Addend \cdot Increment = \frac{2^{63}}{SysClk}. \tag{8.2}$$

Pro zvolené τ lze odvodit hodnoty jednotlivých registrů. Vzhledem k tomu, že registry jsou celočíselné, je třeba tuto hodnotu zaokrouhlit.

$$Increment = round\left(\tau \cdot 2^{31}\right) \tag{8.3}$$

$$Addend = round\left(\frac{2^{63}}{SysClk \cdot round\left(\tau \cdot 2^{31}\right)}\right) \tag{8.4}$$

$$\tau_r = \frac{round\left(\tau \cdot 2^{31}\right)}{2^{31}} \tag{8.5}$$

Z (8.3) a (8.4) lze odvodit základní hodnoty registrů a podle (8.5) lze zpětně dopočítat reálnou dobu, o kterou se bude zvětšovat sub-sekundová část. Pokud je například frekvence hodin $SysClk = 72\,\mathrm{MHz}$ a počáteční velikost přírůstku zvolena $\tau = 20\,\mathrm{ns}$, pak výsledné hodnoty registrů budou následující:

$$Increment = round\left(20 \cdot 10^{-9} \cdot 2^{31}\right) = 43 \tag{8.6}$$

$$Addend = round\left(\frac{2^{63}}{72 \cdot 10^{6} \cdot 43}\right) = 2979125335 \tag{8.7}$$

$$\tau_{\mathrm{r}} = \frac{43}{2^{31}} = 20{,}023\,\mathrm{ns}\,. \tag{8.8}$$

Hodnota registru *Increment* se již nebude měnit. Hodnota registru *Addend* bude sloužit k jemnému doladění frekvence hodin. Pro potřeby napojení na PTPd je třeba dodefinovat funkci, která má jako vstupní parametr odchylku Adj hodin v ppb (parts per billion 10^{-9}). Je tedy potřeba tuto odchylku přepočítat na novou hodnotu registru $Addend_{\mathrm{adj}}$.

$$Addend_{\mathrm{adj}} = \frac{Addend}{1 - Adj} \tag{8.9}$$

Maximální odchylku hodin, jakou je tímto způsobem možné dosáhnout lze spočítat podle vztahu

$$Adj = \frac{Addend_{\mathrm{adj}} - Addend}{Addend_{\mathrm{adj}}}\,. \tag{8.10}$$

Dosazením minimální a maximální hodnoty registru *Addend* lze získat maximální a minimální odchylku, kterou je možné tímto způsobem nastavit. Pro maximální hodnotu registru a předchozí příklad vychází maximální možná nastavitelná odchylka

$$Adj = \frac{(2^{32} - 1) - Addend}{1} = \frac{(2^{32} - 1) - 2979125335}{(2^{32} - 1)} = 30{,}6\,\%\,. \tag{8.11}$$

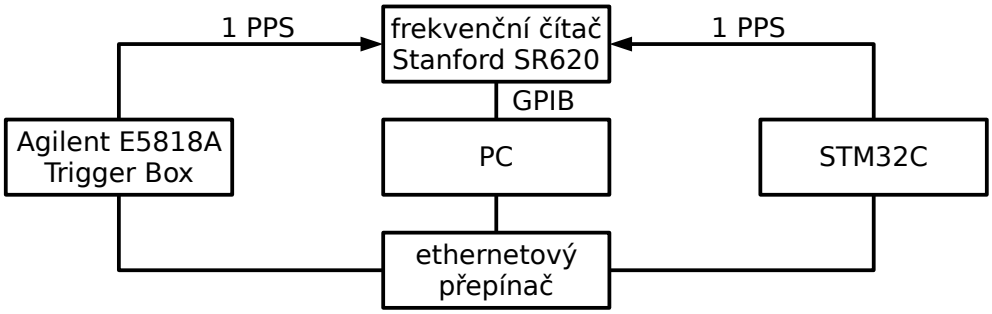
Tato hodnota plně postačuje k pokrytí odchylek běžných oscilátorů a krystalů, která se pohybuje do 100 ppm.

Využití hardwarové podpory protokolu IEEE 1588 v procesoru STM32 je ale velice omezené. Synchronizovanou časovou stupnici lze použít pouze k hardwarovému generování PPS výstupu a ke generování jednoho přerušení v definovaný čas. Ke značkování externích událostí a ke generování více externích impulzů v přesný čas je potřeba vše řešit pomocí přerušení a softwarové obsluhy.

Výsledky této implementace byly použity při tvorbě aplikační poznámky AN3411 pro STMicroelectronics [53].

8.1.1 Porovnání hardwarové podpory v STM32

Procesor STM32 bylo možné využít pro zjištění, jaký skutečný vliv má jeho hardwarová podpora protokolu IEEE 1588. Byl uspořádán experiment, kdy se zařízení s procesorem STM32 synchronizovalo na Trigger Box firmy Agilent a byl měřen jejich výstup 1 PPS podle obr. 8.3.



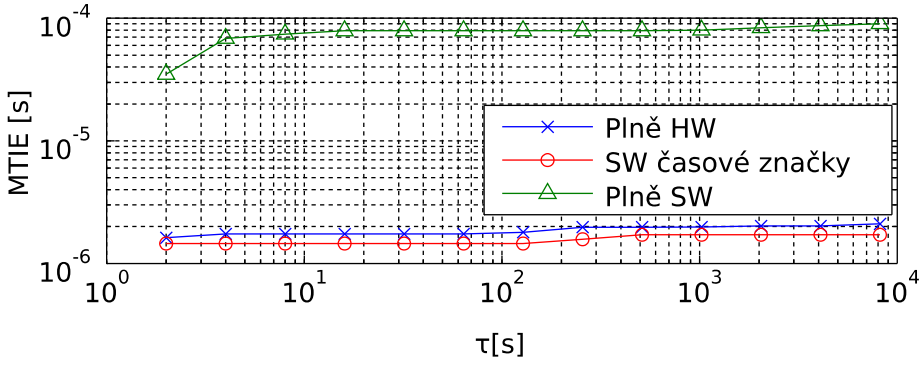
Obr. 8.3: Uspořádání experimentu měření přesnosti synchronizace

Byly proměřeny tři způsoby synchronizace. První způsob byl s plnou hardwarovou podporou, kdy se pakety značkují přímo v MAC s přesností 20 ns a hodiny lze nastavit s krokem 1 ppb.

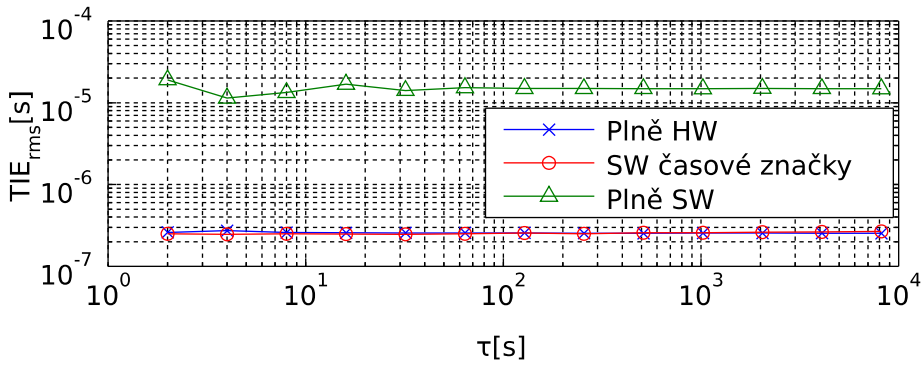
Druhý způsob využíval stejný přesný čas, ale časové značky paketů byly generovány až v obsluze přerušení při příjmu paketu pomocí software.

Třetí způsob byl čistě softwarový a nevyužíval žádnou část hardwarové podpory. Byl použit běžný čítač, jehož frekvenci čítání bylo možné nastavit s krokem 30 ppm, a byl inkrementován každých 300 ns.

Z grafů na obr. 8.4 a 8.5 je zřejmé, že nejlepší přesnosti synchronizace bylo dosaženo s použitím přesné a jemně nastavitelné časové základny v procesoru určené pro tyto účely. Hodnoty metrik TIE_{rms} a MTIE pro získávání časových značek pomocí hardwarové podpory nebo až v přerušení vycházejí totožně.



Obr. 8.4: MTIE různých způsobů synchronizace s STM32



Obr. 8.5: TIErms různých způsobů synchronizace s STM32

Rozdíl je pouze v tom, že softwarové získávání časových značek vykazovalo asymetrii, takže výsledná časová stupnice byla sice stabilní, ale o 1,2 µs posunutá, na rozdíl od plně hardwarového řešení, kde průměrná odchylka od hlavních hodin byla pouze 8 ns.

8.1.2 Porovnání rychlosti synchronizačních paketů

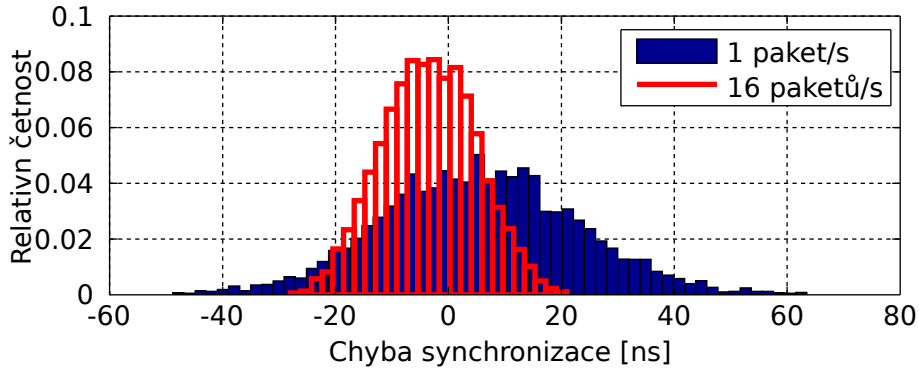
Pro různé rychlosti synchronizačních paketů vycházela výsledná přesnost různě. Čím častější byly synchronizační zprávy, tím menší rozptyl měl signál PPS oproti hlavním hodinám. Přírůstek sub-sekundového registru byl nastaven na 20 ns podle (8.3), čímž byla určena rozlišovací schopnost časových značek.

Bylo provedeno několik měření, kdy byl k internímu oscilátoru procesoru připojen pouze externí krystal, nebo byl použit externí oscilátor.

Tab. 8.1: Naměřené hodnoty přesnosti synchronizace STM32 pro různé konfigurace (Osc. – s externím oscilátorem, Krystal – s interním oscilátorem a externím krystalem)

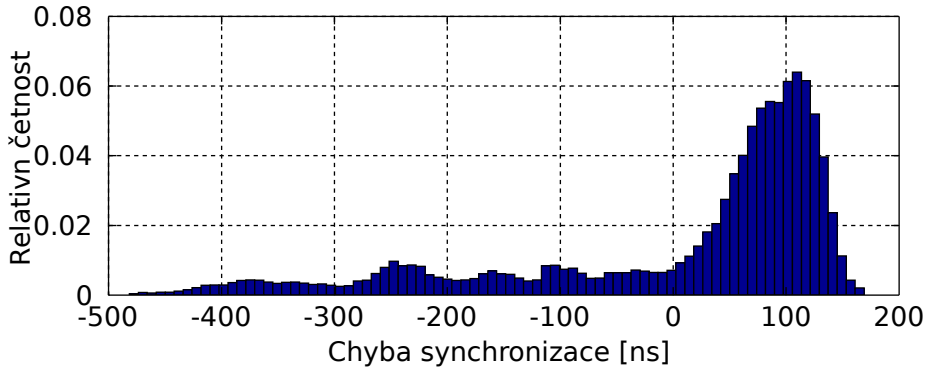
Konfigurace	Průměr	Směrodatná odchylka	MTIE
Osc. 1 paket/s	6 ns	17 ns	113 ns
Osc. 16 paketů/s	-3 ns	8 ns	49 ns
Krystal 1 paket/s	8 ns	146 ns	651 ns

Z hodnot uvedených v tab. 8.1 vyplývá, že rychlost synchronizace již nemá takový vliv na přesnost synchronizace. Je to dáno především rozlišovací schopností časových značek, která dosažitelnou přesnost limituje. Pokud se zvýšila četnost synchronizačních paketů 16×, snížila se maximální chyba synchronizace pouze na polovinu. Naměřené histogramy chyb synchronizace jsou uvedeny na obr. 8.6.



Obr. 8.6: Histogram měřeného PPS při použití externího oscilátoru.

Díky hardwarové chybě v procesoru je ale zásadní rozdíl při použití externího oscilátoru nebo interního oscilátoru s externím krystalem. Důvod zhoršení v případě použití pouze krystalu je diskutován v kapitole 8.1.3. Naměřený histogram chyb synchronizace je uveden na obr. 8.7.



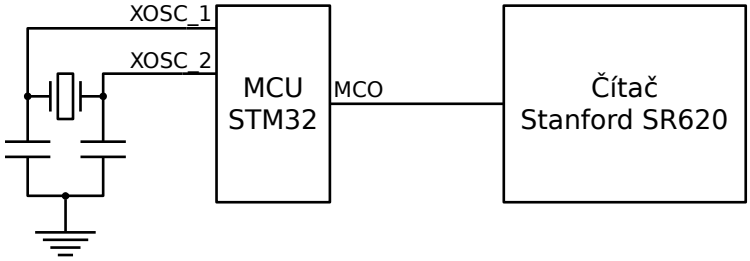
Obr. 8.7: Histogram měřeného PPS při použití krystalu

8.1.3 Nestabilita krystalového oscilátoru u STM32

Při implementaci byl nejprve použit běžný krystal a byl využit oscilátor v procesoru STM32. Po několika měřeních se ukázalo, že krystal nekmitá stabilně na jedné frekvenci, ale že s periodou zhruba jedné minuty prudce vzroste frekvence o 5 až 9 Hz a pak pomalu klesá na původní hodnotu. V normálním provozu se tato chyba neprojeví, ale při přesném časování se jedná o viditelný problém, který generuje chybu synchronizace řádově stovky ns.

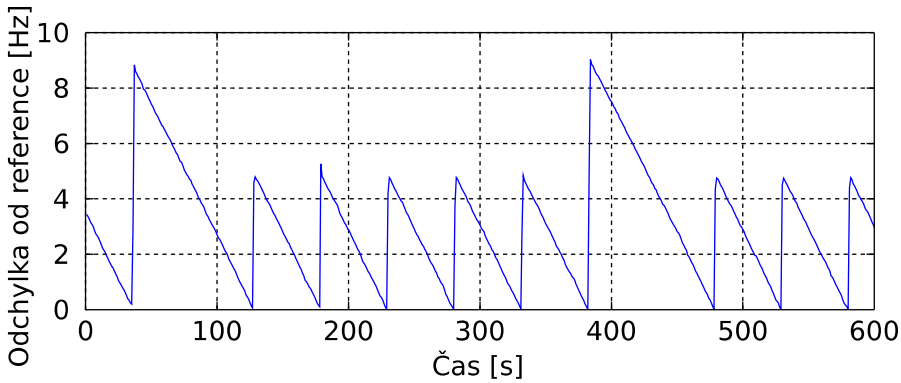
Pro otestování byl použit krystal o frekvenci 25 MHz. HSE oscilátor v procesoru, který používal připojený krystal, byl pomocí vnitřní struktury procesoru přímo napojen na externí výstup MCO, viz obr. 8.8. Tento výstup může být nakonfigurován tak, aby následoval vybraný zdroj vnitřních hodin. Výstup z MCO byl přímo měřen pomocí čítače Stanford SR620 a byla

zaznamenávána naměřená frekvence. Frekvence nebyla měřena přímo na krystalu, aby se vlastním měřením oscilátor neovlivňoval.



Obr. 8.8: Blokové schéma testování oscilátoru STM32

Použití kompletního externího krystalového oscilátoru tuto chybu zcela eliminovalo. Bohužel na tuto chybu není uživatel upozorněn v žádném dokumentu. Tato chyba je navíc závislá na použitém krystalu. Skoky ve frekvenci se pro různé krystaly pohybovaly v rozmezí 2 až 16 Hz. Příklad skokového kolísání frekvence je na obr. 8.9.



Obr. 8.9: Kolísání frekvence oscilátoru STM32 v čase (odchylka od základní frekvence 24,999408 MHz)

8.1.4 Zhodnocení implementace v STM32

V rámci zvolené granularity a vlastností hardwarové implementace fungovala samotná synchronizace velice dobře. Pakety byly značkovány konzistentně a bylo možné odečíst konstantní asymetrii. Po vyřešení problému s oscilátorem se celá synchronizace zpřesnila.

V používaném procesoru byla ale podpora protokolu IEEE 1588 velice omezená vzhledem k dalšímu použití. Bylo sice možné dosáhnout přesné časové synchronizace vnitřních hodin, ale již bylo problematické tento čas reálně využít. Pro generování událostí existoval pouze jeden hardwarový výstup, který mohl přímo generovat pouze signál PPS. Dále bylo možné vytvořit jedno přerušení, které mohlo spustit nějakou událost. Toto přerušení ale muselo být obslouženo softwarově, a tím se mohla narušit

jinak vysoká přesnost synchronizace. Interní přerušení bylo možné navázat jako hardwarovou událost pouze pro vnitřní čítač/časovač TIM2, ten ale vývodově kolidoval s připojenou ethernetovou MAC, takže byl k dispozici pouze jeden skutečně hardwarový výstup, který bylo možné programovat na libovolnou událost.

Získávání časových značek také nebylo jednoduše možné, protože přesná vnitřní časová stupnice nedisponuje vlastností input capture, takže nemůže uložit časovou značku k externí události. To lze opět realizovat pouze přes přerušení nebo pomocí DMA.

Zásadním problémem přitom je, že nelze korektně vyčíst aktuální čas, protože je uložen ve dvou 32-bitových registrech. V mikrokontroléru není implementována vlastnost zablokování jednoho registru při vyčtení druhého. Při jejich postupném vyčtení tedy může nastat změna vnitřních hodin. Dále nelze udělat interně záznam obou registrů najednou a pak je postupně vyčíst. Musí se vyčítat jednotlivě a na několikrát, a pak podle hodnot odhadovat, jaká hodnota byla skutečně vyčtena. Byl proto vytvořen jednoduchý postup, který zaručí bezpečné vyčtení časové značky, i když to přímým způsobem není možné.

Vyčítání aktuálního času je tedy třeba rozdělit na tři fáze:

- vyčtení sekundového registru do proměnné *T1*,
- vyčtení sub-sekundového registru do proměnné *TS*,
- vyčtení sekundového registru do proměnné *T2*.

Pokud je hodnota *T1* shodná s hodnotou *T2*, není třeba již další porovnání a skutečně vyčtený čas je dvojice (*T1*, *TS*). Toto je nejčastější případ v reálném nasazení.

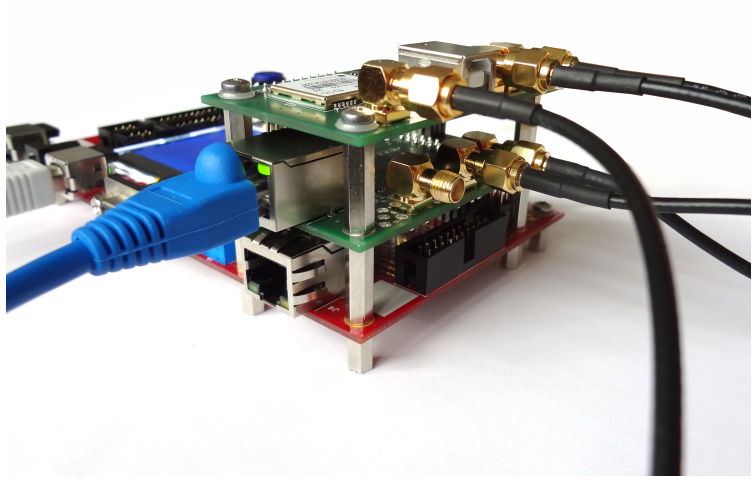
Pokud se ale hodnoty *T1* a *T2* liší, pak došlo k inkrementaci sekundového registru právě ve chvíli, kdy byl vyčítán sub-sekundový registr. Nevíme ale, jestli k inkrementaci došlo před vyčtením sub-sekundového registru nebo až po něm. Je proto nutné porovnat hodnotu *TS*, jestli se blíží více k celé sekundě nebo k nulové hodnotě. Pokud se blíží k nulové hodnotě, pak správný čas je dvojice (*T2*, *TS*). Pokud se blíží k celé sekundě, pak správný čas je (*T1*, *TS*).

8.1.5 Implementace s použitím ethernetové PHY s podporou IEEE 1588

Hardwarová podpora v procesoru a jeho MAC není jediný způsob, jak dosáhnout přesného značkování paketů. Díky popisovaným vlastnostem a omezenému využití přesného času v procesoru byla testována i možnost použití externí ethernetové PHY s podporou protokolu IEEE 1588.

Existují i specializované ethernetové PHY, které mají podporu již na takto nízké úrovni. Externí obvody navíc disponují pokročilým generováním

externích impulzů, náběžných a sestupných hran v přesný čas, generováním signálu s přesnou frekvencí a hardwarové značkování vstupních událostí.



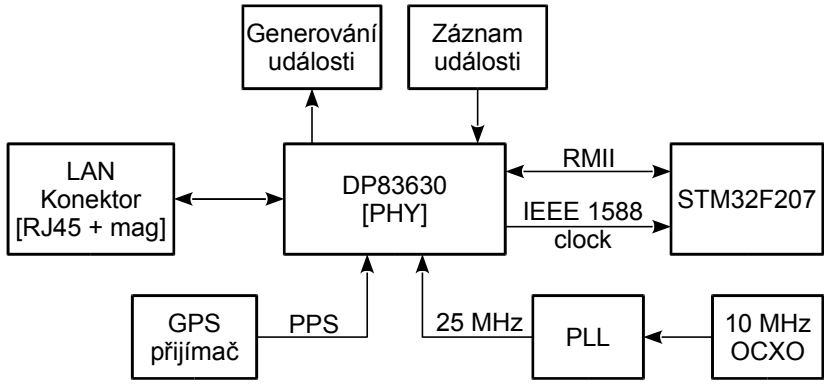
Obr. 8.10: Zařízení s podporou IEEE 1588 s DP83630

Platforma pro testování byla rozšířena o modul s PHY DP83630 [30], který disponuje výše jmenovanými vlastnostmi. Ukládání časových značek se děje na nižší úrovni. Obvod dále disponuje zdrojem synchronizovaného hodinového signálu, který lze zpětně zavést do procesoru. Tímto způsobem lze celý procesor provozovat na frekvenci, která je synchronizovaná na hlavní hodiny a všechna časování tak trvají definovaný čas.

Vzniklo experimentální zařízení založené na Sleepy Cat KITu [54] a je ukázáno na obr. 8.10. Zařízení zároveň obsahuje GPS přijímač uBlox LEA 6T, takže může být použito jako hlavní hodiny. O přesné měření časových značek se stará obvod DP83630. Blokové schéma celého experimentálního zařízení je ukázáno na obr. 8.11. Hlavním prvkem tedy již není procesor, ale ethernetová PHY, která se stará o přesné časování a procesor ji jen konfiguruje.

Podpora této ethernetové PHY byla zahrnuta do protokolového zásobníku modifikovaného PTPd. Pro komunikaci s PHY byla použita modifikovaná knihovna TI-EPL [55].

Ethernetová PHY DP83630 umožňuje získávání časových značek s rozlišovací schopností 8 ns, generování výstupních událostí v definovaný čas (náběžná hrana, sestupná hrana), generování signálů o dané frekvenci a střídě odvozených ze synchronizované časové základny, generování referenčního signálu s frekvencí od 1 MHz do 125 MHz a jemné nastavení frekvence časové základny s krokem 2^{-32} ns na každou periodu referenčního signálu. Pro referenční signál 25 MHz odpovídá možnost nastavení kroku 0,058 ppb.



Obr. 8.11: Blokové schéma zařízení s podporou IEEE 1588

PHY disponuje osmi univerzálními vstupy a výstupy. Pokud nejsou potřeba indikační LED, pak se dají použít další tři univerzální vstupy a výstupy. Dále disponuje jedním výstupem referenčních hodin, který je možné zapojit přímo na vstupní hodiny, nebo na vnitřní korigované hodiny. To otevírá možnost použít tento hodinový signál jako vstup pro hodinový signál procesoru. Po provedení tohoto nastavení budou všechny časovače v procesoru počítat s přesnou frekvencí. Přesný čas pak lze jednorázově korigovat.

Komunikace mezi PHY a procesorem se obecně děje po dvou kanálech, jeden kanál jsou vodiče MDC/MDIO, což je jednoduchá synchronní sběrnice pro přenos řídicích příkazů, a druhý kanál je hlavní datový přes MII/RMII. Tato PHY umožňuje předávání časových značek přes obě rozhraní. Standardní způsob přes MDC/MDIO přistupuje k registrům PHY přímo. Při použití datového rozhraní jsou použity speciální pakety nazývané PHY Control Frames, které PHY neodesílá, ale interpretuje.

8.2 Zařízení pro synchronizaci s podporou protokolu IEEE 1588

Pomocí znalostí z řešení předchozích problémů byl vyvinut a realizován funkční vzorek zařízení SyncBox [56] pro synchronizaci s podporou protokolu IEEE 1588. Zařízení dále disponuje vstupy pro značkování externích událostí a výstupy pro generování impulzů a výstupních signálů o zvolených parametrech.

Zařízení lze nastavovat přes ethernetové připojení. Ke komunikaci se zařízením je použit standard SCPI-99 [57]. Komunikační standard je implementován pomocí knihovny SCPI parser [58], která vznikla jako dílčí výsledek řešení.

Na obr. 8.12 je vidět realizace zařízení SyncBox a vnitřní uspořádání. Moduly jsou realizovány v podobě zásuvných karet, které lze podle potřeby vyměnit a nahradit jinou variantou.

Obr. 8.12: Zařízení SyncBox

8.2.1 Vstupní a výstupní události a hodinové signály

Na obr. 8.13 je zobrazeno blokové schéma celého zařízení. Zařízení je vytvořeno univerzálně tak, aby jakákoli jednotka mohla generovat výstupní signály a aby jakákoli jednotka mohla značkovat vstupní události. V celém systému je totiž několik modulů, které to umí nezávisle na sobě.

Vstup ANT slouží pro zapojení antény GPS přijímače. Vstup EXT CLK slouží pro zapojení vstupního referenčního hodinového signálu. Přes multiplexery lze tento signál vybrat a pomocí PLL následně odvodit libovolnou frekvenci v rozsahu 1 MHz až 100 MHz. PLL může fungovat i v režimu, kdy pouze propouští vstupní signál na výstup.

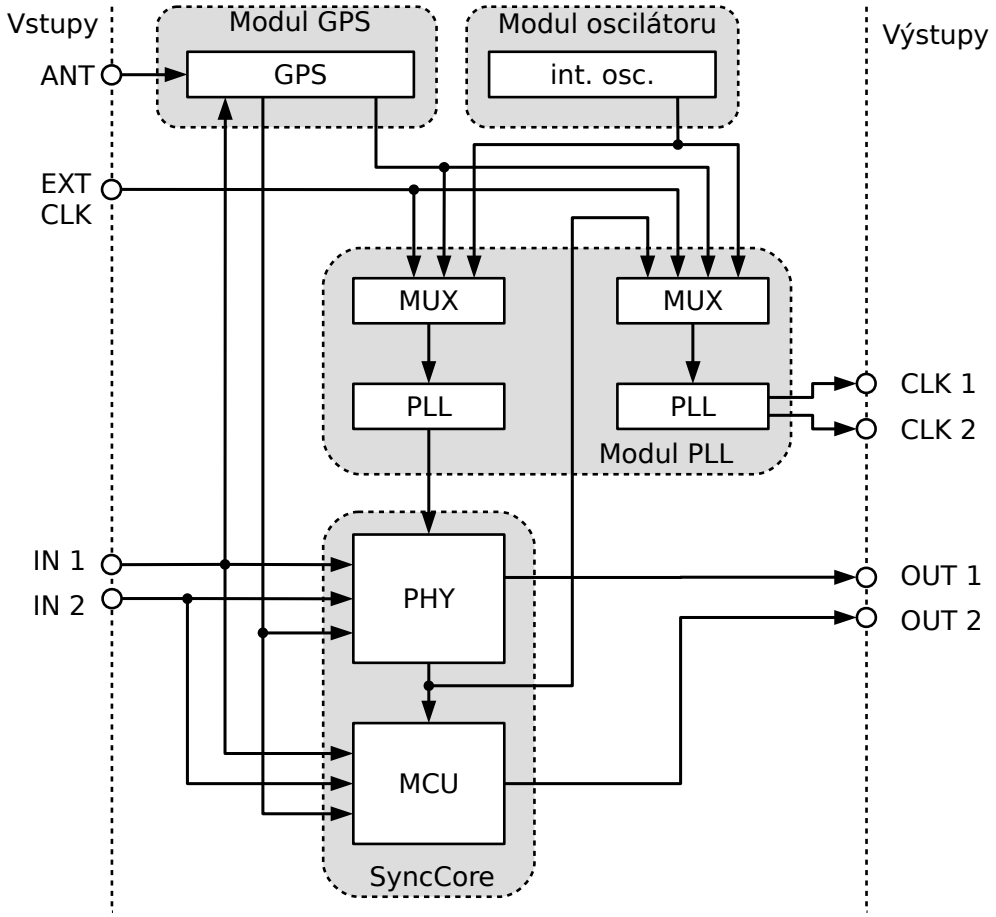
Vstupy IN1 a IN2 slouží pro značkování externích událostí. Tyto vstupy jsou přivedeny do všech modulů, které umí značkovat vstupní události. Lze tak vybrat vhodný modul podle situace nebo porovnat vlastnosti jednotlivých modulů.

Výstupy CLK1 a CLK2 jsou výstupy PLL a mohou být napojeny na jeden ze čtyř zdrojů hodin. Výsledná frekvence může být navíc pomocí PLL upravena na libovolnou požadovanou výstupní frekvenci. Výběr signálu je

tvořen výstupem synchronizovaných hodin z PHY, externím vstupem, výstupem synchronizovaných hodin z GPS nebo přímo výstupem vnitřního oscilátoru.

Výstupy OUT1 a OUT2 slouží pro generování událostí a časových kódů. Výstupy lze naprogramovat tak, aby v určený čas vytvořily náběžnou nebo sestupnou hranu nebo periodický signál s definovaným počátkem/fází, frekvencí a střídou.

Výstup OUT2 lze navíc použít pro generování časových kódu jako např. IRIG-B. Procesorový modul lze nastavit tak, aby běžel na synchronizované frekvenci, a jakékoli časové průběhy, které generuje, jsou tedy také synchronizované.



Obr. 8.13: Blokové schéma celého zařízení SyncBox

Podobně, jako jsou generované signály CLK, lze generovat i hodinový signál do procesorového modulu. Tento hodinový signál lze odvodit od všech zdrojů hodin jako výstupy CLK. Hlavní rozdíl je tedy v nastavení PLL, která zde slouží ke generování hodinového signálu 25 MHz z velkého

rozsahu vstupních referenčních hodin. Lze tak například použít stabilní externí oscilátor hodinový signál 10 MHz a ten pak pomocí PLL vynásobit na 25 MHz pro procesorový modul.

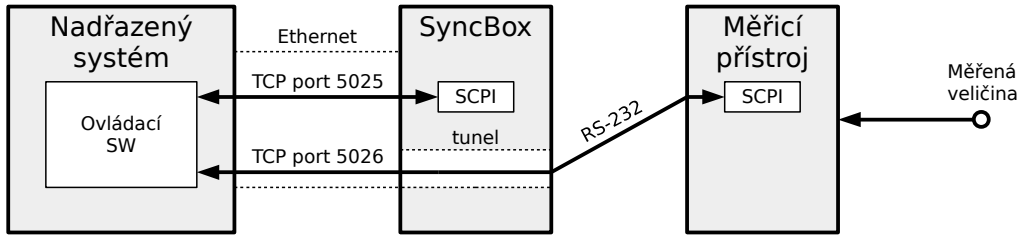
Značkování a generování událostí umí samotný GPS přijímač. GPS přijímač disponuje i funkcionalitou generování periodického hodinového signálu odvozeného od synchronizovaných hodin. Informace z přijímače lze vyčítat po sériové lince a dále zpracovávat.

Stejnou funkcionalitu umožňuje i ethernetová PHY DP83630. Ta může značkovat jednorázové vstupní události a nebo generovat jednorázové a periodické výstupní události.

Pokud je procesorová jednotka zapojena tak, že hlavní hodiny jsou odvozeny od synchronizované reference, pak všechny operace v jednotce jsou také synchronizované. Lze tak jednoduše vytvořit zpoždění pomocí časovače, které bude přesně odpovídat synchronizované časové stupnici. V tomto režimu lze pak procesor také použít pro značkování externích událostí a generování přesně časovaných průběhů.

8.2.2 Doplňkové funkce

SyncBox je dále vybaven sériovým portem, který slouží pro připojení dalších přístrojů. Tento sériový port není přímo ovládán, ale data jsou předávána pomocí TCP spojení nadřazenému systému. Tento mechanizmus se nazývá „tunelování“ a jeho princip je znázorněn na obr. 8.14.



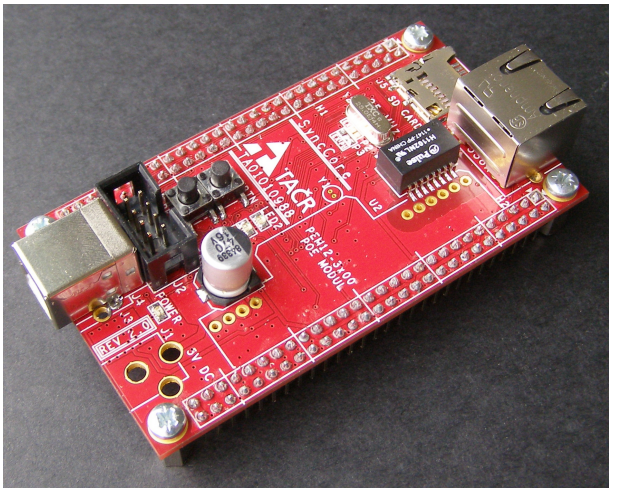
Obr. 8.14: Mechanizmus tunelování spojení na měřicí přístroj

Nadřazený systém komunikuje přímo se SyncBox a pomocí zprostředkovaného tunelu také přímo s dalším přístrojem. Tento mechanizmus je vhodný pro rozšíření funkcionality existujícího měřicího přístroje, který nemá Ethernet.

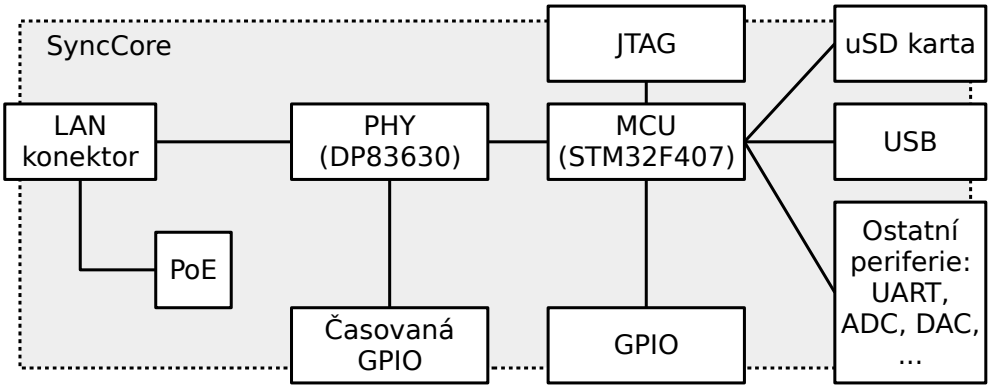
SyncBox implementuje i funkci „tunelování“ po SPI sběrnici. V zařízení jsou dvě sběrnice, jedna se chová jako master a druhá jako slave. Připojené zařízení si tedy samo určuje odesílání dat. Tento režim slouží k připojení dalších procesorových modulů, které zvládají sbírat data rychleji, než je možné přenést po sériové lince, ale zároveň nemají k dispozici Ethernet.

8.2.3 Procesorový modul SyncCore

Hlavní komponenta celého zařízení je deska SyncCore, viz obr. 8.15, která je použita v dalších funkčních vzorcích. SyncCore je osazen procesorem STM32F407 a obsahuje ethernetovou PHY DP83630 s podporou protokolu IEEE 1588. Modul lze dále osadit PoE modulem, a tak celý SyncCore napájet přímo z datového kabelu. Pro potřeby ukládání dat je k dispozici konektor na microSD kartu. Všechny nepoužité piny procesoru jsou vyvedeny na rozšiřující dvouřadé konektory po stranách. Na tyto konektory jsou také vyvedeny vstupy a výstupy událostí z ethernetové PHY, viz obr. 8.16.



Obr. 8.15: Modul SyncCore



Obr. 8.16: Blokové zapojení modulu SyncCore

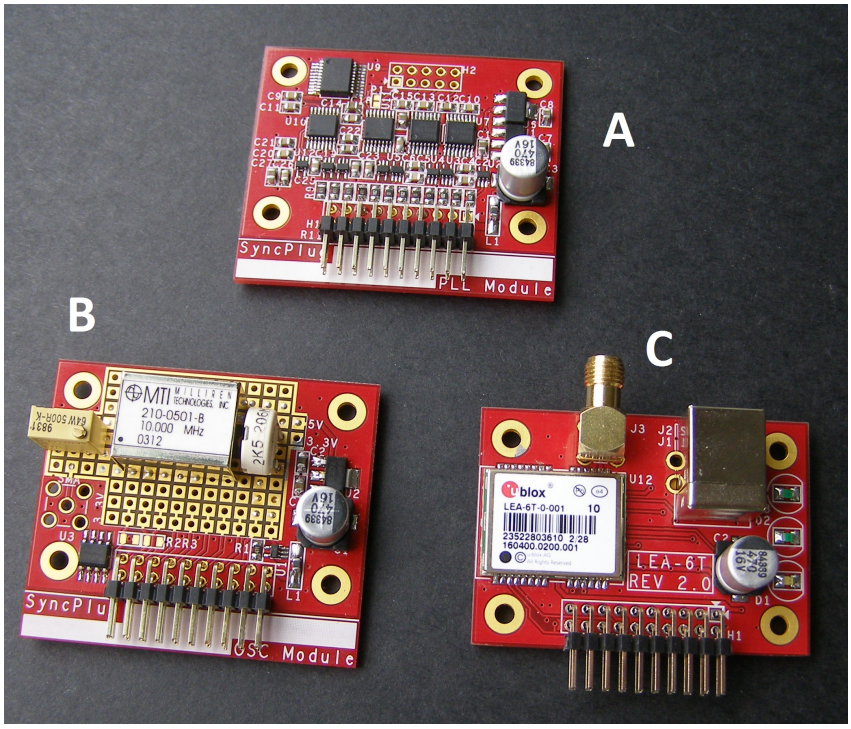
8.2.4 Doplňkové moduly SyncPlug

Zařízení je modulární a klíčové komponenty lze nahradit za jiné. Lze například vyměnit nebo vynechat GPS přijímač, nebo lze zvolit jiný oscilátor. Každý zásuvný modul má na sobě osazenou EEPROM s daty o sobě, takže procesor může určit vhodné propojení vnitřních komponent samostatně bez nutnosti manuální konfigurace po výměně komponenty. Na obr. 8.17 jsou ukázány některé existující moduly.

Modul řízení hodin obsahuje multiplexery a PLL, lze tak propojit externí vstupy na správné vstupy vnitřních modulů, nebo naopak napojit externí výstupy na výstupy vnitřních modulů. Jednotky PLL pak slouží pro úpravu hodinového signálu pro interní použití a zároveň pro generování požadovaných frekvencí na výstupu.

Modul oscilátoru je realizován jako univerzální a je na něm možné experimentovat s různými oscilátory. Na obrázku je příklad termostatovaného oscilátoru MTI-Milliren MTI 210 [36].

Modul GPS přijímače je osazen přijímačem uBlox LEA-6T. Tento přijímač je speciálně určen pro časové účely a má k dispozici vstupy a výstupy pro značkování a generování událostí a pro generování signálů o zvolené frekvenci.



Obr. 8.17: Jednotlivé moduly SyncPlug a) modul řízení hodin, b) modul termostatovaného oscilátoru, c) modul GPS přijímače.

8.2.5 Funkce hlavních hodin – SyncBox GPS

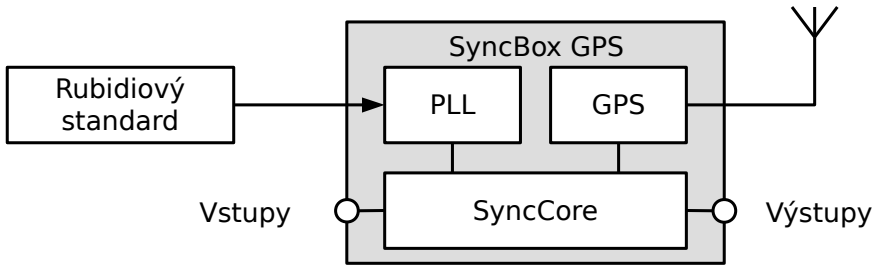
Ve funkci hlavních hodin obsahuje SyncBox zabudovaný GPS přijímač. Jako zdroj hodin lze použít vnitřní termostatovaný oscilátor, tepelně kompenzovaný oscilátor nebo externí zdroj hodin, například z Rb standardu. Zařízení se chová jako hlavní hodiny protokolu IEEE 1588 a pravidelně vysílá synchronizační pakety, pokud v síti není lepší zdroj hodin dle BMCA.

Hlavní hodiny disponují stejnou sadou vstupů a výstupů jako podružné hodiny, takže je možné využít i je. Lze například generovat odvozené signály od referenčních hodin, jako je 1 PPS signál, nebo různé časové kódy pro zastaralé systémy, které nemají Ethernet. Časový kód může být například IRIG-B.

Výstupní signály jsou synchronizované na GPS. Zvolený referenční zdroj hodin (interní nebo externí) slouží pouze pro zaručení krátkodobé stability a pro režim překlenutí doby, kdy dojde k výpadku signálu GPS.

V režimu hlavních hodin přejímají přesnost zvoleného přijímače a zvolených referenčních hodin. Touto problematikou se dále zabývá [6] a [59]. Při použití přijímače uBlox LEA-6T byla na výstupu 1PPS naměřena přesnost proti UTC(TP) ±50 ns RMS a 200 ns špička.

Na obr. 8.18 je uveden příklad zapojení SyncBox GPS. K zařízení lze připojit externí zdroj stabilního signálu, např. rubidiový standard, a celé zařízení použít jako oscilátor řízený GPS. Vstupy a výstupy zařízení fungují jako u standardního zařízení SyncBox a lze je tedy využít pro získávání časových značek nebo pro generování dalších událostí a hodinových frekvencí.



Obr. 8.18: Blokové schéma SyncBox GPS

8.2.6 Funkce podružných hodin

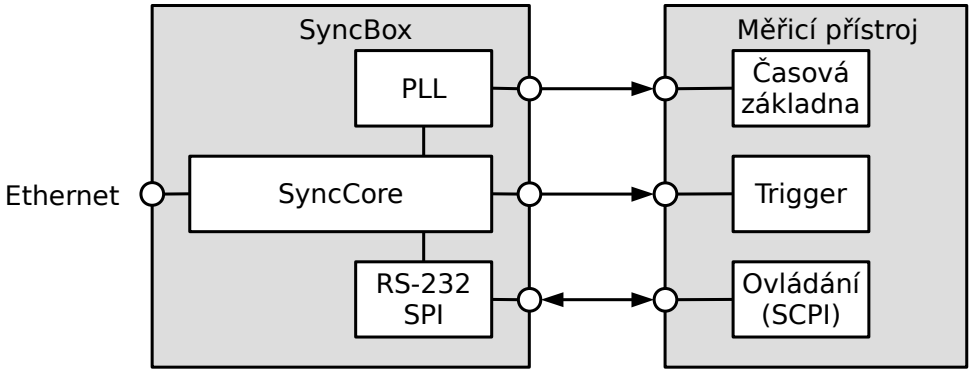
Ve funkci podružných hodin může být zařízení provozováno s obyčejným nebo tepelně kompenzovaným oscilátorem. Frekvence a čas jsou synchronizovány pomocí protokolu IEEE 1588. Zařízení je například určeno pro rozšíření funkcionality laboratorních měřicích přístrojů, které disponují pouze vstupem pro synchronizační signál nebo vstupem pro spuštění odměru – trigger. Laboratorní přístroje jsou navíc většinou vybaveny

portem GPIB a sériovou linkou. Právě sériovou linku je možné k SyncBoxu také připojit. Všechny příkazy do přístroje pak lze přes SyncBox „tunelovat“ přes Ethernet pomocí protokolu TCP.

V tomto režimu lze generovat naplánované události a lze značkovat příchozí události. Funkce pro generování periodických průběhů a časových kódu jsou také dostupné. Lze tak použít zařízení pro synchronizaci dalších podružných hodin původně určených pouze pro časové kódy, jako např. IRIG-B.

Modul SyncBox implementuje SCPI protokol a disponuje sadou příkazů pro ovládání synchronizace, vstupů a výstupů. Zároveň disponuje příkazy pro nastavení externího sérového portu. Všechna data, která přijdou na port tak mohou být přenášena po TCP portu a obráceně všechna data z TCP port mohou být přenášena na sériový port. Tím je realizována funkce „tunelování“ komunikace.

Na obr. 8.19 je zobrazeno blokové schéma možného použití s měřicím přístrojem. Měřicí přístroj, který nedisponuje rozhraním Ethernet tak může být rozšířen o tuto funkcionalitu. Navíc jsou k dispozici funkce spouštění externí události, které mohou vyvolat trigger v měřicím přístroji a provést odměr. Zařízení lze použít i pro generování časové základny, která je synchronizovaná na hlavní hodiny. Funkce „tunelování“ lze použít pro přímou komunikaci s měřicím přístrojem.



Obr. 8.19: Blokové schéma použití zařízení SyncBox s běžným měřicím přístrojem

8.3 Měřicí ústředna s protokolem IEEE 1588

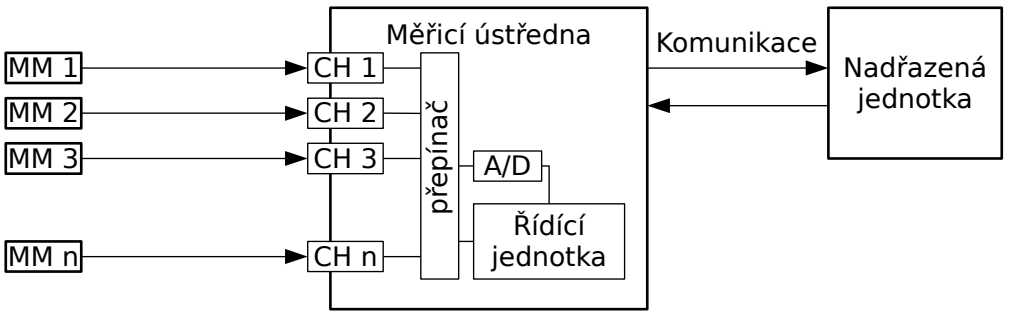
Pro sběr dat ze snímačů se používají systémy sběru dat, které se někdy označují jako měřicí ústředny. Jádrem měřicí ústředny je procesorová jednotka a sada vstupních kanálů, které tato jednotka obsluhuje. Měřicí ústředna může data předzpracovávat a dále je ukládat nebo zobrazovat. K měřicí ústředně pak může být připojena nadřazená jednotka, která tato data dále zpracovává.

V klasickém pojetí zpracovává měřicí ústředna přímo analogové vstupní hodnoty. Pro potřeby rozsáhlejších systému byla navržena a zkonstruována měřicí ústředna, které používá „chytré“ měřicí moduly a časovou synchronizaci pro zabezpečení návaznosti jednotlivých kanálů.

8.3.1 Klasická centralizovaná měřicí ústředna

Jádrem měřicí ústředny je jednotka s procesorem, která čte data z jednotlivých vstupních kanálů. Pojmem vstupní kanál je zde míněno připojení vstupního elektrického analogového signálu, např. ze senzoru, který se po příslušné úpravě úrovní digitalizuje. Měřicí ústředna může mít lokální zpracování výsledků měření z jednotlivých kanálů, zobrazení na zabudované zobrazovací jednotce, nebo záznam na zabudované paměťové médium.

Blokové schéma klasické měřicí ústředny je uvedeno na obr. 8.20. Měřené analogové signály jsou přiváděny z měřicích míst *MM 1* až *MM n* na vstupy jednotlivých kanálů *CH 1* až *CH n* měřicí ústředny. Připojení jednotlivých kanálů na vstup analogově/číslicového převodníku (A/D) je realizováno prostřednictvím analogového přepínače. V ústředně může být jeden, případně i více A/D převodníků, avšak počet převodníků bývá podstatně nižší než je počet vstupních kanálů. Výhodou centralizované měřicí ústředny je jednoduché zajištění synchronizace současného odběru vzorků ve vybraných kanálech s využitím A/D převodníku. Měřicí ústředna může být připojena k nadřazené jednotce prostřednictvím rozhraní Ethernet, USB, RS-232, RS-485 apod.



Obr. 8.20: Blokové schéma klasické měřicí ústředny

Uspořádání dle obr. 8.20 odpovídá centralizovanému měření, kdy se měřené signály v analogové formě musejí přivést až na vstupy jednotlivých kanálů měřicí ústředny. V případě rozhlehlého objektu, na němž by se měření provádělo, by to představovalo vedení signálů dlouhými vodiči. Takové uspořádání je náchylné na rušení a má řadu dalších nevýhod [60]. Ve většině případů je ale perioda vzorkování řádově delší než zpoždění přenosové cesty a není proto potřeba provádět žádné kompenzace nebo synchronizaci.

8.3.2 Distribuovaná měřicí ústředna

Distribuovaná měřicí ústředna přidává chytré měřicí moduly, které předávají svá data pomocí datové sítě. Tento mechanizmus eliminuje rušení, které by jinak vzniklo na dlouhých vedeních od měřicích míst, ale zavádí nové požadavky získávání časových značek k naměřeným vzorkům a tím potřebu synchronizace časových základen.

Problém připojení měřicích míst řeší měřicí ústředna s protokolem IEEE 1588 navržená v rámci projektu TAČR TA01010988 [60]. Jednotlivé měřicí kanály ústředny dle obr. 8.21 jsou koncipovány tak, že každý z nich obsahuje vstupní blok, blok A/D převodu, řídicí blok a blok rozhraní Ethernet. Jedná se tak o kompletní měřicí kanál, který může být připojen přímo na senzor nebo příslušné měřicí místo, a není třeba přenášet analogový elektrický měřený signál dlouhým kabelem. Pro co nejkompaktnější konstrukci je použito řešení s procesorem STM32F407, který zastává funkci řídicího bloku, funkci A/D i funkci řízení komunikace s rozhraním Ethernet. Pro spolupráci s rozhraním Ethernet je doplněn precizní ethernetovou PHY, která se také využívá v procesu synchronizace.

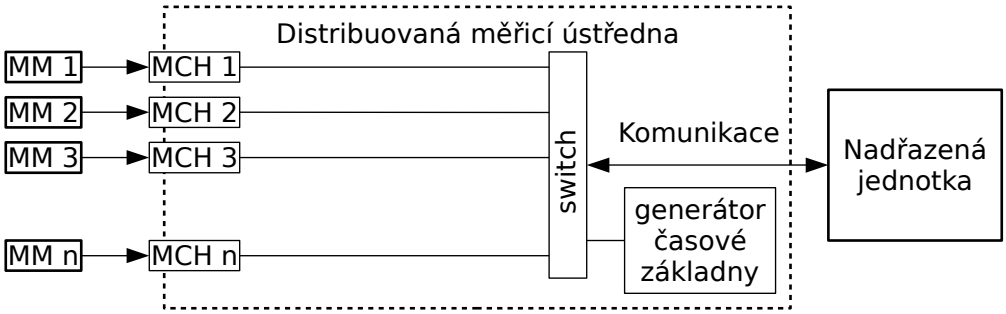
Obr. 8.21: Blokové schéma jednoho měřicího kanálu ústředny, MCH

Zjednodušené blokové schéma měřicí ústředny s protokolem IEEE 1588 je na obr. 8.22. Obsahuje měřicí kanály MCH, které jsou zapojeny přímo na měřicí místa MM. Ovládání měřicích kanálů a přenos naměřených dat se děje prostřednictvím rozhraní Ethernet. Vzájemné propojení jednotlivých kanálů i jejich připojení na řídicí jednotku je přes síťový přepínač. Ten umožňuje nejen přenos zpráv mezi kanálem a řídicí jednotkou, ale i komunikaci mezi jednotlivými kanály a generátorem časové základny.

Každý měřicí kanál obsahuje svůj lokální krystalový generátor časové základny. Při požadavku synchronního mnohokanálového měření však nelze tyto časové základny přímo využít, protože nemají dostatečnou stabilitu a nejsou na sebe vzájemně navázané. Synchronizace je řešena speciálním modulem v zapojení, který se stará o synchronizaci časových základen a využívá k tomu protokol IEEE 1588.

V měřicí ústředně mohou být jednotlivé měřicí kanály umístěny bezprostředně vedle sebe, avšak v případě potřeby je možné měřicí kanál umístit odděleně s připojením na vzdálenost odpovídající použitému komunikačnímu standardu. V případě Ethernetu je maximální komunikační vzdálenost jednoho spoje přibližně 100 m. Pokud by ani tato vzdálenost nestačila, je možné použít prodloužení s pomocí optické linky. Tato linka se

vytvoří vložením dvou shodných bloků obousměrných převodníků metalického Ethernetu na optický a zpět propojených příslušným optickým kabelem. Takovým způsobem lze dosáhnout zvětšení vzdálenosti měřicích modulů i na stovky metrů. Použití běžně dostupných převodníků ale sníží přesnost dosažitelné synchronizace řádově o mikrosekundy.



Obr. 8.22: Blokové uspořádání distribuované měřicí ústředny

8.3.3 Moduly měřicí ústředny

Jednotlivé moduly distribuované měřicí ústředny měří napětí na vstupech A/D převodníků a naměřená data posílají dále prostřednictvím síťového přepínače do nadřazené řídicí jednotky. Nadřazená jednotka řídí veškeré moduly, sbírá z nich naměřená data a ukládá je do přehledně organizovaného CSV souboru, který může být následně dále zpracován.

Jednotlivé moduly měřicí ústředny jsou v této konkrétní realizaci složeny z modulů SyncCore popsaných v kapitole 8.2.3. Tyto moduly jsou univerzální a lze je použít pro mnoho aplikací, kde je potřeba přesná synchronizace.

Pro jednoduchost nasazení měřicí ústředny se používá napájení měřicích kanálů po kabelu Ethernetu. V každém měřicím modulu byl navíc osazen příslušný modul Power over Ethernet (PoE), který zajišťuje napájení a současně i jeho galvanické oddělení. Díky tomu jsou jednotlivé moduly vůči sobě galvanicky odděleny. Toto ve výsledku znamená, že ke každému modulu stačí vést pouze jeden kabel, který přenáší jak data tak napájení pro modul.

Napájecí PoE modul byl zakoupen jako již hotový prvek osazený čipem AS1135 od společnosti Akros Silicon. Tento modul je plně kompatibilní se standardem PoE+ definovaným normou IEEE 202.3at-2009. Specializovaný integrovaný obvod je pro korektní fungování PoE+ nezbytný, protože výše zmíněný standard definuje sadu zpráv a příkazů, které si nejprve musí vyměnit síťový přepínač (zdroj PoE napájení) a řídicí elektronika napájecího modulu (spotřebič), než je zapnuto napájení pro dané zařízení. Pokud úspěšně proběhne tato komunikace, může dané zařízení odebírat až 25,5 W.

Aby bylo možné jednotlivé moduly napájet z rozhraní Ethernet, je třeba použít síťový přepínač, který dokáže poskytovat PoE pro jednotlivé zařízení k němu připojená. Lze použít například i běžný síťový přepínač TP-Link TL-SF1008P.

8.3.4 Synchronizace času v měřicí ústředně

Kvalita synchronizace jednotlivých kanálů závisí především na PDV, kterou do komunikace zanáší přenosová cesta. V případě obyčejného síťového přepínače se může zpoždění paketů měnit a může být různé v obou komunikačních směrech.

Při použití speciálních průmyslových síťových přepínačů s podporou protokolu IEEE 1588 je situace o mnoho lepší, protože dokáží tato zpoždění kompenzovat. Bohužel jsou ale taková zařízení velice drahá (30 až 40-ti násobná cena oproti obyčejným síťovým přepínačům). Vlastnosti obyčejných a průmyslových síťových přepínačů jsou popsány v kapitolách 5.3.3 a 5.3.4.

U obyčejných síťových přepínačů navíc hrozí i proměnné zpoždění paketů v závislosti na množství přenášených dat. Proto byl použit koncept časového multiplexu, popsaného v kapitole 7. Tímto způsobem je přenosový kanál rozdělen do dílčích částí, které se vzájemně neovlivňují.

8.3.5 Sběr a odeslání naměřených dat modulem

Každý z vnitřních A/D převodníků může vzorkovat vstupní signál s maximální frekvencí 2,4 MSa/s, nicméně použitý mikrokontrolér umožňuje taktéž zřetězit všechny tři dostupné převodníky tak, že měří na jednom pinu v prokládaném režimu. V tomto režimu lze dosáhnout maximální vzorkovací frekvence 7,2 MSa/s při maximálním rozlišení 12 bitů.

V procesoru není využit blok podpory protokolu IEEE 1588, ale je použita externí ethernetová PHY, která tento protokol také podporuje. Externí obvod PHY disponuje přesným řiditelným oscilátorem s malým jitterem, jehož odchylka od nominální frekvence je neustále kompenzována pomocí údajů z protokolu IEEE 1588. PHY je nakonfigurována tak, aby poskytovala výstup tohoto dolaďovaného oscilátoru přímo procesoru. Ten pak běží na frekvenci, která je přímo odvozená od hlavních hodin. Vzorkovací frekvence A/D převodníků je tedy také synchronizována na hlavní hodiny.

Z PHY je také využit blok univerzálních vstupů a výstupů s přesným časováním. Ty mohou být využity pro generování přerušení v mikrokontroléru. Lze tak vytvořit funkci naplánování odměru v jeden čas.

8.3.6 Řízení distribuovaného systému

Řídicím prvkem měřicí ústředny je SCPI server, který implementuje standardy IEEE 488.2 a SCPI-99. Ten poskytuje jednoduché a unifikované rozhraní pro ovládání distribuovaného systému prostřednictvím rozhraní Ethernet. SCPI server je implementován s pomocí knihovny SCPI Parser [58], která je popsána v kapitole 13.1.

Měřicí systém disponuje mimo jiné těmito příkazy pro ovládání:

`DIST:RUN ON/OFF`	spuštění/ukončení odměru
`DIST:RUN?`	dotaz na stav odměru
`DIST:TIME?`	dotaz na čas začátku měření
`DIST:SYNC`	vynucení obnovení synchronizace

Vzhledem k použití časového multiplexu ale nejsou data přenášena pomocí samotného SPCI protokolu, ale jsou vytvořeny speciální komunikační kanály, a zařízení sama odesílají data po startu měření v přiřazených časových slotech.

8.4 Implementace protokolu IEEE 1588 v OS Linux

Implementace protokolu IEEE 1588 spočívá ve vytvoření softwarové komponenty podle příslušného standardu. Existuje několik komerčních i otevřených implementací tohoto protokolu. Pro operační systém Linux existují dvě významné otevřené implementace a to PTPd [34] a linuxptp [61].

Existuje řada síťových karet, které podporují protokol IEEE 1588. Podpora některých karet je možná pouze pomocí proprietárních řešení, některé jsou ale přímo podporovány jádrem OS Linux. Hardwarovou podporu umí plně využít pouze projekt linuxptp.

Bez hardwarové podpory umožňuje Linux získávat časové značky s granularitou 1 ns. Nejmenší korekce časové základny je menší než 1 ppb. Operační systém při startu vyhodnotí dostupné časovače v počítači a vybere ten, který je principiálně nejstabilnější a nejpřesnější. Starší počítače disponuji registrem Time Stamp Counter (TSC), který se v prvních implementacích odvozoval přímo od frekvence procesoru. Pokud ale procesor umožňuje měnit frekvenci za běhu, pak se mění i frekvence aktualizace tohoto čítače. Moderní systémy jej již inkrementují z hodinového signálu, který se nemění při změně frekvence procesoru. Další možností je High Precision Event Timer (HPET), který je součástí nových počítačů.

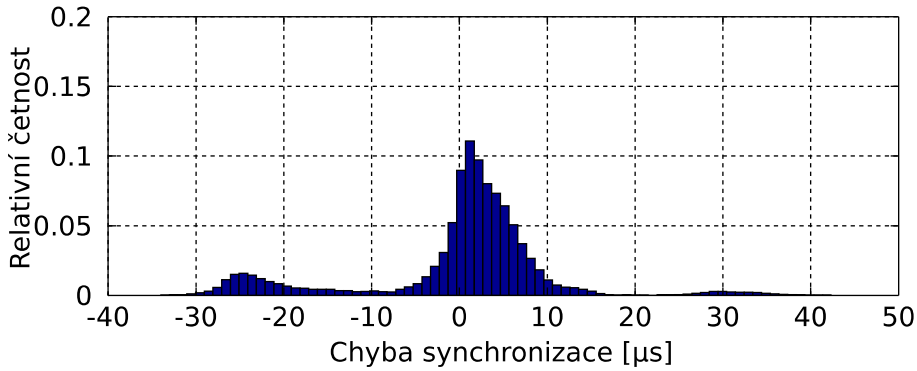
Hlavní výhodou těchto časovačů je jejich vysoká rozlišovací schopnost. Pokud časovač běží na frekvenci procesoru, která je větší než 1 GHz, pak je rozlišovací schopnost takového časovače lepší než 1 ns. Operační systém Linux umí této přesnosti využít.

Časové značky příchozích paketů jsou zjišťovány již v ovladači. Ve spojení vysoké přesnosti časovače a získávání časové značky již v ovladači lze dosáhnout lepší časové synchronizace než při jejich ukládání až v uživatelské aplikaci.

S hardwarovou podporou se časové značky získávají ještě na nižší úrovni, ale s časovou stupnicí síťové karty. Pokud je potřeba synchronizovat systémový čas, je třeba spustit další proces, který synchronizuje čas síťové karty na systémový čas. Tím se vysoká přesnost částečně degraduje.

Podpora získávání časových značek paketů je v Linuxu součástí síťového zásobníku a Linuxového jádra pod jednotným API. Pro přidání nového hardware tak stačí implementovat podporu tohoto API a všechen existující software s tím bude umět pracovat.

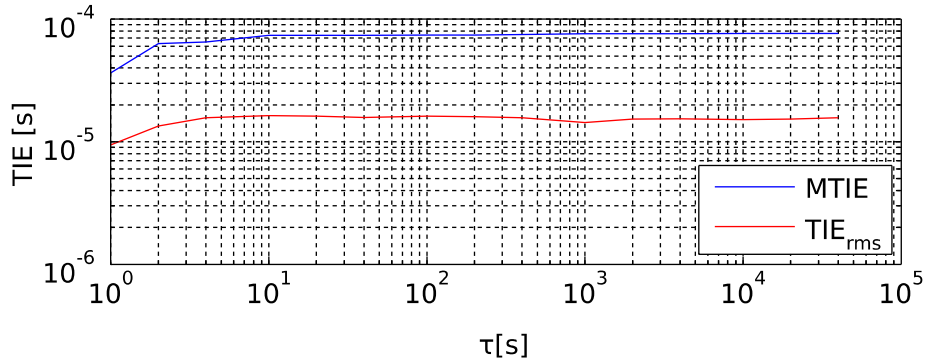
Na počítači nebyl k dispozici hardwarový výstup 1 PPS, takže naměřená data jsou jen odhad přesnosti synchronizace. Na obr. 8.23 je uveden histogram chyb synchronizace. Pokud by se měřilo navíc i PPS, byl by celý histogram posunutý, protože díky softwarovému zjišťování času odeslání a přijetí paketu nejsou tyto časové značky symetrické. Na obr. 8.24 jsou uvedeny metriky MTIE a TIE_{rms} vycházející z odhadu chyb synchronizace. U obyčejného operačního systému mohou být navíc tyto asymetrie závislé na zátěži systému.



Obr. 8.23: Histogram chyby sinchronizace v OS Linux

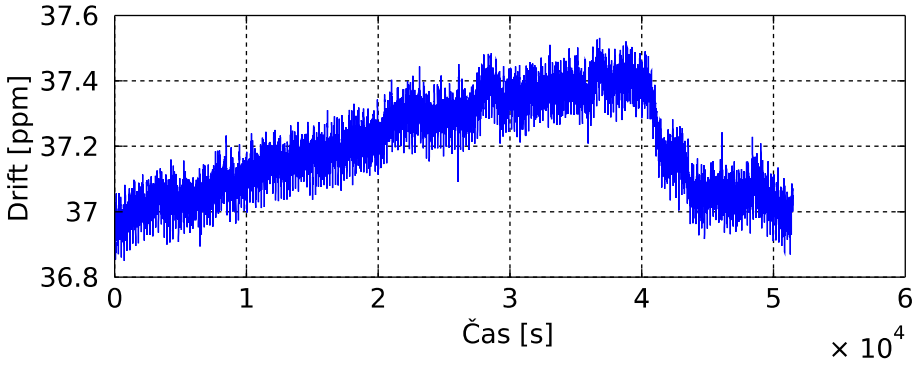
Operační systém Linux lze standardně považovat za Soft Real Time operační systém. Existuje sada úprav zvaná jako RT-Preempt Patch, díky které se Linux promění v Hard Real Time operační systém [62]. Na rozdíl od jiných RTOS není třeba provádět žádné změny programu, pouze se při spuštění zvolí, že má být použit RT plánovač. Všechny funkce operačního systému jsou implementovány stejně dobře jak při použití RT plánovače,

tak při použití běžného plánovače. Použití RTLinuxu je tedy přímočaré a jednoduché, například na rozdíl od RTX. V tomto režimu by asymetrie značkování paketů byla konstantní a bylo by možné ji jednorázově kompenzovat.



Obr. 8.24: Metriky TIE v Linuxu

Během měření byl sledován drift hodin. Průběh driftu je uveden na obr. 8.25. Změna frekvence hodin byla dána především změnou teploty v místnosti, protože místnost nebyla nijak klimatizována a byl použit běžný počítač, který nemá nijak tepelně kompenzovaný oscilátor.



Obr. 8.25: Naměřený drift hodin během synchronizace v Linuxu

8.5 Implementace protokolu IEEE 1588 v OS Microsoft Windows

V operačním systému Windows existují dva základní přístupy. První přístup využívá specializovaného hardware v kombinaci se specializovaným software. Druhý přístup využívá čistě softwarového řešení.

Specializovaný software a hardware mají ve Windows nevýhody v tom, že každý výrobce je řeší po svém a není tedy jednoduše možné nasadit jiný hardware nebo software. Pokud například vznikne nová verze protokolu

a výrobce hardware to nebude reflektovat, nelze tuto verzi vůbec provozovat.

Specializovanou implementací je například síťová karta NI PCI-1588 [47]. Tato karta disponuje možností hardwarové podpory protokolu a dále generování nebo značkování externích událostí. Instalací verzí ovladače se volí i verze PTP protokolu a není tedy možné provozovat obě verze zároveň.

Čistě softwarové řešení lze také implementovat, ale v OS Windows nejsou již tak dobré podmínky pro přesné časování. Časové značky lze získávat s granularitou 100 ns, bohužel je to v některých případech jen zdánlivá přesnost, protože čas je odvozen od časovače, který se inkrementuje po 1/64 s, tedy asi 15 ms. Nejmenší korekce časové základny je s krokem 6,4 ppm.

Navržené řešení implementace protokolu IEEE 1588 bylo založeno na upravené verzi PTPd. Byla nahrazena a upravena volání Linuxového API za volání Win32 API.

Pro získání přesného času v systému Windows slouží funkce `GetSystemTimeAsFileTime`, která má granularitu 100 ns. Na Windows XP má bohužel reálnou rozlišovací schopnost 15 ms. Od Windows 8 Microsoft konečně zareagoval na špatnou implementaci přesné časové stupnice. Od této verze je k dispozici funkce `GetSystemTimePreciseAsFileTime`, která se snaží získat časovou značku s lepší přesností než 1 µs.

Kalibrace časové základny je možná pomocí volání API funkce `SetSystemTimeAdjustment`. Tato funkce umožňuje nastavit inkrementaci časové základny po jiném než základním kroku. Inkrementaci časové základny lze měnit s krokem 100 ns. Hodnota se přičítá k systémovému času každý tik hodin, kterých nastává 64 za sekundu. Zvýšením hodnoty inkrementu o jeden krok tak sníží frekvenci hodin o 6,4 ppm. Tento krok je příliš hrubý pro přesnou časovou synchronizaci a algoritmus v PTPd na něj není optimalizovaný.

Získávání časových značek je možné pouze na úrovni aplikace. Bylo provedeno i několik pokusů se získáváním časových značek na úrovni ovladače pomocí WinPCAP, ale bohužel WinPCAP používá časové značky odvozené od své volně běžící časové stupnice, která nijak nekoresponduje se systémovým časem a nelze ji ani nijak ovlivnit.

Naměřená data jsou pouze odhadem dosažené přesnosti synchronizace, protože počítač nedisponoval hardwarovým výstupem pro generování PPS. Výsledná synchronizace tedy bude posunutá o asymetrii značkování příchozích a odchozích paketů. Tato asymetrie se může se zátěží měnit. Vzhledem k tomu, že Windows neumožňují fungovat jako RTOS, tak tu není ani prostor pro kompenzaci této asymetrie.

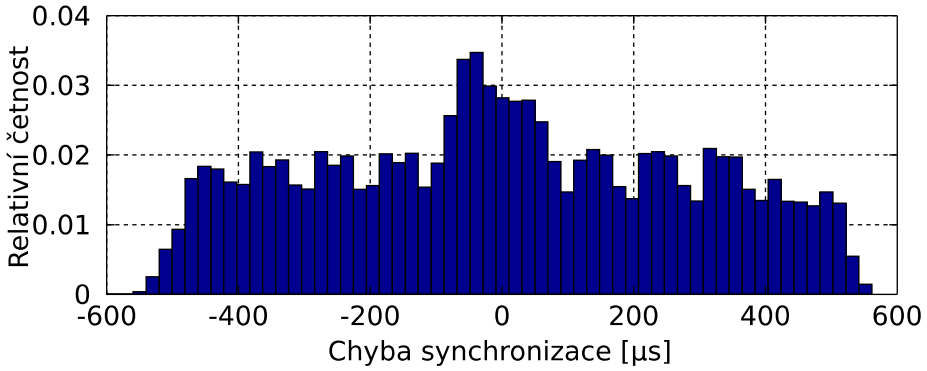
Na obr. 8.26 je zobrazen histogram chyb synchronizace. Z naměřených dat je zřejmé, že není možné dosáhnout přesnosti lepší než 1 ms. Vzhledem

ke špatné podpoře přesné časové stupnice v systému je ale výsledek uspokojující.

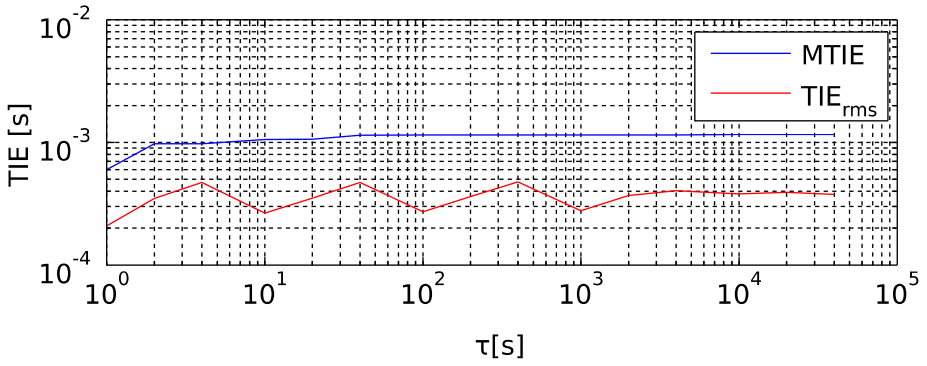
Na obr. 8.27 je zobrazen graf MTIE a TIE_{rms}. MTIE pro dlouhé časy vychází lepší než 1,2 ms. TIE_{rms} vychází zhruba 0,5 ms podle zvoleného intervalu τ.

Z naměřeného driftu (viz obr. 8.28) je patrná omezená možnost korekce. Regulační algoritmus měnil drift, ale ten se ve skutečnosti měnil pouze v krocích 3 ppm. Tato nemožnost přesné korekce frekvence hodin se pravděpodobně projevuje i v průběhu TIE_{rms}.

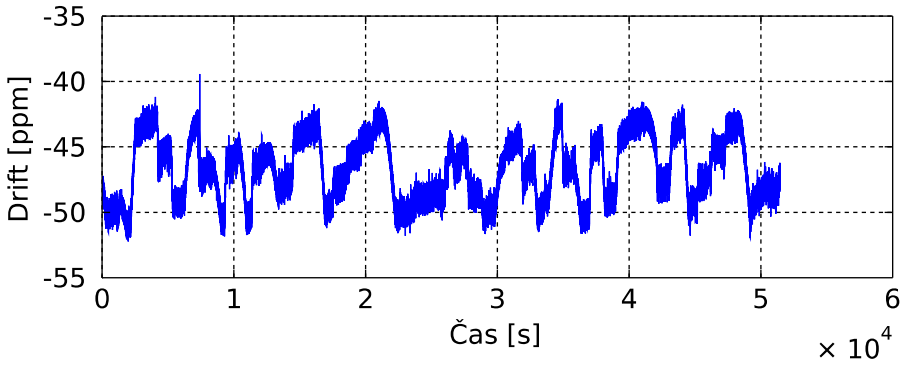
Na přesnost synchronizace měla největší vliv nemožnost plynule korigovat drift hodin. Algoritmus v PTPd sice nastavoval korekci v jednotkách ppb, ale vlivem zaokrouhlení na násobky 6,4 ppm časová stupnice nereagovala na změnu a zachovala si původní frekvenci. Teprve až byla provedena korekce větší než 6,4 ppm, byla tato hodnota reálně nastavena pomocí změny inkrementu systémových hodin.



Obr. 8.26: Histogram chyby synchronizace ve Windows



Obr. 8.27: Metriky TIE ve Windows



Obr. 8.28: Naměřený drift během synchronizace ve Windows

Popisované výsledky byly měřeny pouze v režimu podružných hodin. Pokud je počítač s OS Windows nastaven jako volně běžící hlavní hodiny, odpadne hlavní problém v podobě nemožnosti nastavovat plynule frekvenci časové základny. V tomto režimu se projeví pouze nepřesné získávání časových značek a jejich malá rozlišovací schopnost. To lze dlouhodobým průměrováním korigovat, a proto může počítač s Windows pracovat mnohem lépe jako hlavní hodiny než jako podružné hodiny.

8.6 Implementace protokolu IEEE 1588 v OS IntervalZero RTX

Operační systém RTX není úplně plnohodnotný systém, ale jde o Hard Real-time rozšíření systému Windows. V tomto systému běží úlohy na úrovni jádra s definovaným maximálním zpožděním reakce na vnější událost. Systému RTX lze přidělit procesorová jádra, která bude moci výhradně využívat. Dále je možné přidělit jiné hardwarové prostředky (např. PCI karty) pouze pro RTX. V práci byla použita verze IntervalZero RTX 2011. Systém byl vybrán partnerskou firmou v rámci řešení projektu TAČR TA01010988 [63].

8.6.1 Vlastnosti operačního systému RTX 2011

Jádro i uživatelské aplikace běží na systémové úrovni Ring 0. Paměť je sdílená pro celý systém RTX, a tak libovolný proces může přistupovat k paměti jiných procesů v RTX.

V systému lze využívat tři druhy aplikací. Standardní aplikace pro Windows, aplikace, které jsou propojené na RTX, ale nejsou Real-time, a čistě Real-time aplikace, které nemají vazbu na Windows.

Systém implementuje vlastní RTX API, které se snaží být z části kompatibilní s Win32 API. Z čistě Real-time aplikací lze volat vybrané funkce systému Windows, je ale třeba dbát na to, že jejich volání není

deterministické. Lze tak používat například alokace paměti nebo přístup k souborům.

Pro komunikaci s okolními procesy jak RTX, tak Windows, lze využít synchronizační objekty jako mutexy a semafory, nebo sdílenou paměť. Systém neobsahuje frontu, tu je nutné implementovat vlastními silami s pomocí sdílené paměti a zámků.

Systém obsahuje obsluhu pádu Windows zvané STOP (známá modrá obrazovka) a může dále fungovat. V takové situaci pak může bezpečně zastavit např. řízený proces a vyvolat restartování celého systému. Po pádu Windows ale nejsou k dispozici některé funkce, jako dynamické alokace paměti, a tak je nutné veškeré volání těchto funkcí testovat na podmínku pádu Windows.

Pokud nastane kritická chyba v systému RTX, je vyvolána podobná obrazovka jako Windows STOP, ale je zelená a ukazuje podmínky, za kterých k chybě v RTX došlo.

8.6.2 Implementace časové stupnice

Systém neumožňuje jemně nastavovat časovou stupnici, a tak bylo přistoupeno k vytvoření vlastní časové stupnice odvozené od funkce, `QueryPerformanceCounter`, QPC. Granularita této stupnice je pak závislá na frekvenci procesoru. Pro procesor s taktovacím kmitočtem větším než 1 GHz pak granularita vycházela lepší než 1 ns. Výsledný čas se počítal numericky až ve chvíli, kdy o něj bylo požádáno, a možnost korekce časové stupnice byla lepší než 1 ppb.

Získání hodnoty QPC trvalo na testovaném systému zhruba 155 ns. Převod na aktuální čas trval dalších 100 ns, takže přečtení jedné časové značky trvalo v průměru 250 ns.

Nad časovou stupnicí byl implementován jednoduchý plánovač událostí. Tento plánovač sloužil k přesnému naplánování události a k tomu využíval mechanizmu periodické kontroly časové stupnice střídané se spaním. Pokud se začal blížit okamžik spuštění události, plánovač se přepnul do režimu nepřetržitého vyčítání času, a tím zvýšil svoji přesnost.

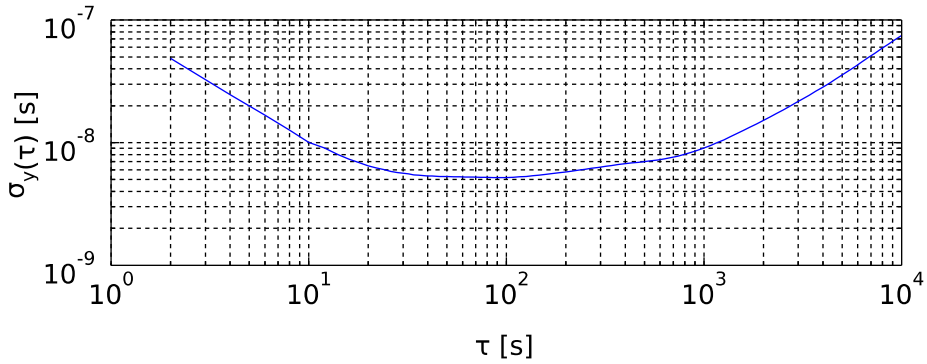
V systému byla připojena přídavná karta se vstupy a výstupy NI PCI-6503. Tuto kartu lze jednoduše ovládat zápisem do paměťových registrů, a tím nastavovat výstupy.

Aby bylo možné prokázat schopnost přesného plánování událostí, byl systém spuštěn bez synchronizace času a v plánovači bylo nastaveno generování výstupního pulzu každou sekundu. Tento sekundový pulz byl porovnáván s rubidiovým standardem PRS10 [64], který deklaruje krátkodobou i dlouhodobou stabilitu lepší než 10^{-10} s.

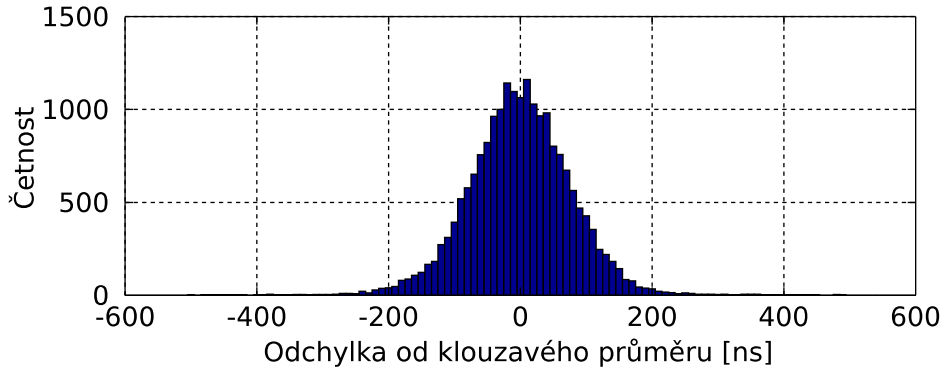
Z porovnání se standardem byla nejprve vypočtena Allanova odchylka (obr. 8.29) a díky ní byl následně vypočten odhad krátkodobé přesnosti

generování PPS pulzů. Ten byl vypočten jako rozdíl aktuálně naměřené odchylky od klouzavého průměru. Velikost okna klouzavého průměru byla odhadnuta z Allanovy odchylky jako τ minimální hodnoty.

Na obr. 8.30 je vyobrazen histogram odchylek od klouzavého průměru. Z tohoto histogramu lze odhadnout, jak dobře lze v RTX generovat časovou stupnici a jak lze generovat výstupní události. Většina hodnot spadá do intervalu ±250 ns, což je zároveň délka dvou volání funkce na zjištění aktuálního času.



Obr. 8.29: Allanova odchylka PPS výstupu RTX



Obr. 8.30: Histogram odchylek PPS od klouzavého průměru

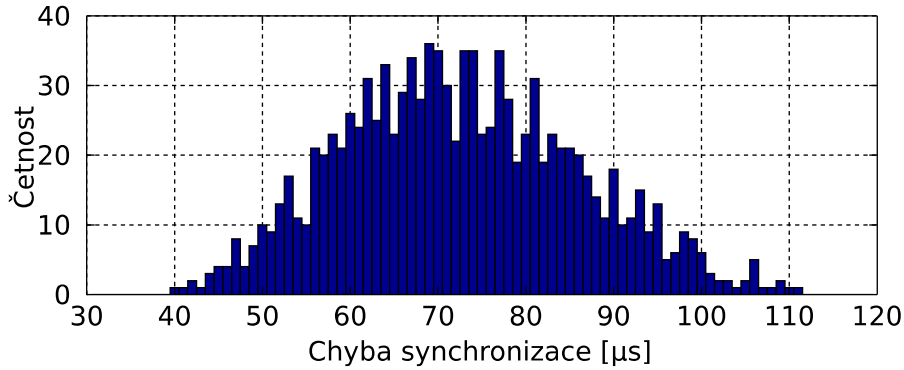
8.6.3 Implementace PTP protokolu

Implementace PTP protokolu využívá jako základ upravenou verzi PTPd. Modifikace vychází z verze pro Windows, ale vzhledem k velkým nekompatibilitám v API funkcích a v odlišné časové základně se tyto verze liší výrazně. V této implementaci byla použita vlastní verze časové základny, která disponuje možností kalibrace.

Při testování byly použity hlavní hodiny s hardwarovou podporou protokolu, které byly synchronizovány na GPS pro větší stabilitu, popsané v kapitole 8.1.5. Hlavní hodiny i podružné hodiny v podobě počítače s RTX

generovaly PPS signál, který byl porovnáván na čítači. Výsledek je tedy skutečný rozdíl časových základen obou zařízení.

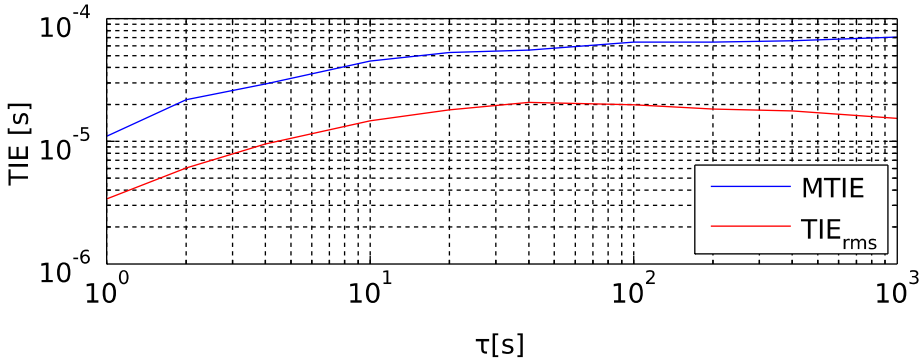
Histogram naměřených odchylek je na obr. 8.31. Z histogramu je vidět, že střední hodnota není v nule, takže se projevila asymetrie značkování příchozích a odchozích paketů. Vzhledem k povaze operačního systému ji lze bez potíží kompenzovat jednorázovou konstantou.



Obr. 8.31: Histogram chyby synchronizace RTX

Na obr. 8.32 jsou znázorněny metriky MTIE a TIE_{rms}. Lze například provést porovnání s implementací v Linuxu, kde v běžném operačním systému bylo dosaženo MTIE pro čas 1000 s hodnoty 76 µs a v RTX hodnoty 70 µs. TIE_{rms} pro čas 1000 s naopak vycházela v Linuxu 16 µs a v RTX 48 µs. Lze tedy říct, že v porovnání maximální chyby synchronizace si vedou oba systémy podobně, ale v porovnání efektivní chyby synchronizace je na tom podstatně lépe operační systém Linux.

Bohužel se nepodařilo naměřit delší datové vzorky, protože systém RTX nebyl stabilní při použití aplikace pro synchronizaci času. Aplikace byla velice podobná verzi ve Windows i v Linuxu, kde funguje bez problémů. Díky špatné celkové koncepci RTX se zde projevovaly nekompatibility, které vždy po čase skončily pádem systému.



Obr. 8.32: Metriky TIE synchronizace RTX

8.6.4 Zhodnocení implementace v RTX

V dokumentaci k RTX je jako velký benefit považováno to, že implementuje Win32 API, takže programátor zvyklý programovat v tomto API nemá problém s přechodem. Bohužel je to jen marketingový trik. Běžný programátor pro Windows neprogramuje na tak nízké úrovni, jako je Win32 API, a použije raději vyšší programovací jazyk jako C# a patřičné knihovny, které jsou odlišné od Win32 API.

Další problém je v tom, že všechny funkce jsou implementovány s prefixem Rt, takže pokud chce programátor použít existující kód nebo knihovny pro Win32 API, musí je celé projít a přepsat. Není tedy možné sdílet kód s částí aplikace, která nemá běžet pod RTX.

Zásadní nedostatky jsou na úrovni celé koncepce systému RTX 2011 a implementace API. Mnohá volání vypadají podobně, ale neimplementují celou funkcionalitu, nebo ji implementují chybně. RTX nekontroluje vstupní parametry funkcí, takže drobná chyba způsobí pád aplikace. Vzhledem k tomu, že všechny aplikace běží v jednom paměťovém prostoru a na úrovni jádra, nelze ani oddělit chybu v jedné aplikaci. Drobná chyba tak může způsobit pád aplikace, která díky těmto koncepčním problémům způsobí pád celého subsystému RTX a ten způsobí pád celého systému Windows. Toto chování se bohužel potvrdilo a při implementaci a ladění aplikace pro synchronizaci nastalo mnoho pádů celého systému a bylo nutné neustále restartovat počítač. Díky chybám v samotném jádru operačního systému nebylo ani možné ladit aplikace a odhalit skutečný zdroj problémů a pádů.

Systém není vhodný pro synchronizaci času ani pro přesné časování. Nedisponuje totiž funkcí k získání informace o čase s lepší granularitou, než 10 ms. Vzhledem k časování má tedy horší vlastnosti, než samotný systém Windows. Pro potřeby udržení přesné časové stupnice je potřeba ji implementovat vlastními silami a nepoužívat systémový čas.

I přesto, že se jednalo o Hard Real Time operační systém, tak implementace přesné časové synchronizace dosahovala podobných parametrů, jako synchronizace v běžném operačním systému, jako je Linux.

V porovnání se svobodnou implementací RTOS postavené na Linuxu se všechny tyto vlastnosti podílejí na celkovém špatném hodnocení celého systému RTX.

Díky špatnému návrhu konceptu RTX 2011 a chybám v jeho implementaci nebylo možné aplikaci pro synchronizaci času úplně vyladit, aby systém běžel bez problémů. Vzhledem k tomu, že použití tohoto systému nemá žádné benefity oproti použití jiného Hard Real Time operačního systému, např. RTLinuxu, bylo upuštěno od dalšího testování a ladění implementace.

8.7 Zhodnocení výsledků

Protokol PTP podle standardu IEEE 1588 byl implementován v řadě zařízení a v různých operačních systémech, aby byly ověřeny vlastnosti těchto zařízení a systémů vzhledem k přesné synchronizaci času.

Nejprve byla vytvořena implementace na hardwarové platformě s procesorem STM32, který obsahuje podporu pro tento protokol přímo v jednom ze svých funkčních bloků. Byl testován vliv hardwarové podpory protokolu na přesnost synchronizace a výsledky byly zpracovány v kapitole 8.1. Během implementace byla odhalena hardwarová chyba v procesoru STM32, která způsobuje nestabilitu časové základny při použití krystalu s interním oscilátorem.

Výsledky implementace byly použity k návrhu a konstrukci funkčních vzorků. Jedním z těchto funkčních vzorků bylo zařízení pro synchronizaci s protokolem IEEE 1588, které může sloužit jako hlavní i jako podružné hodiny. Zařízení je popsáno v kapitole 8.2. Dalším funkčním vzorkem byla měřicí ústředna s protokolem IEEE 1588 popsaná v kapitole 8.3. Byly navrženy a implementovány chytré senzory, které komunikují s využitím sítě Ethernet a které využívají protokol IEEE 1588 ke své časové synchronizaci.

Přesná časová synchronizace může být provozována i na běžných počítačích, kde byla implementována do různých operačních systémů. Pro OS Linux existují otevřené implementace, ze kterých byl vybrán projekt PTPd. V kapitole 8.4 jsou popsány vlastnosti systému Linux z pohledu přesné synchronizace. V kapitole 8.5 je popsána implementace protokolu v OS Windows. V kapitole 8.6 je popsána implementace v OS RTX, který je nadstavbou OS Windows a přidává do něj Real-Time vlastnosti.

Operační systém Windows obsahuje API pro dolaďování systémových hodin, ale toto API neumožňuje přesnou časovou synchronizaci, a proto byla dosažena přesnost synchronizace pouze 1 ms. Operační systém RTX neobsahuje žádné API pro dolaďování systémových hodin, a tak byla implementována vlastní časová stupnice, která to umožňuje. Pomocí této stupnice bylo dosaženo přesnosti synchronizace lepší než 100 µs. Na systému Linux je situace mnohem lepší, protože obsahuje API pro přesné dolaďování systémových hodin. Na tomto systému byla dosažena schodná přesnost jako na systému RTX.

V celkovém porovnání se tedy systém RTX a Linux chovají z pohledu přesnosti synchronizace velice podobně a systém Windows je o řád horší. Překvapením bylo, že v Real-Time systému RTX není možné dosáhnout lepší přesnosti synchronizace než v běžném operačním systému, jako je Linux. I přes to, že je systém RTX na trhu již několik let, obsahoval řadu chyb, které zamezovaly stabilnímu provozu aplikace pro synchronizaci.

9 Závěr

Práce se zabývá synchronizací času v heterogenním systému a je cílena na instalace bez požadavků na speciální hardware. V takových případech je ale zapotřebí zjistit parametry dané sítě a odhadnout dosažitelnou přesnost. Hlavním cílem této práce bylo otestování vlastností různých způsobů synchronizace a měření vlastností přenosové cesty, především jejího zpoždění.

9.1 Porovnání výsledků práce se stanovenými cíli

a) Návrh a realizace metody měření zpoždění přenosové cesty.

V rámci řešení byl navržen měřicí přístroj a metoda pro měření zpoždění paketů v síti s ohledem na přesnou časovou synchronizaci (viz kapitola 5). Měření využívá protokolu IEEE 1588, nevyužívá však tento protokol k synchronizaci, ale pouze k přesnému měření zpoždění. Zařízení bylo kalibrováno a po té ověřeno sadou testů na známých zpožděních. Následně bylo provedeno několik vzorových měření síťových prvků a celých sítí. Zařízení pro lokální měření dosahuje rozlišovací schopnosti 8 ns. V případě zařízení pro měření rozsáhlých sítí, kde jednotlivé části měřicího zařízení byly synchronizovány pomocí GPS, je dosaženo přesnosti lepší než 1 µs.

b) *Návrh a implementace způsobu ověření správnosti protokolu.*

V době řešení práce neexistoval automatizovaný způsob ověření implementace. Byl proto vytvořen nástroj, který může sledovat časovou synchronizaci v síti a vyhodnocovat odchylky od standardu (viz kapitola 6). Nástroj lze použít i off-line ze záznamu paketů, a tak zpětně vyhodnotit případné problémy. Nástroj byl otestován ve známých sítích a byla ověřena jeho funkčnost. Byla díky němu například odhalena chyba nedodržení synchronizačních intervalů u komerčního přístroje LANTIME M600.

c) *Návrh a realizace metody pro eliminaci změn zpoždění přenosové cesty při různých zátěžích.*

Navržená metoda pro eliminaci změn zpoždění při zatížení sítě je popsána v kapitole 7. Metoda nevyžaduje speciální síťové prvky ani úpravu protokolového zásobníku. Je ale nutné, aby tuto metodu využívaly všechny komunikující strany. Metoda využívá rozdělení komunikace do časových oken a přiřazení časových oken jak jednotlivým zařízením tak vlastní synchronizaci času. Navržená metoda pak byla použita ve funkčním vzorku popsaném v kapitole 8.3.

d) *Implementace protokolu IEEE 1588 do různých systémů, ověření chování a dosažitelné přesnosti.*

V kapitole 8 je popsáno několik implementací protokolu jak na úrovni mikrokontroléru (viz kapitola 8.1), tak v rámci operačních systémů na běžném PC.

U mikrokontroléru byla provedena implementace na STM32 s využitím hardwarové podpory při získávání časových značek. Tato implementace je popsána v kapitole 8.1. Bylo dosaženo přesnosti synchronizace lepší než 120 ns při použití externího oscilátoru. Dále byla ověřena funkce ethernetové fyzické vrstvy DP83630, která také obsahuje hardwarovou podporu získávání časových značek. Na rozdíl od implementace v mikrokontroléru ale umožňuje lepší využití synchronizované časové základny. V rámci praktického ověření implementace byla vyvinuta a realizována řada funkčních vzorků (viz kapitoly 8.2 a 8.3).

Byla ověřena implementace v OS Linux (viz kapitola 8.4) s tím, že bylo dosaženo přesnosti synchronizace lepší než 100 µs. V případě implementace v OS Windows (viz kapitola 8.5) byla sice dosažena přesnost synchronizace lepší než 1,2 ms, což je ale v řadě případů nevyhovující. Shodné přesnosti 100 µs jako v případě OS Linux bylo dosaženo při implementaci v RTOS RTX (viz kapitola 8.6), kde však nebylo možné dosáhnout stabilního chodu aplikace kvůli chybám v OS RTX.

Z uvedeného plyne, že vytyčené cíle práce byly splněny. Výsledky dosažené v této práci byly publikovány (viz kapitola 11).

9.2 Originální výsledky dosažené v této práci

- Návrh a realizace měřicího zařízení pro stanovení zpoždění přenosové cesty z pohledu přesné časové synchronizace (viz kapitola 5)
- Návrh a realizace analyzátoru komunikace, který vyhodnocuje správnost implementace protokolu dle IEEE 1588 (viz kapitola 6)
- Implementace protokolu v systému RTX (viz kapitola 8.6)
- Praktické vyjádření odhadu dosažitelné přesnosti pomocí efektivního počtu bitů (viz příloha 13.4)
- Softwarová knihovna pro zpracování příkazů dle standardu SCPI na straně měřicího přístroje (viz příloha 13.1), která je používaná vědeckými pracovišti v různých částech světa.

9.3 Doporučení pro další rozvoj a realizace v praxi

Měření zpoždění paketů na přenosové cestě je momentálně implementováno pouze pro 100Mbps Ethernet. Tímto způsobem je funkcionalita omezena v sítích implementujících i 1Gbps Ethernet, kde dochází v aktivním sítovém prvku k dalším prodlevám způsobeným změnou rychlosti na daném portu. V další verzi měřicího zařízení by tedy bylo dobré implementovat podporu i pro vyšší komunikační rychlosti.

Analyzátor protokolu by bylo vhodné rozšířit o podporu sond v síti, takže by bylo možné lépe sledovat provoz i jiných druhů provozu protokolu PTP než jen E2E multicast. Takto by bylo možné sledovat provoz unicastové komunikace nebo P2P komunikace.

V operačním systému Microsoft Windows jsou od verze 8 k dispozici nové funkce pro přesné měření času, zejména pak funkce `GetSystemTimePreciseAsFileTime`. Tyto nové funkce se v aktuální implementaci nepoužívají a bylo by tedy vhodné je použít v nové verzi a otestovat jejich vliv na výslednou přesnost synchronizace.

10 Literatura

[1] Jeremy Elson, Lewis Girod a Deborah Estrin, „Time Synchronization for Wireless Sensor Networks“, in *Proceedings of the 15th International Parallel & Distributed Processing Symposium*, roč. 2001.

[2] L. Lamport, „Time, Clocks, and the Ordering of Events in a Distributed System“, *Commun. ACM*, roč. 21, č. 7, s. 558–565, čvc. 1978.

[3] „Timing characteristics of synchronous Ethernet equipment slave clock“, *ITU-T G.8262/Y.1362*, led. 2015.

[4] Telecommunications and Timing Group, „IRIG serial time code formats“, *IRIG standard 200-04*, zář. 2004.

[5] „Time and Standard Frequency Station DCF77 (Germany)“. [Online]. Dostupné z: https://www.eecis.udel.edu/~mills/ntp/dcf77.html. [Viděno: 15-čer-2016].

[6] V. Vigner, J. Roztočil a B. Čemusová, „Evaluation of timing GPS receivers for industrial applications“, in *Proceedings of the 12th IMEKO TC10 Workshop on Technical Diagnostics*, Universita di Firenze, 2014, roč. 2013, s. 177–182.

[7] J. Elson, L. Girod a D. Estrin, „Fine-grained Network Time Synchronization Using Reference Broadcasts“, *SIGOPS Oper. Syst. Rev.*, roč. 36, č. SI, s. 147–163, pro. 2002.

[8] S. Ganeriwal, R. Kumar a M. B. Srivastava, „Timing-sync Protocol for Sensor Networks“, in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, New York, NY, USA, 2003, s. 138–149.

[9] J. E. Elson, „Time Synchronization in Wireless Sensor Networks“, University of California, Los Angeles, 2003.

[10] F. Cristian, „Probabilistic clock synchronization“, *Distrib Comput*, roč. 3, č. 3, s. 146–158, zář. 1989.

[11] K. A. Marzullo, „Maintaining the Time in a Distributed System: An Example of a Loosely-coupled Distributed Service (Synchronization, Fault-tolerance, Debugging)“, Stanford University, Stanford, CA, USA, 1984.

[12] D. L. Mills, „Adaptive hybrid clock discipline algorithm for the network time protocol“, *IEEE/ACM Transactions on Networking*, roč. 6, č. 5, s. 505–514, říj. 1998.

[13] D. L. Mills, „Internet time synchronization: the network time protocol“, *IEEE Transactions on Communications*, roč. 39, č. 10, s. 1482–1493, říj. 1991.

[14] „IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems“, *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, s. 1–269, čvc. 2008.

[15] „IEEE Standard for Local and Metropolitan Area Networks – Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks“, *IEEE Std 802.1AS-2011*, s. 1–292, bře. 2011.
[16] Audio Engineering Society, „AES recommended practice for digital audio engineering – Synchronization of digital audio equipment in studio operations. (Revision of AES11-2003)“, *AES11-2009*, led. 2015.
[17] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt a G. Gaderer, „White rabbit: Sub-nanosecond timing distribution over ethernet“, in *Control and Communication 2009 International Symposium on Precision Clock Synchronization for Measurement*, 2009, s. 1–5.
[18] A. Ademaj a H. Kopetz, „Time-Triggered Ethernet and IEEE 1588 Clock Synchronization“, in *Control and Communication 2007 IEEE International Symposium on Precision Clock Synchronization for Measurement*, 2007, s. 41–43.
[19] „Definitions and terminology for synchronization networks“, *ITU-T G.810*, srp. 1996.
[20] S. Bregni, „Measurement of maximum time interval error for telecommunications clock stability characterization“, *IEEE Transactions on Instrumentation and Measurement*, roč. 45, č. 5, s. 900–906, říj. 1996.
[21] W. J. Riley, *Handbook of Frequency Stability Analysis*. National Institue of Standards and Technology, 2008.
[22] D. W. Allan a H. Hellwig, „Time deviation and time prediction error for clock specification, characterization, and application“, s. 29–36, 1978.
[23] M. Maróti, B. Kusy, G. Simon a Á. Lédeczi, „The Flooding Time Synchronization Protocol“, in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, New York, NY, USA, 2004, s. 39–49.
[24] S. Bregni a S. Maccabruni, „Fast computation of maximum time interval error by binary decomposition“, *IEEE Transactions on Instrumentation and Measurement*, roč. 49, č. 6, s. 1240–1244, pro. 2000.
[25] C. Demichelis a P. Chimento, „IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)“, *RFC3393*, lis. 2002.
[26] „Internet protocol data communication service – IP packet transfer and availability performance parameters“, *ITU-T Y.1540*, bře. 2011.
[27] A. Morton a B. Claise, „Packet Delay Variation Applicability Statement“, *RFC5481*, bře. 2009.
[28] „Network performance objectives for IP-based services“, *ITU-T Y.1541*, kvě. 2012.
[29] D. A. Howe, D. U. Allan a J. A. Barnes, „Properties of Signal Sources and Measurement Methods“, in *Thirty Fifth Annual Frequency Control Symposium. 1981*, 1981, s. 669–716.

[30] „DP83630 Precision PHYTER™ – IEEE 1588 Precision Time Protocol Transceiver“. Texas Instruments, 2013.
[31] J. Breuer, V. Vigner a J. Roztočil, „Precise packet delay measurement in an Ethernet network“, *Measurement*, roč. 54, č. 54, s. 215–221, srp. 2014.
[32] STMicroelectronics, „STM32F205xx, STM32F207xx, STM32F215xx and STM32F217xx advanced ARM-based 32-bit MCUs“, *RM003 Reference manual*, úno. 2015.
[33] „FreeRTOS – Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions“. [Online]. Dostupné z: http://www.freertos.org/. [Viděno: 15-čer-2016].
[34] „PTPd“, *GitHub*. [Online]. Dostupné z: https://github.com/ptpd/ptpd. [Viděno: 15-čer-2016].
[35] V. Vigner a J. Breuer, „Sleepy Cat Kit“, 2011. [Online]. Dostupné z: http://pck338-242.feld.cvut.cz/sckit/. [Viděno: 15-čer-2016].
[36] MTI-MILLIREN Technologies, Inc., „MTI 210 series OCXO“, bře. 2015.
[37] „Calnex – Paragon-X“. [Online]. Dostupné z: https://www.calnexsol.com/en/solutions/paragon-x. [Viděno: 15-čer-2016].
[38] „Timing requirements of slave clocks suitable for use as node clocks in synchronization networks“, *ITU-T G.812*, 2014.
[39] MEINBERG RADIO CLOCKS GmbH & Co. KG, „MBGPROTOSIM - NTP/PTP Simulation Software“. [Online]. Dostupné z: https://www.meinbergglobal.com/english/archive/mbgprotosim.htm. [Viděno: 15-čer-2016].
[40] „International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication“. [Online]. Dostupné z: http://www.ispcs.org/. [Viděno: 22-čer-2016].
[41] J. Kaisrlík, „Monitor protokolu pro přesnou časovou synchronizaci“, České vysoké učení technické v Praze, 2013.
[42] J. Breuer a J. Roztočil, *Programové vybavení pro monitoring Ethernetových sítí s protokolem IEEE 1588*. 2012.
[43] EtherCat Technology Group, „EtherCAT – The Ethernet Fieldbus“. [Online]. Dostupné z: https://www.ethercat.org/. [Viděno: 24-čer-2016].
[44] RTnet Development Team, „RTnet – Hard Real-Time Networking for Real-Time Linux“. [Online]. Dostupné z: http://www.rtnet.org/. [Viděno: 24-čer-2016].
[45] V. Vančura, „Modul pro sběr dat s podporou IEEE 1588“, České vysoké učení technické v Praze, 2013.
[46] P. Hájek, „Modul pro synchronizaci sběru dat“, České vysoké učení technické v Praze, 2013.

[47] „NI PCI-1588 Interface – National Instruments“. [Online]. Dostupné z: http://sine.ni.com/nips/cds/view/p/lang/cs/nid/202345. [Viděno: 15-čer-2016].
[48] J. Breuer a V. Vigner, „Sleepy Cat IDE“. [Online]. Dostupné z: http://pck338-242.feld.cvut.cz/scide/. [Viděno: 15-čer-2016].
[49] „NetBeans IDE“. [Online]. Dostupné z: https://netbeans.org/. [Viděno: 15-čer-2016].
[50] „GCC ARM Embedded in Launchpad“. [Online]. Dostupné z: https://launchpad.net/gcc-arm-embedded. [Viděno: 15-čer-2016].
[51] „Open On-Chip Debugger“. [Online]. Dostupné z: http://openocd.org/. [Viděno: 15-čer-2016].
[52] J. Breuer, J. Fischer a J. Roztočil, „Comparison of IEEE 1588 Implementations using the STM32 Connectivity Line processor“, in *IMEKO Symposium TC-4, TC-19 & IWADC Instrumentation for the ICT Area*, Košice, SK, 2010, s. 70–73.
[53] STMicroelectronics, „IEEE 1588 precision time protocol demonstration for STM32F107 connectivity line microcontroller“, *AN3411*, čvc. 2011.
[54] J. Breuer, J. Fischer, J. Roztočil a V. Vigner, *Modul hlavních hodin*. 2011.
[55] „TI-EPL – Ethernet Phyter Library for DP83630 and DP83640 modified to be used with FreeRTOS“, *GitHub*. [Online]. Dostupné z: https://github.com/j123b567/ti-epl. [Viděno: 25-čer-2016].
[56] V. Vigner, J. Breuer, J. Roztočil a J. Fischer, *Modul pro časovou synchronizaci sběru dat = TriggerBox*. 2013.
[57] „Standard Commands for Programmable Instruments (SCPI)“, *SCPI-99*, kvě. 1999.
[58] J. Breuer, „Open Source SCPI device library“, *GitHub*. [Online]. Dostupné z: https://github.com/j123b567/scpi-parser. [Viděno: 15-čer-2016].
[59] J. Breuer, B. Čemusová, J. Fischer, J. Roztočil a V. Vigner, „Synchronization of Distributed Systems Using GPS“, in *Advanced Distributed Measuring Systems – Exhibits of Application*, Aalborg, DK: River Publishers, 2012, s. 95–120.
[60] J. Breuer, V. Vigner, J. Roztočil a J. Fischer, *Měřicí ústředna s protokolem IEEE1588*. 2013.
[61] „The Linux PTP Project“. [Online]. Dostupné z: http://linuxptp.sourceforge.net/. [Viděno: 15-čer-2016].
[62] „Real-Time Linux Wiki“. [Online]. Dostupné z: https://rt.wiki.kernel.org/. [Viděno: 23-čer-2016].
[63] V. Haasz, *„Projekt TAČR – č. TA01010988 – Časově synchronní distribuované systémy pro sběr dat a řízení procesů. Odborná zpráva o postupu prací a dosažených výsledcích za rok 2013. Příloha k průběžné zprávě za rok 2013.“* Praha, ČVUT 2014.

[64] „PRS10 – Rubidium Frequency Standard“. [Online]. Dostupné z: http://www.thinksrs.com/products/PRS10.htm. [Viděno: 23-čer-2016].
[65] „SCPI Parser / Interpreter Source Code. JPA Consulting Ltd.“ [Online]. Dostupné z: https://www.jpacsoft.com/. [Viděno: 16-čer-2016].
[66] „IEEE Standard Codes, Formats, Protocols, and Common Commands for Use With IEEE Std 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation“, *IEEE Std 488.2-1992*, s. 1–254, pro. 1992.
[67] S. Houlihan a A. Kalman, „Applying Standard Commands for Programmable Instruments (SCPI) to CubeSats“, *AIAA/USU Conference on Small Satellites*, srp. 2014.
[68] O. Guinnard, M. Stephan, R. Houlmann a H. Zbinden, „Easy ϕ“. [Online]. Dostupné z: http://www.easy-phi.ch/. [Viděno: 16-čer-2016].
[69] „From Beginners to Real Engineers“, *Red Pitaya*. [Online]. Dostupné z: http://redpitaya.com/. [Viděno: 27-čer-2016].
[70] D. Ries a J. Breuer, „e-mail: scpi-parser: possible issue with SCPI_ParamString“, 17-zář-2014.
[71] R. Reeder, W. Green a R. Shillito, „Analog-to-Digital Converter Clock Optimization: A Test Engineering Perspective“, *Analog Dialogue*, č. 42–2, s. 6–12, úno. 2008.
[72] S. Bregni, „Clock stability characterization and measurement in telecommunications“, *IEEE Transactions on Instrumentation and Measurement*, roč. 46, č. 6, s. 1284–1294, pro. 1997.
[73] M. Shinagawa, Y. Akazawa a T. Wakimoto, „Jitter analysis of high-speed sampling systems“, *IEEE Journal of Solid-State Circuits*, roč. 25, č. 1, s. 220–224, úno. 1990.
[74] J. Breuer a J. Roztočil, „Evaluation of Accuracy of Timestamping Using Direct Memory Access Controller“, in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS 2011) Proceedings*, Munich: Technical University of Munich, roč. 2011.
[75] „Red Pitaya V1.1 Open Source Instrument – Elektor“. [Online]. Dostupné z: https://www.elektor.com/red-pitaya-instrument. [Viděno: 29-čer-2016].

11 Seznam vlastních publikací

Publikace vztahující se k tématu této práce

Publikace v impaktovaných časopisech

Breuer, J. - Vigner, V. - Roztočil, J.
Precise packet delay measurement in an Ethernet network
In: Measurement. 2014, vol. 54, no. 54, p. 215-221. ISSN 0263-2241. (40 %)

Publikace ostatní

Breuer, J. – Vigner, V. – Roztočil, J.
Device for Precise Packet Delay Measurement
In: Proceedings of the 12th IMEKO TC10 Workshop on Technical Diagnostics. Florencie: Universita di Firenze, 2013, p. 66-71. ISBN 978-88-903149-8-8.

Breuer, J. – Roztočil, J. – Vigner, V.
Live Demonstration: Precise Time Protocol Tester
In: 14th IMEKO TC10 Workshop on Technical Diagnostics. New Perspectives in Measurements, Tools and Techniques for systems reliability, maintainability and safety. Milan, Italy, June 27-28, 2016, p. 63.

Breuer, J. – Čemusová, B. – Fischer, J. – Roztočil, J. – Vigner, V.
Synchronization of Distributed Systems Using GPS
In: Advanced Distributed Measuring Systems – Exhibits of Application. Aalborg: River Publishers, 2012, p. 95-120. ISBN 978-87-92329-72-1. (35 %)

Breuer, J. – Fischer, J. – Roztočil, J.
Comparison of IEEE 1588 Implementations Using the STM32 Connectivity Line Processor
In: IMEKO Symposium TC-4, TC-19 & IWADC Instrumentation for the ICT Area. Košice: Technical University of Košice, 2010, p. 70-73. ISBN 978-80-553-0424-3.

Breuer, J.
Hardware-Assisted IEEE 1588 Implementation Using the STM32 Connectivity Line Processor
In: POSTER 2010 - Proceedings of the 14th International Conference on Electrical Engineering. Prague: CTU, Faculty of Electrical Engineering, 2010, ISBN 978-80-01-04544-2.

Breuer, J. – Roztočil, J.
Evaluation of Accuracy of Timestamping Using Direct Memory Access Controller
In: 2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS 2011) Proceedings. Munich: Technical University of Munich, 2011, p. 1. ISBN 978-1-61284-892-1.

Breuer, J.
High Speed Timestamping in Microcontroller Using Direct Memory Access
In: POSTER 2011 - 15th International Student Conference on Electrical Engineering. Prague: CTU, Faculty of Electrical Engineering, 2011, p. 1-4. ISBN 978-80-01-04806-1.

Vigner, V. – Breuer, J.
Precise Synchronization in Large Distributed Systems
In: IDAACS 2013 – Proceedings of the 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems. Berlin: IEEE, 2013, p. 226-230. ISBN 978-1-4799-1426-5.

Breuer, J. – Fischer, J. – Roztočil, J. – Vigner, V.
Modul hlavních hodin
[Funkční vzorek]. 2011. (30 %)

Breuer, J. – Fischer, J. – Roztočil, J. – Vigner, V.
Modul obyčejných hodin
[Funkční vzorek]. 2011. (30 %)

Breuer, J. – Vigner, V. – Roztočil, J.
PTP Tester
[Funkční vzorek]. 2012. (40 %)

Vigner, V. – Breuer, J. – Roztočil, J. – Fischer, J.
Modul pro časovou synchronizaci sběru dat = TriggerBox
[Funkční vzorek]. 2013. (35 %)

Breuer, J. – Vigner, V. – Roztočil, J. – Fischer, J.
Měřicí ústředna s protokolem IEEE1588
[Funkční vzorek]. 2013. (35 %)

Breuer, J. – Roztočil, J.
Programové vybavení pro monitoring Ethernetových sítí s protokolem IEEE 1588
[Software splňující podmínky RIV (dřív Autorizovaný)]. 2012. (70 %)

Breuer, J. – Vigner, V. – Roztočil, J.
Precise Time Protokol daemon pro systém Windows
[Software splňující podmínky RIV (dřív Autorizovaný)]. 2013. (40 %)

Kaše, M. – Breuer, J. – Roztočil, J.
SW modul podporující protokol IEEE 1588 v systému DisCo
[Software splňující podmínky RIV (dřív Autorizovaný)]. 2013. (30 %)

Ostatní publikace

Breuer, J. – Vigner, V.
Sleepy Cat: an Open Source Development Platform for STM32F2 Microcontrollers
In: POSTER 2012 – 16th International Student Conference on Electrical Engineering. Praha: Czech Technical University in Prague, 2012, p. 1-4. ISBN 978-80-01-05043-9.

Breuer, J. – Vigner, V.
SCPI Parser Library for Small Devices
In: POSTER 2013 – 17th International Student Conference on

Electrical Engineering. Prague: Czech Technical University, 2013, p. 1-4. ISBN 978-80-01-05242-6.

Breuer, J. – Roztočil, J. – Vigner, V.
Software pro měření a kalibraci časových stupnic
[Software splňující podmínky RIV (dřív Autorizovaný)]. 2012. (30 %)

Breuer, J. – Vigner, V.
SCPI Parser
[Jiný software (nesplňující podmínky RIV)]. 2013.

Vigner, V. – Breuer, J. – Roztočil, J.
Synchronizační modul s GPS přijímačem Trimble
[Funkční vzorek]. 2013. (40 %)

Breuer, J. – Fischer, J. – Haasz, V. – Roztočil, J. – Vigner, V.
Měřicí systém pro sběr a analýzu dat
[Výzkumná zpráva]. 2014. 25 s.

Ohlasy

(Breuer, J. – Vigner, V. – Roztočil, J., 2014) citován v:

Chen, X. – Guo, H. – Crossley, P., „Interoperability Performance Assessment of Multivendor IEC61850 Process Bus“, IEEE Transactions on Power Delivery, roč. 31, č. 4, s. 1934–1944, srp. 2016.

Catelani, M. - Ciani, L., „Editorial“, Measurement, roč. 54, s. 178–179, srp. 2014.

12 Seznam symbolů a zkratek

API	Application Programming Interface
BC	Boundary Clock
BMCA	Best Master Clock Algorithm
E2E	End to End
GPIB	General Purpose Interface Bus
GPS	Global Positioning System
IRIG	Inter-Range Instrumentation Group
MAC	Media Access Controller
MTIE	Maximum Time Interval Error
NTP	Network Time Protocol
OC	Ordinary Clock
OCXO	Oven-Controlled Crystal Oscillator
P2P	Peer to Peer
PDV	Packet Delay Variation
PHY	Physical layer
PoE	Power Over Ethernet
PPS	Pulse Per Second
PTP	Precise Time Protocol dle standardu IEEE 1588
RTOS	Real-Time Operating System
SCPI	Standard Commands for Programmable Instruments
TAI	Temps Atomique International
TC	Transparent Clock
TCP	Transmission Control Protocol
TCXO	Temperature-Compensated Crystal Oscillator
TE	Time Error
TIE	Time Interval Error
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
UTP	Unshielded Twisted Pair
WSN	Wireless Sensor Network, bezdrátová senzorová síť

13 Přílohy

13.1 SCPI Parser

Implementace měřicích zařízení pro účely této práce nebo pro účely vedených bakalářských prací vyžadovala komunikační protokol. Bylo rozhodnuto o použití protokolu SCPI. Implementace na straně řídicí jednotky je poměrně jednoduchá a lze ji realizovat například pomocí knihovny VISA. Na straně zařízení již tak jednoduchá situace není a na počátku neexistovala dostupná a dobře použitelná knihovna pro implementaci tohoto protokolu. Existovala komerční knihovna JPA-SCPI Parser [65], která ale byla napsána velice komplikovaným způsobem, který neumožňoval její jednoduché použití.

V bakalářských pracích studenti implementovali některé aspekty tohoto protokolu, ale většinou velice zjednodušeně a bez implementace zásadních funkcí protokolu (stavový model, atp.).

Vznikla tedy univerzální, znovu použitelná, otevřená knihovna pro implementaci SCPI protokolu na straně zařízení. V této knihovně stačí pouze nadefinovat základní funkce pro čtení vstupu a výstupu, definovat strom příkazů pomocí vzorů a následně již jen implementovat jednotlivé příkazy.

Klíčové vlastnosti této knihovny jsou

- Implementace dle specifikace SCPI-99 [57] a IEEE 488.2 [66] včetně stavového modelu.
- Uvolněno pod permisivní FreeBSD licencí
- Orientace na nízkou paměťovou náročnost – použití v MCU
- Varianty pro ARM, PIC16, PIC18, PIC32mx, AVR a všechny běžné procesory s UNIXlike OS (x86, x86_64, ARM Cortex-A8, …).
- Veřejně dostupná [58]

Knihovna implementuje funkci pro zpracování vzorů příkazů, příkazy tedy lze definovat stejně, jako je běžné v dokumentaci k SCPI přístrojům. Knihovna implementuje základní vzory podle tab. 13.1. Vzory lze přímo tímto způsobem psát do definice zdrojového souboru a knihovna podle nich bude rozpoznávat jednotlivé příkazy.

Tab. 13.1: Podporované vzory příkazů

Vlastnost	**Příklad vzoru**
Krátká a dlouhá forma	`MEASure` znamená i příkaz `MEAS` i příkaz `MEASURE`
Obecné příkazy	`*CLS`
Složené příkazy	`CONFigure:VOLTage`

Dotazy	`MEASure:VOLTage?, *IDN?`
Volitelné části příkazu	`MEASure:VOLTage[:DC]?`
Číselný sufix příkazu	`OUTput#:FREQunecy`

Dále jsou připraveny API funkce pro všechny základní datové typy. Tyto funkce slouží pro jednoduché vypisování odpovědí a pro zjednodušení zpracování uživatelských parametrů. Podporované datové typy jsou v tab. 13.2. Datové typy lze i kombinovat a odvozovat a tak definovat nové datové typy. Lze tak například definovat odvozený typ logické hodnoty, který bude reagovat na „ON“, „OFF“, „0“ nebo „1“ a vše ostatní bude považovat za chybu.

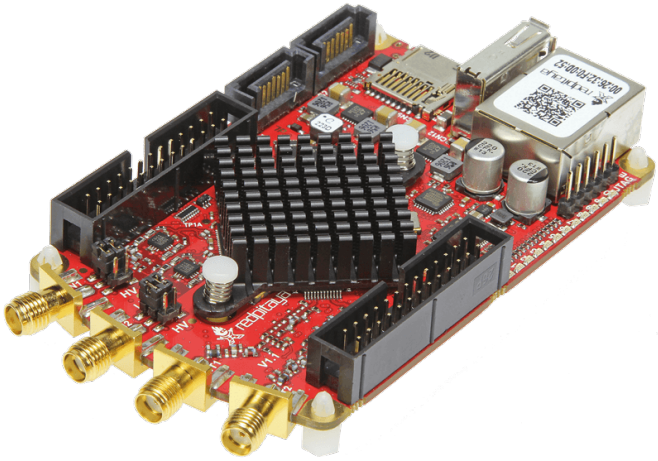
Tab. 13.2: Podporované datové typy

Typ	**Příklad**
Desítkové číslo	10, 10.5
Desítkové číslo se sufixem	-5.5 V, 1.5 KOHM
Hexadecimální číslo	#HFF
Binární číslo	#B11
Osmičkové číslo	#Q77
Text	"text", 'text'
Binární data s definovanou délkou	#12AB
Konstantní data	MINimum, DEFault, INFinity
Seznam čísel	(1,2:50,80)
Seznam kanálů	(@1!2:3!4,5!6)
Ostatní výrazy	(1)

Po čas existence této knihovny se stala populární a začala se používat v několika projektech po celém světě od studentských projektů, přes akademické laboratoře po komerční měřicí systémy. Řada uživatelů přispěla do knihovny svými modifikacem.

Nejzajímavější projekty, které dnes tuto knihovnu využívají, jsou implementace SCPI do CubeSat firmou Pumpkin, Inc. [67], Otevřený standard pro měřicí přístroje Easy ϕ od Université de Genève [68] na obr. 13.1 a otevřená platforma pro implementaci měřicích přístrojů Red Pitaya [69] na obr. 13.2. Další firmy a organizace používají tuto knihovnu pro implementaci svých měřicích zařízení, např. institut Ultra Cold Neutron Group, Paul Scherrer Institute [70] ji používá pro implementaci vlastních experimentálních přístrojů.

Obr. 13.1: Easy ϕ – Otevřený standard pro vědecké přístroje (fotografie © 2016 University of Geneva [68])



Obr. 13.2: Red Pitaya – Otevřený měřicí přístroj (fotografie © 2016 Elektor International Media B.V. [75])

13.2 Sleepy Cat IDE

Vývojové prostředí Sleepy Cat IDE, SC-IDE, vzniklo jako odpověď na neexistenci programátorsky přívětivého vývojového prostředí pro mikrokontroléry. Všechna dosud existující vývojová prostředí nedisponovala komfortem, na který je zvyklý například programátor z PC platformy.

Vývojové prostředí vzniklo za podpory grantu FRVŠ 2011/2011. Zásluhou tohoto grantu vzniklo vývojové prostředí pro výuku mikroporcesorové techniky. Díky dobré použitelnosti byly všechny další funkční vzorky, nástroje a knihovny pro mikrokontroléry programovány v tomto prostředí. [35]

Vývojové prostředí umožňuje základní strukturování projektu do virtuálních složek a tím např. rozlišit jednotlivé knihovny. Lze tak mít přehledně v jednom projektu více různých knihoven. Častá vada různých vývojových prostředí je, že není možné mít dva soubory se stejným názvem, ale v jiných adresářích. Při kompilaci nastane kolize a projekt nelze celý sestavit. V tomto prostředí je to elegantně vyřešeno taj, že každý soubor se kompiluje do vnořené složky podle adresáře, ve kterém je uložen. Tím je zaručena bezkolizní kompilace souborů se stejným názvem ale v různých knihovnách.

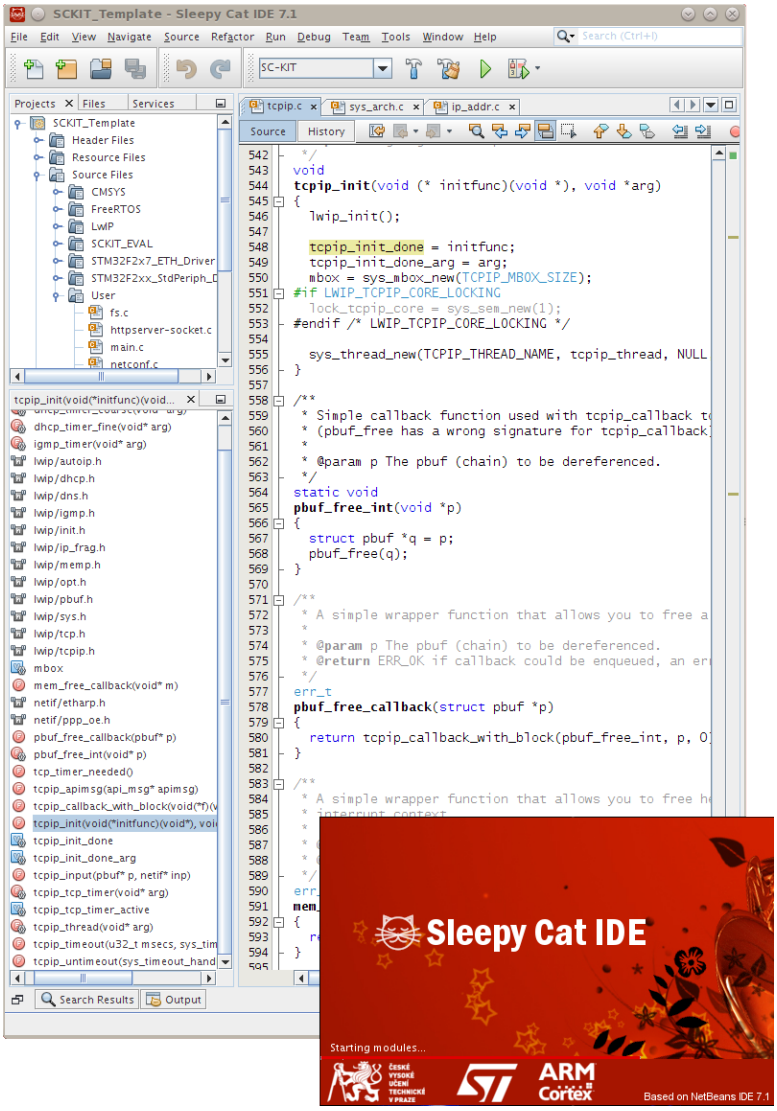
Vývojové prostředí disponuje sadou kompilačních profilů, lze tak jednoduše přepínat různé varianty sestavení. Typické využití je sestavení pro produkční nastavení nebo pro ladění. Tyto profily lze ale využít i pro vytváření binárního firmware pro různé typy hardware, který sdílí stejný kód, ale liší se jen několika direktivami preprocesoru.

Všechny soubory projektu jsou využity pro generování automatického doplňování. Není tak třeba pamatovat si přesné názvy funkcí v projektu nebo používat vyhledávání. Při otevření souboru lze navíc použít navigátor, díky kterému lze zobrazit seznam všech funkcí, proměnných a datových typů a jednoduše je v souboru lokalizovat.

Lze definovat vzory pro automatické generování kódu pro různé zkratky. Po zapsání např. `/**<Enter>` nad deklarací funkce se vytvoří automaticky komentář dané funkce a připraví se položky pro jednotlivé parametry a návratovou hodnotu. Komentáře se generují ve formátu JavaDoc a lze pomocí nich vygenerovat dokumentaci ke zdrojovým kódům.

Vývojové prostředí disponuje funkcí ladění kódu, kdy lze přidávat kukátka na proměnné. Lze kód kdykoli zastavit a pouhým najetím myši na proměnnou se zobrazí její hodnota. Lze přidávat breakpoint, který slouží k zastavení programu při průchodu daným místem a watchpoint, který slouží k zastavení programu při čtení/zápisu paměťového místa.

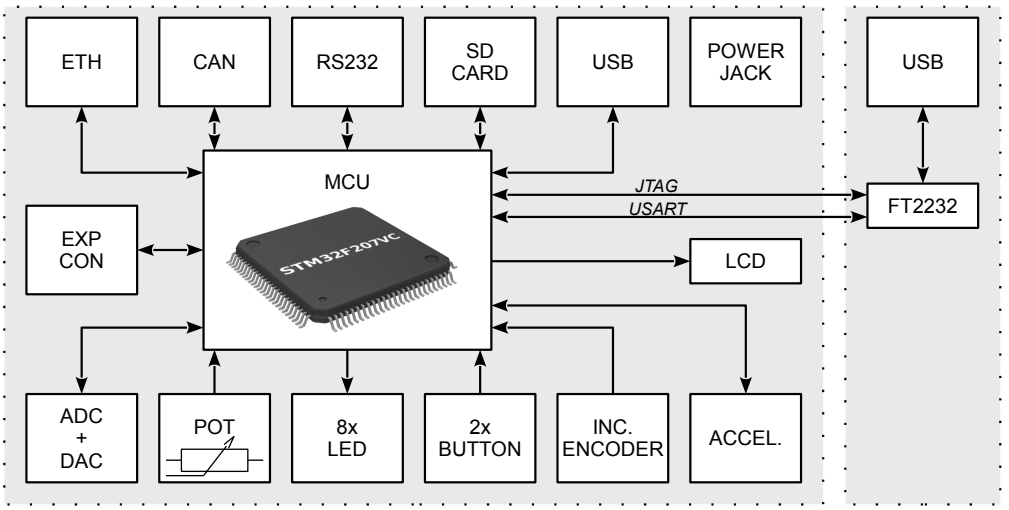
Při ladění lze zobrazovat běžící procesy podporovaných RTOS včetně základní diagnostiky obsazené paměti apod.



Obr. 13.3: Sleepy Cat IDE

13.3 Sleepy Cat KIT

Za podpory grantu FRVŠ 2011/2011 vznikl vývojový kit pro výuku mikroprocesorové techniky s mikrokontrolérem ARM Cortex-M3. Vývojový kit disponuje integrovaným JTAG adaptérem pro nahrávání a ladění kódu a integrovaným USB/UART převodníkem pro přímé vypisování ladících informací. Jak JTAG tak převodník UART/USB jsou vyvedeny do společného USB, ze kterého je možné kit napájet. Stačí tedy pouze jeden propojovací kabel.



Obr. 13.4: Blokové schéma Sleepy Cat KIT

Na obr. 13.4 je znázorněno blokové schéma a všechny vyvedené periferie. Pro tuto práci je především důležitý expanzní konektor, který bylo možné použít pro připojení externí ethernetové PHY. Díky tomu se tento kit stal platformou pro vývoj zařízení s přesnou časovou synchronizací jak přímo na procesoru, tak i s využitím speciální ethernetové PHY.

Na obr. 13.5 je vidět fotografie výsledného kitu. Zajímavé jsou na něm integrovaný maticový displej a analogové vstupy a výstupy vyvedené přes BNC konektory. Veškeré expanzní konektory byly řešeny tak, aby je nebylo snadné zničit.

Vývojový kit byl vybaven vlastním vývojovým prostředím, které je popsané v kapitole 13.2. Aby se studentům s kitem dobře pracovalo, byl vybaven i krabičkou na přenášení, do které se vešel kit, napájecí zdroj, kabeláž a případně malé nepájivé pole. Krabička s kitem je vyobrazena na obr. 13.6.

Pro studenty vznikly i podklady pro výuku a zadání úloh. Za dobu výuky vzniklo mnoho zajímavých úloh, zejména osciloskop ve webovém rozhraní, magická kreslící tabulka, automaticky generované bludiště ovládané nakláněním desky, zobrazení jednoduché obrázkové prezentace apod.

Pro kit byly dostupné vzorové příklady. Byly zprovozněny vzorové příklady od STMicroelectronics na ovládání periferií. Byl vytvořen příklad na komunikaci po TCP/IP – jednoduchý webserver a byl vytvořen příklad pro použití operačního systému FreeRTOS na kitu.

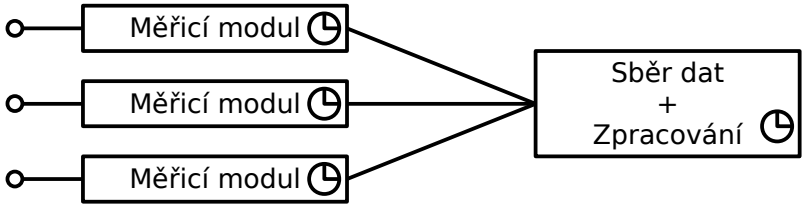
Obr. 13.5: Sleepy Cat KIT

Obr. 13.6: Krabička pro přenášení kitu pro studenty

13.4 Odhad efektivního počtu bitů heterogenního systému pro sběr dat

13.4.1 Popis problému

Celý problém lze popsat na příkladu vícekanálového heterogenního vzorkovacího systému. Systém se skládá z jednotlivých dílčích měřicích modulů, které jsou vzájemně propojeny a synchronizovány.



Obr. 13.7: Heterogenní systém pro sběr dat

Při následném zpracování dat z jednotlivých modulů má každý vzorek svoji časovou značku, která je ale určena s nejistotou. Tato nejistota je především dána přesností synchronizace. Pro potřeby praktického měření lze odhadnout vlastnosti signálu, který získáme z více synchronizovaných měřicích modulů.

Základní popis lze předvést na zjednodušeném zapojení, kdy všechny měřicí moduly měří stejný vstupní signál. Pokud při sběru dat sloučíme všechny vzorky dohromady získáme jeden signál. Každý vzorek má vypočítatelnou časovou značku. Přesnost časové značky je dána přesností synchronizace. Na všechny vzorky tak můžeme pohlížet jako na vzorky z jednoho modulu a na nejistotu časové značky jako na jitter vzorkovacího systému.

Metoda má sloužit k získání základní představy o takto získaném heterogenním systému. Porovnání parametrů bylo zvoleno převedení informace o přesnosti synchronizace na efektivní počet bitů (ENOB) analogově digitálního převodníku (ADC), který je možné při dané vzorkovací frekvenci teoreticky dosáhnout.

13.4.2 Výpočet odhadu

Výpočet vychází ze vztahu pro odstup signálu od šumu (SNR) pro vzorkovací systém

$$SNR = -10\log_{10}\left((2\pi f_a t_{jrms})^2 + \frac{2}{3}\left(\frac{1+\epsilon}{2^N}\right)^2 + \left(\frac{2\sqrt{2}V_{NOISErms}}{2^N}\right)^2\right), \tag{13.1}$$

kde SNR je odstup signálu od šumu v dB, f_a je frekvence vstupního sinusového analogového signálu s plným rozkmitem, t_{jrms} je kombinovaná

efektivní hodnota jitteru interního ADC a externích hodin, ϵ je průměrná diferenciální nelinearita (DNL) ADC v LSB, N je rozlišení ADC v počtu bitů a $V_{NOISErms}$ je efektivní vstupní šum ADC. [71]

Pro $\epsilon=0$, $t_{jrms}=0$ a $V_{NOISErms}=0$ se vztah zjednoduší do formy

$$SNR=20\log_{10}(2)\cdot N+10\log_{10}(3/2) \tag{13.2}$$

kde N je rozlišení ADC v počtu bitů. Efektivní počet bitů pak lze spočítat pomocí vztahu

$$ENOB=\frac{SNR-10\log_{10}(3/2)}{20\log_{10}(2)} . \tag{13.3}$$

Pomocí vztahu (13.1) a (13.3) lze odvodit pouze vliv jitteru na efektivní počet bitů pro $\epsilon=0$ a $V_{NOISErms}=0$

$$ENOB=N-\log_2\sqrt{6(\pi f_a t_{jrms}2^N)^2+1} , \tag{13.4}$$

kde N je rozlišení ADC v počtu bitů, f_a je frekvence vstupního sinusového analogového signálu s plným rozkmitem a t_{jrms} je efektivní hodnota jitteru.

Pro potřeby odhadu horní meze můžeme výraz dále zjednodušit. Hodnota pod odmocninou je pro vysoká N mnohonásobně větší než 1 a jedničku v takovém případě můžeme zanedbat. Pro vysoká N lze tedy výraz zjednodušit

$$ENOB=\lim_{N\to\infty}\left(N-\log_2\sqrt{6(\pi f_a t_{jrms}2^N)^2}\right) \tag{13.5}$$

$$ENOB=-\log_2(\sqrt{6}\pi f_a t_{jrms}) .$$

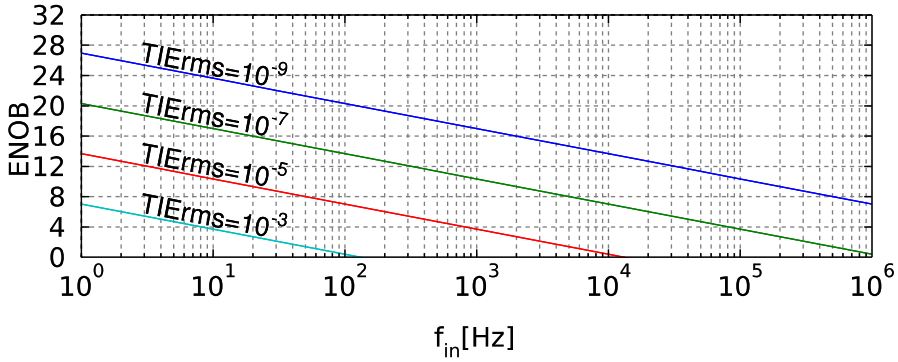
Pro odhad horní meze hodnoty efektivního počtu bitů pro ideální ADC s původně nekonečným počtem bitů pak platí vztah (13.5).

Pro potřeby synchronizace můžeme jako hodnotu jitteru použít efektivní hodnotu Time Interval Error (TIE_{rms}) [72], popsanou v kapitole 4.6.1. ADC bereme jako ideální a ostatní parametry zanedbáme, protože nás zajímá především vliv přesnosti synchronizace a nikoli zvoleného ADC.

Změřením přesnosti synchronizace TIE_{rms} například pomocí PPS výstupu lze následně odhadnout horní mez efektivního počtu bitů vzorkovacího systému pro danou vstupní frekvenci.

Ze závislosti je patrné, že například při $TIE_{rms}=100\,ns$ lze například u vstupního signálu do 5 kHz očekávat ENOB nad 8 bitů. Při požadavku na lepší parametry je třeba dosáhnout lepší synchronizace nebo data z různých zdrojů předzpracovat samostatně. Pokud se například ze signálu detekuje čas nějaké události nezávisle na ostatních kanálech, pak tento čas

lze brát jako potřebu jednoho bitu. V takovém případě stačí horší synchronizace pro srovnání času událostí.



Obr. 13.8: Závislost ENOB na frekvenci vstupního signálu pro různá TIE_{rms}

Pouhé propojení naměřených dat z více zdrojů pak bude v nejlepším případě vykazovat parametry dle obr. 13.8.[73], [74] Při propojování dat z více zdrojů je tedy třeba dbát na dodržení podmínek, za kterých mají smysl. Ze samotné přesnosti synchronizace není tento požadavek patrný a může to svádět k podcenění situace.