České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra řídicí techniky

Diplomová práce

Rozvrhování statického segmentu sítě FlexRay

Bc. David Beneš

Vedoucí práce: Doc. Dr. Ing. Zdeněk Hanzálek

Studijní program: Otevřená informatika, Navazující magisterský

Obor: Počítačové inženýrství

3.1.2012

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 3.1.2012 ….....

Abstract

Diploma thesis describes a FlexRay communication system and propose an algorithm for scheduling signals in the static segment of FlexRay network with graphical display possibility of the resulting communication schedule. Furthermore, the basic variant of proposed scheduling is implemented and is further enhanced. Different variants of parts of the algorithm are implemented. All proposed variants are compared to each other in the performance test, and also compared with the existing scheduling methods. In conclusion, the algorithm requirements are given with instructions about its usage.

Keywords

FlexRay, time constraints, scheduling, frame packing, heuristic

Abstrakt

Diplomová práce popisuje komunikační systém FlexRay a navrhuje algoritmus pro rozvrhování signálů ve statickém segmentu sítě FlexRay s možností grafického zobrazení výsledného rozvrhu komunikace. Dále implementuje základní variantu navrženého algoritmu, a tu dále vylepšuje. Jsou testovány rozličné varianty částí algoritmu pro rozvrhování. Všechny vytvořené varianty jsou nakonec navzájem porovnány ve výkonnostním testu, a také s existujícími možnostmi rozvrhování. V závěru práce jsou vyjmenovány požadavky implementovaného algoritmu s návodem na jeho použití.

Klíčová slova

FlexRay, časová omezení, rozvrhování, frame packing, heuristika

Obsah

Úvod ... 1
1. Specifikace komunikačního systému FlexRay ... 3
1.1. Fyzické uspořádání klastru ... 4
1.2. Architektura nodu ... 6
1.3. Komunikační protokol sítě FlexRay ... 7
1.4. Synchronizace, odolnost vůči rušení ... 9
2. Rozvrhování statického segmentu ... 11
2.1. Stávající možnosti ... 16
2.2. Implementace vlastního algoritmu ... 17
2.2.1. Nastavená omezení ... 17
2.2.2. Popis algoritmu ... 19
2.2.3. Příklad běhu vlastního algoritmu ... 23
2.3. Vylepšení algoritmu ... 27
2.3.1. Rozdělení signálů k nodům ... 27
2.3.2. Frame packing ... 28
2.3.3. Optimalizace zpráv ... 30
2.3.4. Tvorba rozvrhu ... 31
3. Výsledky algoritmu ... 37
3.1. Výkonnostní testy ... 39
3.2. Srovnání s existujícími metodami ... 42
3.3. Ověření algoritmu na fyzické síti ... 43
4. Popis a požadavky algoritmu ... 45
4.1. Předpoklady pro běh algoritmu ... 45
4.2. Popis jednotlivých souborů ... 46
4.2.1. Další pomocné soubory a soubory nevyužívané přímo algoritmem ... 47
4.3. Potřebné úpravy před spuštěním algoritmu ... 48
4.4. Očekávaný vstupní formát souborů pro algoritmus ... 49
4.5. Generovaný výstupní formát souboru ... 50
Závěr ... 51
Seznam literatury a použitých zdrojů ... 53

Příloha A: Seznam použitých zkratek..... 57
Příloha B: Obsah přiloženého CD..... 59

Seznam obrázků

Obrázek 1 - Příklady elektronických systémů v automobilu [21] 2
Obrázek 2 - Přímé propojení dvou nodů 4
Obrázek 3 - Pasivní sběrnice 4
Obrázek 4 - Pasivní hvězda 5
Obrázek 5 - Aktivní hvězda 5
Obrázek 6 - Hybridní topologie 5
Obrázek 7 - Odlišné topologie komunikačních kanálů 6
Obrázek 8 - Architektura nodu 7
Obrázek 9 - Podrobné dělení cyklu 8
Obrázek 10 - Okamžiky kontroly synchronizace a aplikace jejich výsledků 9
Obrázek 11 - Přenos asynchronního a synchronního signálu [11] 13
Obrázek 12 - Znázornění časového okna signálu 15
Obrázek 13 - Ukázka hyperperiody 18
Obrázek 14 - Grafické znázornění signálů vstupujících do algoritmu 24
Obrázek 15 - Grafická reprezentace vytvořených zpráv 25
Obrázek 16 - Grafické zobrazení vytvořených optimalizovaných zpráv 26
Obrázek 17 - Grafická reprezentace celkového vygenerovaného rozvrhu 27
Obrázek 18 - Zpřísňování časových omezení během frame packingu a optimalizace zpráv 27
Obrázek 19 - Příklad problému pro heuristiku a vyřešení ILP 37
Obrázek 20 - Sestava testovací sítě FlexRay 44
Obrázek 21 - Komunikace na sběrnici při praktickém testu zobrazena na osciloskopu [11] 44

Seznam tabulek

Tabulka 1 - Seznam signálů použitých v plně vybaveném automobilu nižší třídy.....14
Tabulka 2 - Seznam signálů na vstupu algoritmu 24
Tabulka 3 - Zprávy vytvořené ze signálů 25
Tabulka 4 - Optimalizované zprávy vytvořené ze zpráv 26
Tabulka 5 - Porovnání výsledků verzí algoritmů 40
Tabulka 6 - Části algoritmu použité v jednotlivých variantách algoritmu 41

Seznam pseudokódů

Pseudokód 1 – Základní struktura algoritmu 19
Pseudokód 2 - Vylepšená verze frame packingu 29
Pseudokód 3 - Vylepšená verze optimalizace zpráv 31
Pseudokód 4 - Spolupráce heuristiky a ILP během tvorby rozvrhu 33
Pseudokód 5 - Tvorba rozvrhu z optimalizovaných zpráv 35

Úvod

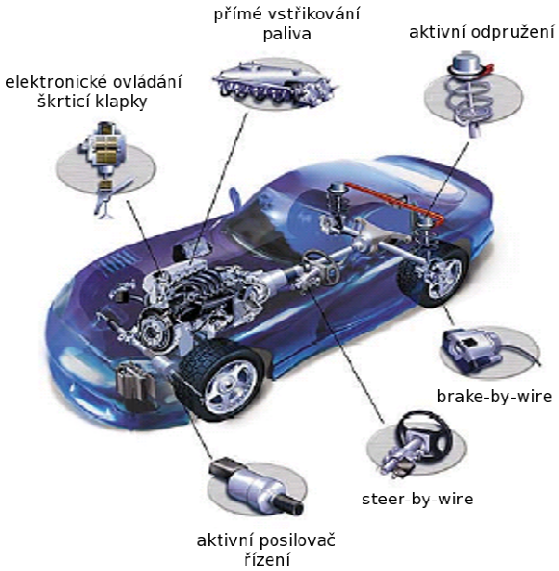
Automobily hrají v každodenním životě lidí, především ve vyspělých zemích, stále důležitější roli. Jen málokdo si už v dnešní společnosti dokáže představit svou činnost bez produktů automobilového průmyslu. Navíc požadujeme, aby byly automobily stále bezpečnější, poskytovaly více pohodlí a komfortu a kladly menší nároky na řidiče. Zároveň moderní společnost požaduje od automobilů šetrnost k životnímu prostředí a tím i zlevnění jejich provozu.

Prostředkem k vyhovění výše uvedeným požadavkům jsou mimo jiné pokročilé elektronické systémy, které jsou využívány v nově vyráběných automobilech. S globálním rozvojem elektroniky se do automobilů dostává stále více pokročilých systémů, které jejích služeb využívají. Pro komplexnější elektronické systémy je nezbytné propojení jejich jednotlivých částí tak, aby spolu mohly vzájemně komunikovat. Například adaptivní tempomat, který udržuje nastavenou rychlost automobilu. Pokud vozidlo s adaptivním tempomatem dojede pomaleji jedoucí vůz, přizpůsobí se jeho rychlosti, aby nedošlo ke kolizi. Aby mohl takový systém fungovat, je nezbytně nutné získávat minimálně údaje z předního senzoru vzdálenosti a aktuální rychlost, aby mohl v případě potřeby ovládat brzdy a plynový pedál. Příklady dalších systémů, které potřebují pro svou optimální činnost komunikovat s ostatními komponentami vozu, jsou zobrazeny na Obrázku 1.

Vzhledem k množství v autech instalovaných systémů není možné propojit jednotlivá zařízení každé s každým. S výhodou se v tomto případě využívají komunikační sběrnice. V současné době je stále ještě nejvyužívanější sběrnicí CAN, která však nevyhovuje požadavkům nejmodernějších aplikací. Nově nasazované elektronické systémy vyžadují velkou propustnost sběrnice, minimální zpoždění a především spolehlivost, což jsou charakteristiky, kterými CAN nedisponuje. Z toho důvodu byl vyvinut nový systém komunikace s názvem FlexRay. Ten plně vyhovuje i nejnáročnějším požadavkům, tj. dává například k dispozici široké přenosové pásmo a podporuje jak time-triggered, tak i event-triggered komunikaci (oproti CAN, která podporuje jen druhou zmíněnou). Komunikace na síti FlexRay je rozdělena na dvě části, přičemž první se využívá především pro časově kritické aplikace (time-triggered) a druhá část je pro ostatní komunikaci (event-triggered). Nicméně i druhou část lze

využít pro přenos kritických dat.

Přestože ke komunikačnímu systému FlexRay je již vydána specifikace, jeho nasazení v praxi stále není běžné. Jedním z důvodů je nutnost konfigurace každého připojeného zařízení na tuto sběrnici. Před konfigurací zařízení je třeba navrhnout harmonogram komunikace. Právě tvorba tohoto harmonogramu je složitou operací, kterou lze pro menší počet zařízení navrhnout ručně. S narůstajícím počtem komunikujících komponent se však ruční návrh stává téměř nerealizovatelným. Automatický návrh harmonogramu komunikace na síti FlexRay, konkrétně první (statický) segment, je cílem této diplomové práce. Nejprve budou analyzovány již existující možnosti tvorby harmonogramu komunikace pro statický segment a následně bude navržen vlastní algoritmus pro vhodné rozvržení přenosu dat po síti FlexRay. Tento algoritmus bude postupně co nejvíce generalizován a upravován dle dalších požadavků, které by mohly vyvstat při potřebě nalezení vhodného rozvrhu komunikace. Výsledky algoritmu budou porovnány se známými výsledky jiných metod tvorby harmonogramu komunikace. Pro potvrzení, že se jedná o validní rozvrh, bude harmonogram vytvořený vzniklým algoritmem prakticky otestován na fyzické síti FlexRay.



Obrázek 1 - Příklady elektronických systémů v automobilu [21]

1. Specifikace komunikačního systému FlexRay

Jedním z impulsů pro vývoj nového komunikačního systému byl požadavek na zavádění systémů x-by-wire. Systémy x-by-wire v automobilovém průmyslu usilují o nahrazení mechanického a hydraulického propojení ovládacích prvků za ovládání pomocí propojení elektrickými signály. Například pro systém steer-by-wire již není oproti minulosti volant automobilu mechanicky spojen s řídícími tyčemi rejdových kol, ale natočení volantu je převáděno na elektrický signál. Ten je přenášen k řídícím tyčím a ty jsou následně podle hodnoty signálu vychýleny tak, aby rejdová kola zatočila do požadovaného úhlu. Převodem otočení volantu na elektrický signál je možné měnit intenzitu natočení rejdových kol v závislosti na aktuálních situacích. Například při jízdě ve vysoké rychlosti není žádoucí, aby byl volant moc citlivý. Naopak při parkování je výhodnější, pokud se malý pohyb volantem promítne do většího natočení rejdových kol. Podobně je možné u systému steer-by-wire kompenzovat například silný boční vítr. Za normálních okolností by řidič musel i při jízdě rovně držet volant natočen proti směru větru. Toto potřebné natočení je možné díky systému steer-by-wire kompenzovat. Dalším příkladem využití je zamezení nabourání vedle jedoucího automobilu v případě, že se řidič rozhodne změnit jízdní pruh, ale nevšimne si, že ve vedlejším pruhu na jeho úrovni jede jiné vozidlo. V neposlední řadě jsou díky nahrazení mechanického propojení elektronikou značně sníženy výdaje na údržbu. Dalšími systémy x-by-wire jsou například drive-by-wire (ovládání plynového pedálu, respektive otáček motoru) a brake-by-wire (ovládání brzd). Je zřejmé, že se jedná o velmi kritické aplikace, a proto je zajištění naprosté spolehlivosti těchto systémů zásadním požadavkem. Nasazení systémů x-by-wire není horkou novinkou, třeba systém fly-by-wire je úspěšně využíván v letectví již několik desetiletí. Nově se však systémy x-by-wire zavádí do automobilového průmyslu. Při využití více systémů, které vyžadují komunikaci jednotlivých částí mezi sebou, je nutné přenášet obrovské množství dat. Nesmíme při tom zapomínat, že určité nezanedbatelné množství elektronických systémů v dnešních automobilech již používáno je. Oblast jejich nasazení sice nebývá kritická, nicméně i ony přenášejí určitý datový objem, který musíme brát v úvahu při návrhu komunikačního systému.

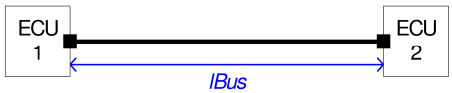
Specifikace FlexRay vznikla z výše zmíněné potřeby širšího, spolehlivějšího

a předvídatelného přenosového pásma v automobilovém průmyslu, a to především pro schopnost využívat stále většího počtu elektronických zařízení, která spolu musí komunikovat. V roce 2000 bylo předními výrobci automobilového průmyslu založeno konsorcium FlexRay, které spolupracovalo na vývoji stejnojmenného komunikačního systému. Konsorcium FlexRay bylo na konci roku 2010 rozpuštěno, protože již publikovalo finální specifikace, čímž byl naplněn důvod jeho založení. Specifikace komunikačního systému FlexRay verze 3.0.1 byla předána organizaci ISO, aby tato specifikace byla vydána jako standard pro silniční vozidla.[9] Navržený komunikační systém sítě FlexRay definuje jak parametry fyzické vrstvy, tak i protokol komunikace a chování jednotlivých zařízení vzhledem k síti. Zároveň doporučuje i architekturu připojených jednotek. Jelikož je již v síti FlexRay definováno prakticky vše, pokud není řečeno jinak, vychází tato kapitola z publikovaných specifikací konsorcia FlexRay, konkrétně ze specifikací komunikačního protokolu[7], a fyzické vrstvy[8]. Dále lze nalézt podrobnější popis jak jednotlivých částí protokolu FlexRay, tak i detaily ohledně fyzické vrstvy v diplomové práci [14], která se zabývala jak teoretickou částí komunikačního systému FlexRay, tak realizací této fyzické sítě.

Samotná síť FlexRay sestává z jednotlivých komunikujících jednotek (nodů), komunikačních kanálů a případně opakovačů. Všechny nody, které jsou navzájem propojeny (mohou spolu tedy komunikovat), dohromady s jejich propojeními vytvářejí klastr.

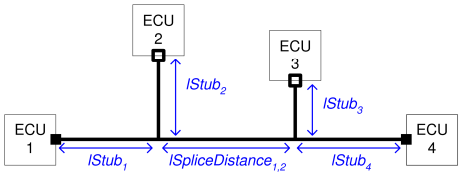
1.1. Fyzické uspořádání klastru

Komunikační systém FlexRay neomezuje topologii klastru, pokud jsou zachovány požadované vlastnosti jako například maximální doba šíření signálu. Topologie nesmí obsahovat žádné logické kružnice, nebo uzavřené smyčky.



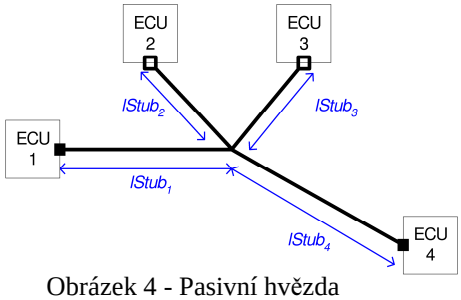
Obrázek 2 - Přímé propojení dvou nodů

Nejjednodušší topologií je přímé propojení dvou nodů (viz Obrázek 2). Maximální délka propojení *Ibus* je dána elektrickými parametry propojení, mírou elektromagnetického rušení atd.

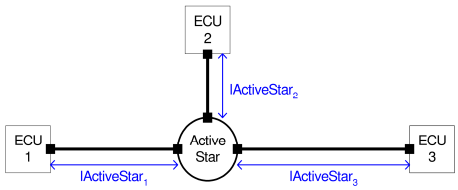


Obrázek 3 - Pasivní sběrnice

Klíčové je, aby signál odeslaný z jednoho nodu dorazil s jistotou do druhého nodu při zachování požadovaných parametrů. Velmi často využívanou topologií je pasivní sběrnice (viz. Obrázek 3). Zde je pro korektní fungování sítě nutné dodržovat maximální délku připojení nodů ke sběrnici a maximální vzdálenosti míst na sběrnici, ke kterým jsou nody připojeny. Další možností je zapojení do pasivní hvězdy (viz. Obrázek 4), což je speciální případ pasivní sběrnice. V tomto zapojení záleží zachování minimálních vlastností přenášených signálů jak na počtu takto propojených nodů, tak i na jejich maximální vzdálenosti od středu hvězdy. Lepší variantou předchozí topologie je aktivní hvězda (viz. Obrázek 5). U té je využito přímých propojení mezi opakovačem a nody. Opakovač má schopnost předávat signály přijaté z jedné větve na větve ostatní s obnovenou charakteristikou signálu. S využitím opakovače určují maximální množství k němu připojených nodů jen jeho vlastnosti.

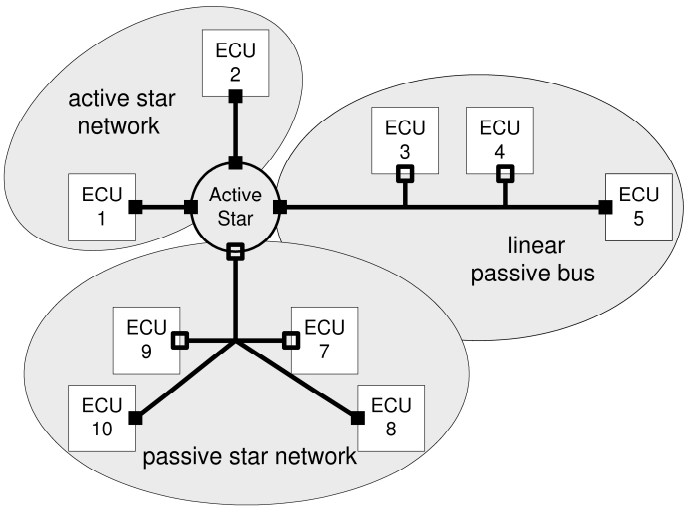


Obrázek 4 - Pasivní hvězda



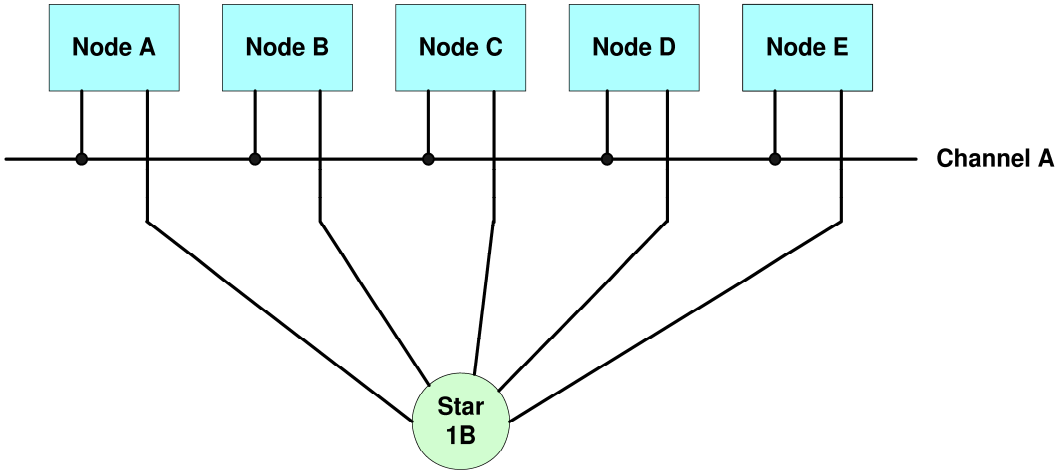
Obrázek 5 - Aktivní hvězda

Z těchto základních topologií je možné jejich kombinováním sestavovat složitější struktury. Například opakovač lze využít pro prodloužení maximální délky propojení. Příklad zapojení základních topologii pro vytvoření hybridní topologie je na Obrázku 6.



Obrázek 6 - Hybridní topologie

Komunikační systém FlexRay umožňuje a zároveň doporučuje využívat dvou oddělených nezávislých komunikačních kanálů. Toho lze využít například pro zajištění funkcionality klastru a to i v tak extrémních situacích, jako je nehoda a při ní mechanické přerušení komunikačního kanálu na jedné straně vozu. I v takových situacích je totiž vyžadována činnost některých ze systémů. Jednotlivé kanály mohou být zapojeny do odlišných topologií jako například na Obrázku 7.

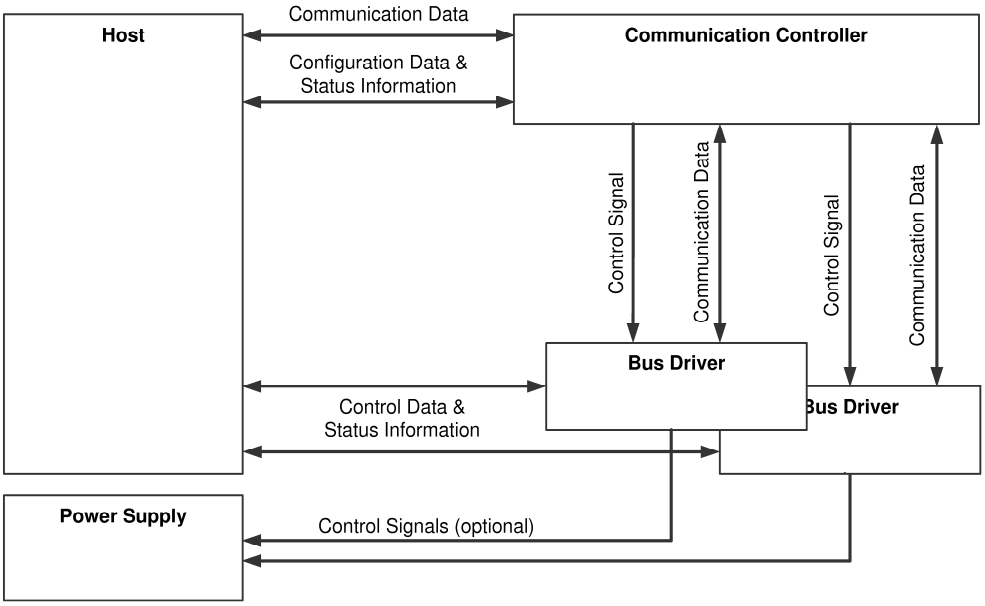


Obrázek 7 - Odlišné topologie komunikačních kanálů

Přestože jsou podporována rozličná fyzická uspořádání klastru, logické uspořádání ve všech případech zůstává sběrnicí.

1.2. Architektura nodu

V souvislosti s přesně definovaným chováním nodu podle specifikace komunikačního systému FlexRay je uvedena a doporučena následující architektura komunikujících zařízení. Architektura popisovaná ve specifikacích FlexRay je zobrazena na Obrázku 8.



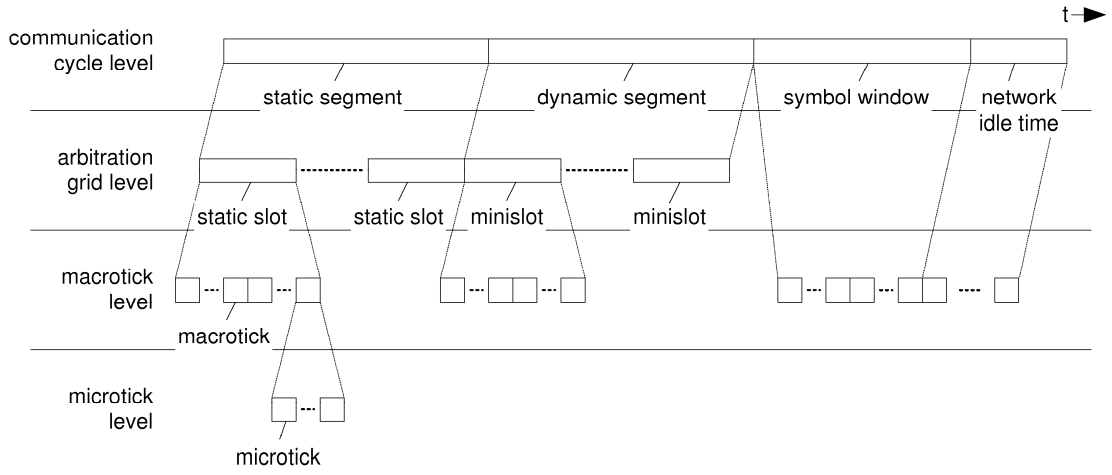
Obrázek 8 - Architektura nodu

Na obrázku jsou zobrazeny jednotlivé části nodu důležité pro node ke komunikaci po síti FlexRay. Část *Host* představuje samotnou funkcionalitu tohoto zařízení. *Power Supply* je část, která zajišťuje informace o napájení. *Bus Driver* zajišťuje přístup na komunikační sběrnici sítě FlexRay. Část *Bus driver* je zdvojena, neboť musí být pro každý kanál separátní. *Communication Controller* zprostředkovává převod dat na sběrnici a operace vyžadované komunikačním protokolem FlexRay. Šipky ukazují, které části mezi sebou komunikují. Detailní popis jednotlivých částí a jejich vlastní architektura je dostupná v [7]. V detailním popisu je taktéž přesně definován způsob, jakým spolu jednotlivé části komunikují. Dodržení doporučené architektury nodu a funkčnosti jednotlivých částí zaručuje dodržení požadavků pro komunikaci na síti FlexRay.

Pokud je potřeba připojit node do více klastrů, je nutné využít pro každý klastr vlastní Communication Controller. Připojení přes jeden Communication Controller zapojený kanálem A z jednoho klastru a kanálem B do druhého klastru není povoleno.

1.3. Komunikační protokol sítě FlexRay

Komunikace na síti FlexRay probíhá v jednotlivých po sobě jdoucích cyklech pevné délky. Cyklus je rozdělen na statický segment, dynamický segment, symbolové okno a klidový stav.



Obrázek 9 - Podrobné dělení cyklu

Statický segment sestává z pevného počtu stejně dlouhých slotů. Statickým slotům jsou přiřazena ID dle jejich pořadí v cyklu. Každý slot je dále dělen na makrotiky a ty dále na mikrotiky, jak je znázorněno na Obrázku 9. V této části je pro přístup ke sběrnici využita statická metoda TDMA. To znamená, že každý slot je přiřazen jednomu nodu, který ho může použít k odesílání dat po síti. Tím je dosaženo silně deterministického chování. Jednomu nodu může být přiřazeno i více slotů, avšak jeden slot není povoleno využívat více nody. Přiřazení slotů k nodům musí být definováno při návrhu komunikace, není jej možno měnit za běhu. Alokace slotů pro jednotlivé kanály je nezávislá. Statický segment se pro své vlastnosti využívá především pro kritické aplikace, neboť tak je zaručeno maximální zpoždění doručení potřebných dat. Tento segment je vhodný i pro signály, které jsou generovány periodicky.

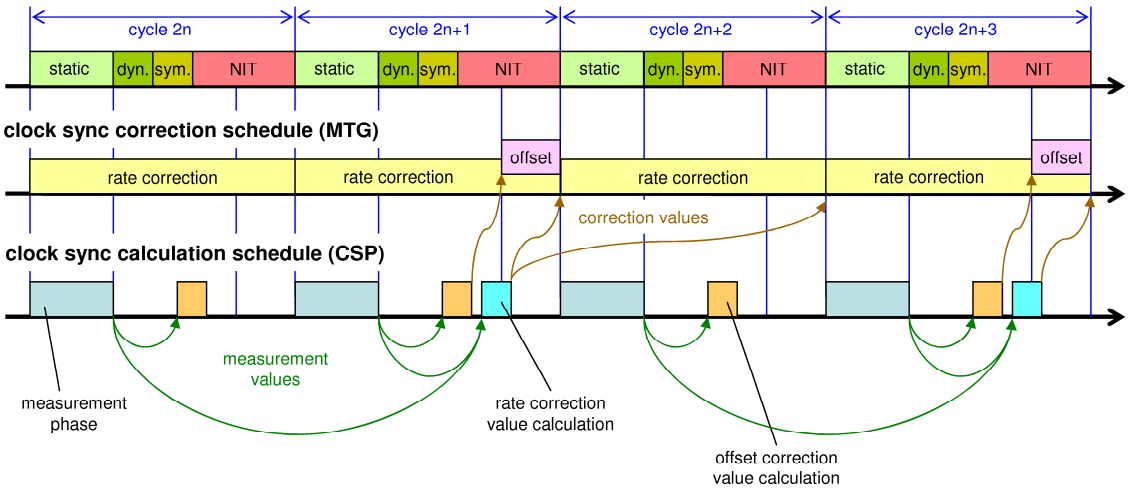
Dynamický segment je složen z minislotů, které nemají pevně stanovenou velikost. Z toho důvodu může dynamický segment v každém cyklu obsahovat různý počet minislotů. Minisloty jsou dále děleny na makrotiky. Dynamický segment má pevnou velikost v makroticích nadefinovanou při návrhu sítě. Dynamický segment není povinnou součástí cyklu. V tomto segmentu nody přistupují k síti dle nastavených priorit. Dynamický segment lze proto využít i pro kritické aplikace, a to v případě že jim je přiřazena vysoká priorita a nebude jich mnoho. Nejpraktičtější je tento segment pro data, která je třeba přenášet sporadicky, či jsou generována pouze při určité události. Vzhledem k přístupu k síti podle priority v dynamickém segmentu a nedeterminističnosti generování jednotlivých signálů, není možné zaručit maximální latenci přenosu signálů s nižší prioritou.

Symbolové okno se využívá pro kontrolu správné funkce některých částí nodů a při startu sítě.

Klidový stav (network idle time, NIT) je poslední částí každého cyklu. V klidovém stavu nody počítají a aplikují své časové korekce. Tato část má tedy proměnnou délku. Změna délky této části se využívá k synchronizaci nodů.

1.4. Synchronizace, odolnost vůči rušení

Aby mohly nody navzájem komunikovat vysokou rychlostí, je nezbytné, aby byly navzájem časově synchronizovány. Synchronizace času je v komunikačním systému FlexRay realizována na několika úrovních. Ve statickém segmentu každého cyklu si všechny nody měří rozdíl mezi skutečným a očekávaným začátkem slotů. Z těchto hodnot si každý node počítá potřebný posun, aby se při dalších cyklech skutečné začátky slotů ve statickém segmentu nelišily od předpokládaných. Tato korekce se provádí úpravou délky klidového stavu v každém lichém cyklu. Dále nody v každém lichém cyklu měří rozdíly předchozích hodnot mezi lichým a sudým cyklem. Z toho je v každém cyklu prováděna úprava frekvence změnou počtu mikrotiků délky komunikačního cyklu. Okamžiky kontroly synchronizace a jejich korekce jsou znázorněny na Obrázku 10.



Obrázek 10 - Okamžiky kontroly synchronizace a aplikace jejich výsledků

Jelikož spolehlivost byla při návrhu sítě FlexRay jednou z klíčových vlastností, je tento požadavek hluboce zanesen v konceptu této sítě. Čtení aktuálního stavu na

sběrnici totiž nevyhodnocuje ve skutečnosti hodnotu jen v jediný okamžik. Hodnota jednoho bitu informace předané vyšším vrstvám je majoritní hodnota pěti čtení ze sběrnice. To je samozřejmě respektováno i při odesílání dat na sběrnici. Například pro odesílání logické 1 to ve výsledku znamená vygenerování pěti logických 1 na síť. Díky tomuto mechanismu získává síť FlexRay značnou robustnost. Pokud by došlo třeba kvůli rušení k poškození přenášené informace takovým způsobem, že by v každém přenášeném bitu informace byly maximálně dvě z pěti hodnot na síti změněny, nody stále získávají platné původní informace. Samozřejmostí je pak využití kontrolního součtu CRC na úrovni protokolu sítě FlexRay. Pokud by i přes výše popsané mechanismy došlo k chybnému přenosu dat, je tento stav alespoň indikován a komunikační jednotky mají možnost na daný problém zareagovat.

Další možností, jak dosáhnout vyšší spolehlivosti v klastru, je využití obou komunikačních kanálů.

2. Rozvrhování statického segmentu

Jsou uvažovány dva druhy úloh prováděných v nodech (ECU). Prvním typem jsou úlohy které běží bez synchronizace s komunikačními cykly. Signály (neboli přenos údajů jedné ECU) generované takovou úlohou jsou označovány za asynchronní. Druhý typ signálů, které jsou generovány úlohami synchronizovanými s komunikačními cykly, se nazývá synchronní. Úkolem je najít vhodné přiřazení signálů do rámců a ty přiřadit do slotů a cyklů, ve kterých budou přenášeny po síti FlexRay. Rámci jsou nazývány jednotlivé sloty v konkrétních cyklech.

Konfigurace sítě FlexRay sestává z velkého množství parametrů, jakými jsou například délka cyklu, počet statických slotů ve statickém segmentu, délka statického slotu a další. Tyto parametry jsou většinou předdefinovány systémovým konstruktérem a dále dodržovány dodavateli ECU. Nastavení používané v této práci je založeno na návrhu sítě dle BMW[4]. Délka komunikačního cyklu F je zvolena 5ms, přičemž pro statický segment jsou vyhrazeny 3ms. Zbytek komunikačního cyklu je využit pro dynamický segment, symbolové okno a klidový stav. Ve zvolené konfiguraci je k dispozici M = 75 statických slotů s délkou L 0,04ms. Pro síť s propustností 10Mb/s je v každém slotu možné přenést W bitů, což je v tomto případě 128 bitů.

Jednotlivé signály jsou definovány následujícími proměnnými (je použita stejná notace jako v [10]):

- N_i unikátní identifikátor ECU, která odesílá signál i,
- T_i perioda signálu i; předpokládá se přenos signálu po síti maximálně jednou za cyklus,
- O_i release time signálu i; určuje nejpozdější chvíli, po které je vygenerován první výskyt signálu i od začátku periody,
- C_i velikost signálu i v bitech,
- A_i stáří signálu i je doba mezi vygenerováním signálu v ECU, která jej odesílá, a přijetím prvního rámce,

- D_i deadline signálu i, což je maximální stáří signálu, do kdy je nutné signál i přenést.

Dalším předpokladem je dodržování standardu AUTOSAR[3] využívaného nejen pro automobilový průmysl, ale i pro letectví a další odvětví. V kontextu AUTOSAR je rámec definován následovně:

- Q_j unikátní identifikátor ECU, která odesílá rámec j,
- S_j identifikátor slotu, ve kterém je využit pro přenos rámce j,
- R_j periodicita AUTOSAR rámce j; perioda rámce vyjádřená v celistvých násobcích délky komunikačního cyklu; tato perioda může nabývat pouze hodnot $\{2^n | n \in N, n \leq 6\}$,
- B_j identifikátor komunikačního cyklu, ve kterém se přenáší první výskyt rámce j, přičemž platí, že $B_j \leq R_j$.

Každý rámec j je tedy charakterizován čtveřicí $\{Q_j, S_j, B_j, R_j\}$. V souladu se standardem je vyžadováno, že rozvrh komunikace je statický a není možné jej za běhu měnit. Dále se předpokládají velikosti signálů maximálně do délky statického slotu. To znamená, že data jednoho signálu nemohou být rozdělena do několika rámců. Fragmentování zpráv je zajištěno pomocí AUTOSAR FlexRay transportní vrstvy protokolu [2].

Převzorkováním asynchronních signálů je možné získat a i z těchto signálů vytvořit signály synchronní pomocí následující úpravy parametrů

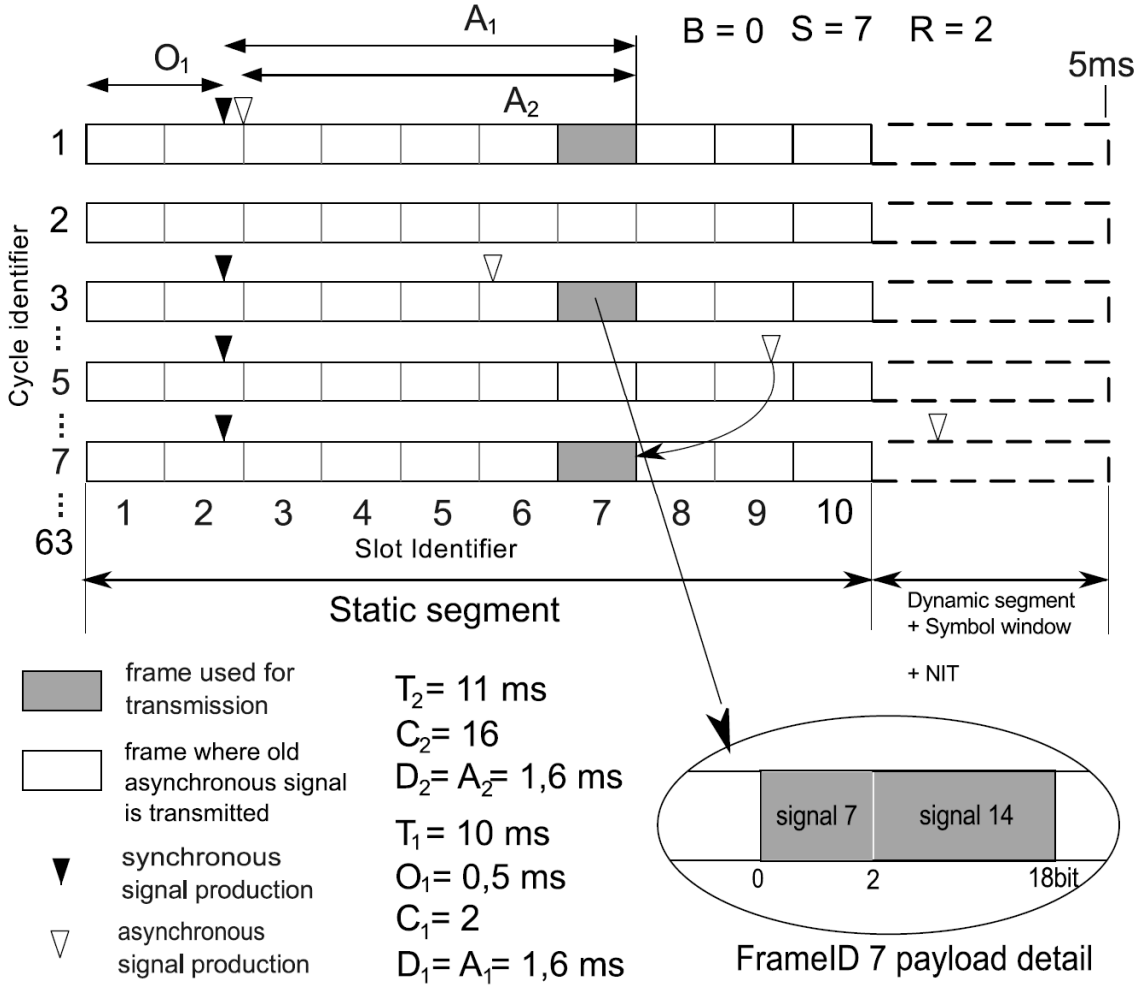
$$D_i = T_i = 2n \cdot F + P + U$$
$$O_i = 0$$

kde P je čas na zabalení signálu i – čas od požadavku úlohy k odeslání signálu i až po okamžik skutečného přenosu signálu; U je čas rozbalení signálu opačným způsobem.

Synchronní signály mohou mít nastaven deadline na kratší hodnotu, než je jejich perioda. V tom případě je deadline omezen následujícím způsobem.

$$D_i^S \geq \begin{cases} P+L+U & pro\ \ 0 \leq O_i - n \cdot F \leq P+M \cdot L+U \\ P+L+G+U & v\ \ ostatních\ \ pripadech \end{cases}$$

kde $n \in 1,2,4,...,64$, M je počet statických slotů, L představuje délku statického slotu a G reprezentuje délku dynamického segmentu, klidového stavu a symbolového okna.



Obrázek 11 - Přenos asynchronního a synchronního signálu [11]

Obrázek 11 znázorňuje situaci, kdy jsou v ECU vygenerovány dva signály – jeden synchronní a jeden asynchronní. Perioda synchronního signálu je 10ms a asynchronního 11ms. Předpokládejme, že oba signály mají deadline roven periodě. Pro splnění deadline je nutné přenášet oba signály každý druhý cyklus, tedy $R_j = 2^1$. V pátém cyklu jsou přenášena pro druhý signál stejná data jako ve třetím cyklu, neboť nová data ještě nebyla vygenerována. To je způsobeno převzorkováním asynchronního signálu, tedy převodem na synchronní signál.

Množiny signálů

Pro potřeby vývoje algoritmu byly použity modifikované množiny signálů Society of Automotive Engineers (SAE). Zpráva SAE [16] definuje množinu signálů posílaných mezi 7 různými subsystémy v modelovém automobilu. Tato množina signálů sestává z 53 jednotlivých a periodických signálů. Periody signálů jsou rozděleny mezi 4 pevně definované periody- 5, 10, 100, 1000ms. Periodické signály vyžadují maximální latenci menší, či maximálně rovnou uvedeným periodám. Jednotlivé signály mají svou maximální latenci definovanou úlohou, ze které vznikly: například signály vytvořené jako důsledek jednání řidiče mají maximální latenci nastavenu na 20ms. Detailnější popis množiny signálů SAE je dostupný v [13].

Kvůli dostupnému širokému přenosovému pásmu sítě FlexRay a vysokým požadavkům elektronických systémů v dnešních automobilech, bylo nutné množinu signálů SAE dále rozšířit. Rozšíření množiny signálů SAE bylo provedeno pomocí generátoru testovacích dat NETCARBENCH [5] následujícím způsobem:

- zvýšení počtu signálů při zachování stejného pravděpodobnostního rozdělení parametrů jako v původní množině signálů,
- přidání omezujících časových parametrů.

Množina signálů použitých v plně vybaveném automobilu nižší třídy je znázorněna v Tabulce 1.[11] Signály jsou posílány mezi 33 nody propojenými do dvou sítí s bránou. Několik z těchto signálů je posíláno jen nastane-li konkrétní událost a mají definován pouze deadline, tj. za jak dlouho od události musí být doručeny. Takové signály jsou převedeny na periodické s periodou rovnou deadline.

Perioda	# signálů	Perioda	# signálů
7	26	100	272
8	51	200	117
10	143	250	22
20	205	400	2
40	7	500	265
50	51	1000	77

Tabulka 1 - Seznam signálů použitých v plně vybaveném automobilu nižší třídy

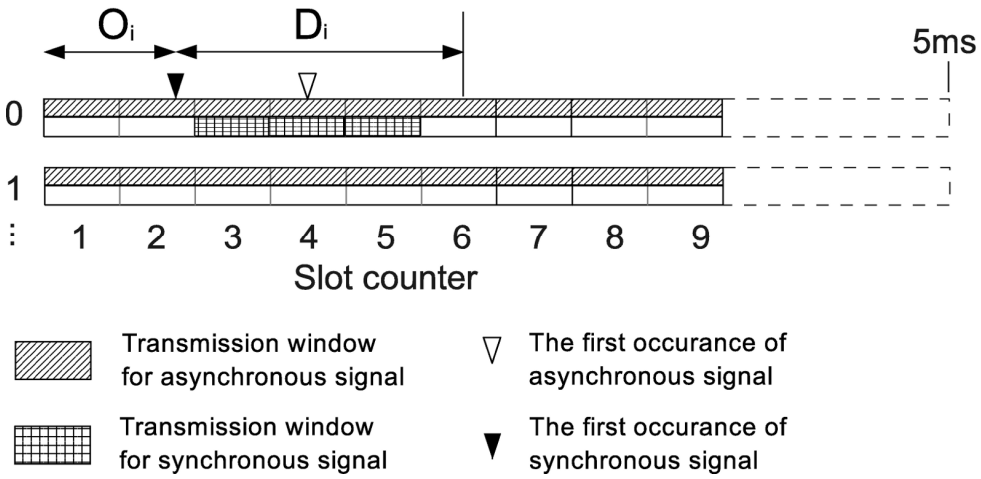
Časová omezení

Prvním krokem po vygenerování množiny signálů je nalezení dvojice $\{E_i, H_i\}$ pro každý signál. E_i udává první rámec, ve kterém může být odeslán první výskyt signálu při dodržení všech časových omezení. H_i určuje poslední rámec, ve kterém je ještě povoleno signál odeslat podle zadaných časových omezení. Dvojice $\{E_i, H_i\}$ je vypočtena následujícím způsobem:

$$E_i = floor\left(\frac{P+O_i}{F}\right)\cdot M + \min\left(ceil\left(\frac{\left(P+O_i - floor\left(\frac{P+O_i}{F}\right)\cdot F\right)}{L}\right), M\right)$$

$$H_i = floor\left(\frac{P+O_i}{F}\right)\cdot M + \min\left(floor\left(\frac{\left(P+O_i+D_i - floor\left(\frac{P+O_i}{F}\right)\cdot F\right)}{L}\right), M\right)$$

Obrázek 12 je grafickým znázorněním dvojice $\{E_i, H_i\}$ pro dva signály – jeden asynchronní a jeden synchronní s nastaveným striktním časovým omezením.



Obrázek 12 - Znázornění časového okna signálu[1]

[1] WARAUS, Denis; 2011

2.1. Stávající možnosti

Konsorcium FlexRay vydalo v roce 2005 specifikaci komunikačního systému FlexRay verze 2.1. Není proto divu, od té doby vzniklo několik studií na téma rozvrhování signálů na síti FlexRay. Rozvrhování signálů v dynamickém segmentu je podobné jedné z nejpoužívanější síti CAN, a je proto hojně v těchto studiích zastoupeno. Dále je uváděno jen rozvrhování statického segmentu sítě FlexRay.

Autoři [15] využívají všech možností voleb parametrů pro konfiguraci cyklu v protokolu sítě FlexRay. Optimalizují tak přenosové pásmo přesně pro rozvrhované signály. Takový přístup je však značně limitující, protože v automobilovém průmyslu je často využíván přírůstkový model[11]. V takovém případě není možné upravovat parametry sítě FlexRay. Podobný přístup je také využit v [20]. V této práci se navíc nejdelší prioritní signály přesunují do dynamického segmentu. Díky tomu lze volit statický slot kratší a zabránit tak zbytečnému mrhání přenosovou kapacitou sítě FlexRay pro signály, které přenáší méně dat. Pokud je takto navržený systém dále rozšířen, nemusí být zaručeno včasné doručení dříve rozvržených signálů. V [17] bylo zkoumáno sdružování signálů do zpráv, které vede k efektivnějšímu využití sítě FlexRay díky potřebě alokace menšího počtu slotů. V uvedené práci je popsáno sdružování signálů do zpráv o jednotné velikosti, a to kvůli definici statických slotů v síti FlexRay, což vede k potřebě užšího přenosového pásma. Dále je v uvedeném článku zformulován způsob nalezení množiny optimálních zpráv ze vstupní množiny signálů pomocí nelineárního celočíselného programování. Je zde navržen způsob rozvrhování pro co nejmenší jitter a zároveň nejmenší počet využitých statických slotů, což zvyšuje efektivitu využití sítě FlexRay. Autoři [10] předpokládají využití standardu AUTOSAR a využívají heuristickou analýzu k rozvržení větších množin signálů. V této práci je však předpokládáno, že sdružování signálů do zpráv je prováděno na aplikační úrovni v ECU.

Další kategorií jsou práce, které se při rozvrhování signálů zabývají zajištěním vyšší míry spolehlivosti i na aplikační úrovni. Vyšší spolehlivost je v tomto případě zajišťována redundantním odesíláním signálů vyžadujících nejvyšší míru jistoty doručení. Touto problematikou se zabývají [23], [25] a [26].

2.2. Implementace vlastního algoritmu

Po prostudování existujících možností rozvrhování statického segmentu sítě FlexRay bylo přistoupeno k návrhu vlastního algoritmu, jak signály v této části rozvrhnout. Nejzajímavější existující prací byla [18], která je velmi inspirativní a to především v oblasti sdružování signálů do zpráv a dále způsobem přidělování zpráv do slotů. Z toho důvodu bylo rozhodnuto, že uvedené myšlenky budou využity i v našem algoritmu. Avšak na rozdíl od [18], náš algoritmus uvažuje release time i deadline signálů, což dělá úlohu rozvrhování statického segmentu sítě FlexRay obtížnější, než v uvedeném článku.

Pro tvorbu algoritmu bylo vybráno prostředí Matlab. Pro grafický výstup zobrazující vypočtený harmonogram komunikace je potřeba mít v prostředí Matlab doplňky XML Toolbox [19] a TORSCHE Scheduling Toolbox [24], pomocí kterých se vygeneruje XML soubor reprezentující výsledný rozvrh. Grafická reprezentace rozvrhu ve formátu XML je poté odeslána webové službě na serveru rtime.felk.cvut.cz, která data zpracuje pomocí nástroje na kreslení ganttových diagramů s využitím Metapostu Psgantt [12].

2.2.1. Nastavená omezení

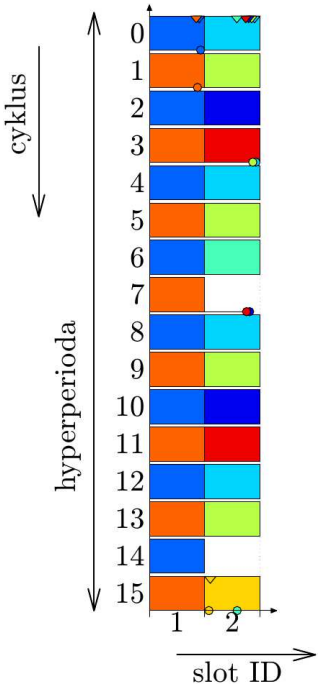
Budeme rozvrhovat signály, které jsou přiřazeny n nodům. Každý signál bude:

- přiřazen k nodu n,
- mít definovánu svou velikost C_i, což je délka signálu v bitech,
- mít svůj release time O_i, označující nejdřívější čas, kdy může být odeslán první bit signálu,
- mít svůj deadline D_i definovaný jako nejpozdější okamžik, kdy musí být poslední bit signálu přijat,
- mít periodu P_i, která vyjadřuje, jak často se signál opakuje.

Hodnoty release time a deadline budou definovány jen pro první výskyt signálu

v hyperperiodě, avšak platí i pro ostatní výskyty signálu v rámci jeho periody s příslušným posunem. Je nutné definovat hyperperiodu, která v našem případě vyjadřuje časový interval, který je periodicky opakován a po kterém začíná vypočtený rozvrh opět od začátku. Pro statický segment sítě FlexRay je tedy možné určit hyperperiodu jako nejmenší společný násobek všech period signálů. Hyperperioda tedy vyjadřuje počet cyklů pro které je vypočten rozvrh. Poté probíhá komunikace znovu stejně jako od začátku rozvrhu. Ukázkový rozvrh s vyznačenou hyperperiodou je na Obrázku 13. Hyperperioda je zde určena nejméně často se opakujícím žlutým signálem, který je umístěn v rozvrhu ve druhém slotu v posledním cyklu.

Velmi důležitým rozhodnutím bylo uvažovat release time a deadline pouze v násobcích délky cyklů přesto, že dle specifikace FlexRay mohou tyto hodnoty uvažovat i okamžiky v rámci cyklů. Rozhodnutí bylo provedeno na základě skutečnosti, že umístění signálu v rámci cyklu je bezvýznamné vzhledem k umístění signálu v cyklu v rámci hyperperiody, neboť délka hyperperiody je až o několik řádů delší než délka cyklu. Díky této volbě může být algoritmus výrazně jednodušší bez většího vlivu na data, která mají být rozvržena. Na vstupu algoritmu budou všechny hodnoty release time a deadline upraveny na celé násobky cyklů takovým způsobem, aby vždy splňovaly původní zadání. Pokud bude například release time signálu uprostřed cyklu, nastavíme release time signálu na konec tohoto cyklu; naopak pokud deadline signálu bude uprostřed cyklu, nastavíme deadline signálu na konec předchozího cyklu. Uvedeným způsobem nejsme schopni rozvrhnout signály, které mají release time i deadline v rámci jednoho cyklu. Takové signály se však v praxi nevyskytují. Toto omezení ve výsledku znamená, že nemusíme určovat, který node bude přiřazen k nižším ID slotům a který k vyšším, čímž lze jinak dosáhnout upřednostnění signálů odesílaných nodem, kterému je přiřazen slot s nižším ID. Zároveň nepovolujeme deadline delší, než je perioda signálu. Pokud by taková situace na vstupu nastala, nastavíme si pro naše potřeby deadline signálu na hodnotu rovnou periodě. Převod release time a deadline na celé násobky cyklů je



Obrázek 13 - Ukázka hyperperiody

proveden podle následujících vzorců:

$$\tilde{O}_i = ceil\left(\frac{E_i}{M}\right)$$

$$\tilde{D}_i = floor\left(\frac{H_i}{M}\right)$$

Dalším zvoleným omezením je, že periody signálů přicházejících na vstup algoritmu mohou nabývat pouze hodnot mocnin dvou. I v tomto případě mluvíme o nepříliš omezující vlastnosti, která však výrazným způsobem zjednodušuje výpočet finálního rozvrhu.

Ve výsledném rozvrhu nepovolujeme jitter, tedy odchylku od periodicity signálu.

2.2.2. Popis algoritmu

První verze algoritmu stejně jako například [18] neuvažovala u signálů release time a deadline, ale pouze sdružovala signály do zpráv a velmi jednoduše je přiřazovala do slotů. Tato verze sloužila k ověření základního rozvrhování a poté byla využita jako základ pro další rozšíření.

Během modifikace základního algoritmu a začleňování podpory časových omezení (release time a deadline) byl algoritmus přetvořen do podoby znázorněné v Pseudokódu 1.

```
Vstup: C, T, O', D'
Výstup: rozvrh

signály = rozděl signály k nodům(C, T, O', D');
for (k=1...#nodů) {
    zprávy(k) = frame packing(signály(k));
    optzprávy(k) = optimalizuj zprávy(zprávy(k));
    rozvrh(k) = vytvoř rozvrh(optzprávy(k));
end;
if (rozvrh.slotů > M)
    error('Nepodařilo se nalézt korektní rozvrh');
end;
```

Pseudokód 1 – Základní struktura algoritmu

Rozdělení signálů k nodům

Díky rozhodnutí o časových omezeních signálů, která mohou nabývat pouze násobků délky cyklu, je možné rozvrhovat signály jednotlivých nodů nezávisle na ostatních nodech. Z toho důvodu jsou v tomto kroku signály rozděleny do skupin podle toho, ke kterému nodu patří. Uvedené rozdělení nám umožní například pro velké množiny signálů rozvrhovat jednotlivé nody paralelně, čímž lze zkrátit dobu výpočtu.

V této části algoritmu jsou tedy projity všechny signály na vstupu algoritmu a jsou z nich vytvořeny množiny signálů podle toho, ke kterému nodu signál náleží. K signálům navíc přiřadíme periody a jejich časová omezení. U časových omezení kontrolujeme, zda jsou nastavena na hodnoty násobků délky cyklu. Pokud nejsou, upravíme tyto hodnoty dle dříve popsaného postupu, tedy zmenšíme časové okno, kdy může být signál přenesen. Tím zaručíme, že vždy dojde k dodržení původních časových omezení.

Asymptotická časová složitost rozdělování signálů k nodům je, $O(S)$ kde S je počet signálů na vstupu algoritmu.

Frame packing

Jak již bylo zmíněno, byla využita myšlenka z [18] nejdříve sdružit signály do zpráv (frame packing). Kvůli rozdílu oproti zmíněnému materiálu v respektování časových omezení signálů, je řešení sdružování signálů do zpráv v této diplomové práci odlišné. Zpráva v zásadě představuje obálku nad více signály. Díky spojení více signálů do jedné zprávy je omezen počet elementů, se kterými je potřeba pracovat a je nutné je rozvrhnout. Právě nižší počet rozvrhovaných elementů značně snižuje výpočetní náročnost tvorby výsledného harmonogramu komunikace. Zpráva má stejné parametry jako signál, tedy velikost C_i, která je pro zprávu rovna součtu velikostí signálů v ní obsažených; velikost zprávy nesmí překročit velikost rámce; periodu T_i, která v tomto kroku musí být stejná jako původní perioda všech signálů ve zprávě; release time $\tilde{O}_i$, odpovídající nejpozdějšímu času release time ze všech signálů přiřazených do zprávy; a podobně deadline $\tilde{D}_i$, který naopak nabývá hodnoty nejdřívějšího z časů deadline signálů obsažených ve zprávě. Tímto nastavením parametrů zpráv zajistíme (podobně jako při rozdělování signálů k nodům), že pokud měl signál některé z časových omezení

uprostřed cyklu, dojde ke zpřísnění všech časových omezení. Tímto způsobem zaručíme, že všechna původní omezení budou zachována.

Sdružování signálů do zpráv funguje odděleně pro signály s různými periodami. Z toho vyplývá, že prvním krokem je rozdělení signálů do skupin podle jejich periody. Poté se postupně prochází skupiny signálů se stejnou periodou. Pro každý signál se hledá zpráva, do které by mohl být signál umístěn. Vybírá se pouze ze zpráv se stejnou periodou, které mají ještě dostatek volného místa, aby se do zprávy signál vešel (a nepřekročila poté zpráva velikost rámce), a zpráva má navíc takové časové omezení, že pokud se do ní signál vloží, je zaručeno, že signál nebude vyslán dříve, než je specifikováno, a zároveň dorazí včas. Pokud je nalezena zpráva, která odpovídá výše popsaným podmínkám, je do ní signál přidán a u zprávy se upraví její velikost dle velikosti vloženého signálu. Dále se musí upravit časová omezení zprávy tak, aby byla splněna omezení všech signálů ve zprávě (tedy i právě přidaného signálu). Pokud není žádná vhodná zpráva nalezena, je vytvořena zpráva nová, do té se vloží aktuální signál a provedou se stejné operace, jako při vkládání signálu do již existující zprávy. V tomto případě zpráva převezme stejné parametry, jaké má právě přidávaný signál.

Sdružení signálů do zpráv je realizováno v asymptotické časové složitosti $O(s^2)$, kde s je počet signálů pro právě procházený node. Tento způsob provádění frame packingu odpovídá strategii First Fit (FF) [6].

Optimalizace vytvořených zpráv

S téměř stoprocentní jistotou nebudou mít všechny zprávy vytvořené v předchozím kroku velikost rovnající se (nebo alespoň blížící se) velikosti rámce. Proto by při posílání těchto zpráv nebylo využito všechno dostupné místo v rámcích. Z uvedeného důvodu byl zařazen krok optimalizace vytvořených zpráv, který se snaží o spojování zpráv již vytvořených. Cílem je co nejlépe využít dostupnou kapacitu rámců. Na rozdíl od předchozího kroku, tato část neprochází zprávy odděleně podle periody, nýbrž všechny vytvořené zprávy (je připuštěno i spojování zpráv s rozdílnou periodou). Maximalizací využití rámců je tedy minimalizován počet zpráv, které bude později nutné rozvrhnout. Pokud se vyskytnou zprávy s rozdílnou periodou, které lze vzhledem k jejich časovým omezením a velikosti spojit, dojde sice k situaci, kdy se

signály obsažené v jedné původní zprávě budou posílat častěji, než je nutné, a budou tak zabírat více přenosového pásma; avšak druhá původní zpráva by stejně musela být přenášena s definovanou periodou. Tím už je zabrán rámec a vzhledem k velikosti zprávy by nebyl plně využit. Z toho vyplývá, že i když budeme posílat více dat, tak ušetříme právě tím, že se zbavíme rámců potřebných pro jednu původní zprávu. Ve výsledku to znamená, že bude potřeba méně zabraných slotů, čímž se zkrátí délka statického segmentu cyklu.

Vytvoření optimalizovaných zpráv je zajištěno postupným procházením všech zpráv. U každé zjišťujeme, zda by nešla spojit se zprávou jinou za dodržení všech dříve uvedených podmínek. Bylo rozhodnuto, že zprávy budou seřazeny od té nejméně časté (tedy s největší periodou) po tu nejčastější. Takový způsob si klade za cíl omezit na nejnižší možnou míru plýtvání dostupnou přenosovou kapacitou. Tím lze dosáhnout nejvyšší možné efektivity využití přenosového pásma (to vede ke snížení počtu slotů potřebných pro rozvrh komunikace). Tento způsob odpovídá strategii First Fit Decreasing (FFD) [6].

Asymptotická časová složitost optimalizace zpráv je $O(m^2)$, kde m vyjadřuje počet zpráv vypočtených v předchozím kroku pro aktuální node.

Tvorba rozvrhu

V této části algoritmu se dostáváme k samotnému vypočtení harmonogramu komunikace pro aktuální node. Optimalizované zprávy jsou seřazeny od nejnižší periody (tedy od těch optimalizovaných zpráv, které jsou posílány nejčastěji) po optimalizované zprávy s nejvyšší periodou. To garantuje, že pokud jsou přiřazovány optimalizované zprávy do slotů, stačí projít pozice slotu, kde může být umístěn první výskyt optimalizované zprávy v rámci její periody. Pokud je v této části slotu nalezeno vhodné místo, je zaručeno, že i pro další výskyty optimalizované zprávy v rámci hyperperiody bude ve slotu místo.

Pro každou optimalizovanou zprávu je hledán slot, který ještě obsahuje dostatek volného místa (volných rámců v rámci hyperperiody) a splňuje požadavky na časová omezení (má v časovém okně definovaném release timem a deadlinem optimalizované zprávy volný rámec).

Asymptotická časová složitost výpočtu rozvrhu pro každý node je $O(\max(m' \cdot i \cdot c, m' \cdot \log(m')))$, kde m' je počet optimalizovaných zpráv pro aktuální node vypočtených v předchozím kroku; i je počet slotů přiřazených danému nodu a c je počet cyklů v hyperperiodě. Složitost $m' \cdot \log(m')$ zde vyjadřuje úvodní seřazení signálů do požadované posloupnosti.

2.2.3. Příklad běhu vlastního algoritmu

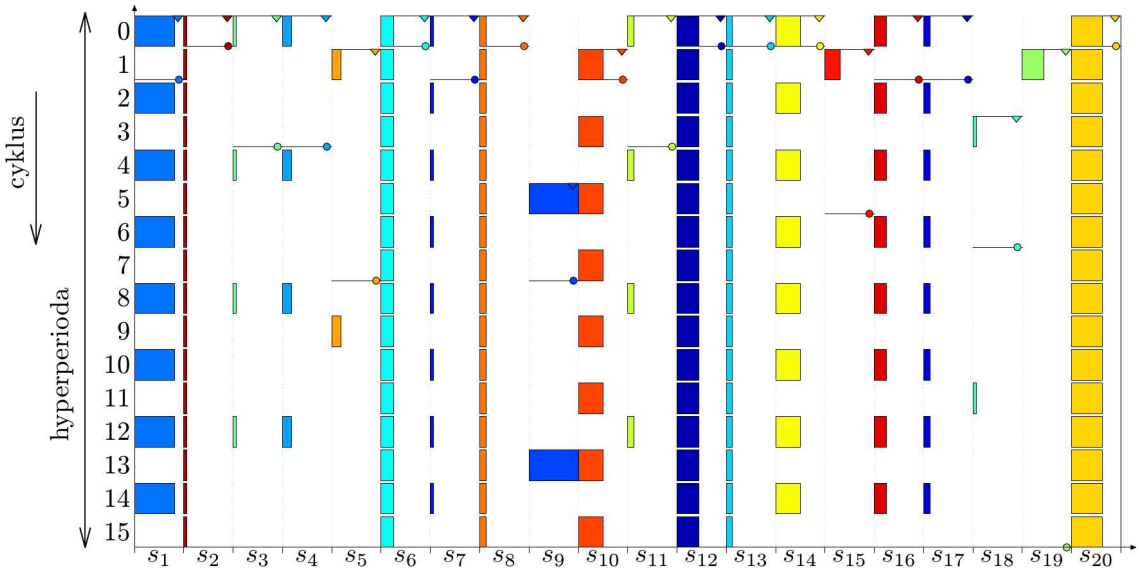
Pro názorné předvedení funkce algoritmu byla připravena množina signálů pro jedno zařízení. Na uvedených signálech jsou vidět postupné výsledky mezi jednotlivými částmi algoritmu.

Vstupní signály jsou popsány Tabulkou 2, ve které jsou všechny parametry vstupních signálů. Tytéž parametry jsou znázorněny graficky na Obrázku 14. Každý sloupec tohoto obrázku reprezentuje jeden signál. Perioda signálu je vyjádřena opakováním signálu ve sloupci. Velikost signálu (množství v něm přenášených bitů) je znázorněna šířkou instance signálu. Release time pro první výskyt signálu je zobrazen vodorovnou čarou s trojúhelníčkem v barvě signálu na její pravé straně v příslušném sloupci. Podobně jsou označeny deadline, s tím rozdílem, že na konci vodorovné čáry je místo trojúhelníku použito kolečko. Signály přesně v tomto stavu jsou výsledkem rozdělení signálů k nodům – parametry signálů jsou již uvedeny v hodnotách, které jsou vyžadovány pro následné zpracování. Pro názornost v ukázce dále pracujeme jen s jedním nodem. Signály v této podobě jsou předány ke zpracování části algoritmu, která sdružuje signály do zpráv.

	C_i	T_i	$\tilde{O}_i$	$\tilde{D}_i$
s_1	26	2	0	2
s_2	2	1	0	1
s_3	2	4	0	11
s_4	6	4	0	13
s_5	6	8	1	8
s_6	8	1	0	1
s_7	2	2	0	2
s_8	4	1	0	1
s_9	32	8	5	8
s_{10}	16	2	1	2

	C_i	T_i	$\tilde{O}_i$	$\tilde{D}_i$
s_{11}	4	4	0	9
s_{12}	14	1	0	1
s_{13}	4	1	0	1
s_{14}	16	2	0	1
s_{15}	10	16	1	6
s_{16}	8	2	0	2
s_{17}	4	2	0	2
s_{18}	2	8	3	7
s_{19}	14	16	1	16
s_{20}	20	1	0	1

Tabulka 2 - Seznam signálů na vstupu algoritmu



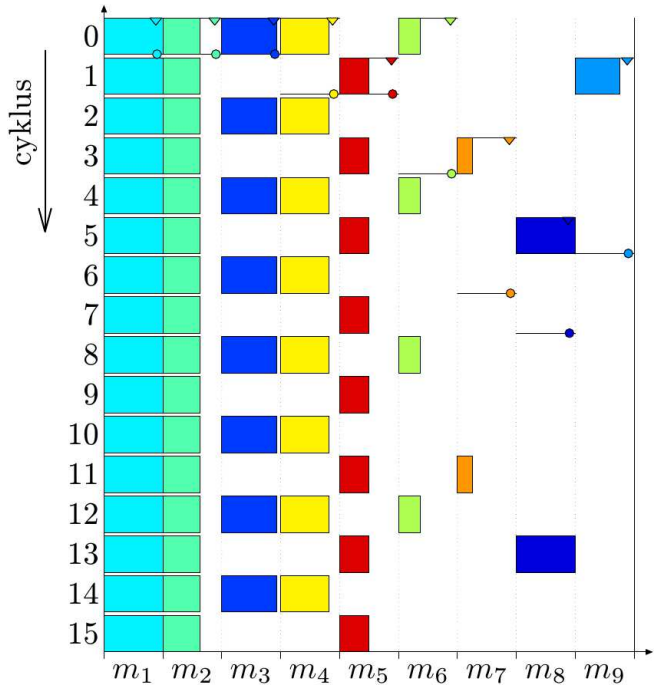
Obrázek 14 - Grafické znázornění signálů vstupujících do algoritmu

Po průchodu výše uvedených signálů frame packingem jsou vytvořené zprávy uvedeny v Tabulce 3 a zobrazeny na Obrázku 15, kde nyní ve sloupcích nejsou zobrazeny signály, nýbrž vytvořené zprávy. Všechny parametry zpráv jsou uvedeny stejným způsobem jako je tomu výše u signálů. Z Tabulky 3 je vidět, že byly sloučeny například signály s_3, s_4 a s_{11} do zprávy m_6. Možnost sloučení uvedených signálů je evidentní z Obrázku 14. U signálů s_5 a s_{18} nebylo sloučení na první pohled úplně evidentní, neboť z nich vytvořená zpráva m_7 musí respektovat časová omezení obou, která jsou odlišná. Zpráva m_8 je vytvořena jen ze signálu s_9, protože signál už dosáhl maximální možné velikosti a jeho sloučení s jakýmkoliv dalším signálem není přípustné. Tyto

vytvořené zprávy jsou dále předány k optimalizaci zpráv.

	C_i	T_i	$\tilde{O}_i$	$\tilde{D}_i$	signály
m_1	32	1	0	1	s_2, s_6, s_8, s_{12}, s_{13}
m_2	20	1	0	1	s_{20}
m_3	30	2	0	1	s_7, s_{14}, s_{16}, s_{17}
m_4	26	2	0	2	s_1
m_5	16	2	1	2	s_{10}
m_6	12	4	0	4	s_3, s_4, s_{11}
m_7	8	8	3	7	s_5, s_{18}
m_8	32	8	5	8	s_9
m_9	32	16	1	6	s_{15}, s_{19}

Tabulka 3 - Zprávy vytvořené ze signálů



Obrázek 15 - Grafická reprezentace vytvořených zpráv

Výsledkem optimalizace výše uvedených zpráv jsou optimalizované zprávy uvedené v Tabulce 4 a graficky zobrazené na Obrázku 16. Díky tomuto kroku byly dále sloučeny zprávy m_5 a m_6 do optimalizované zprávy m_5' a zprávy m_7 a m_9 do optimalizované zprávy m_6'. Tedy dojde ke sloučení zpráv napříč různými periodami. Tento výsledek je předán k samotnému rozvrhnutí.

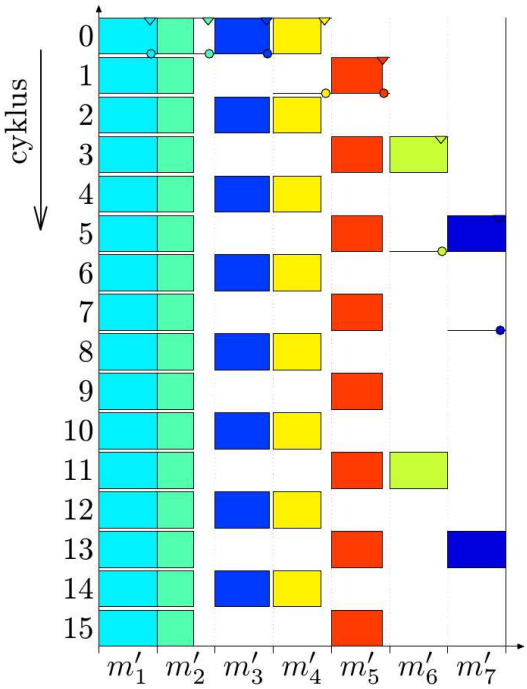
	C_i	T_i	$\tilde{O}_i$	$\tilde{D}_i$	zprávy	signály
m'_1	32	1	0	1	m_1	s_2, s_6, s_8, s_{12}, s_{13}
m'_2	20	1	0	1	m_2	s_{20}
m'_3	30	2	0	1	m_3	s_7, s_{14}, s_{16}, s_{17}
m'_4	26	2	0	2	m_4	s_1
m'_5	28	2	1	2	m_5, m_6	s_3, s_4, s_{10}, s_{11}
m'_6	32	8	3	6	m_7, m_9	s_5, s_{15}, s_{18}, s_{19}
m'_7	32	8	5	8	m_8	s_9

Tabulka 4 - Optimalizované zprávy vytvořené ze zpráv

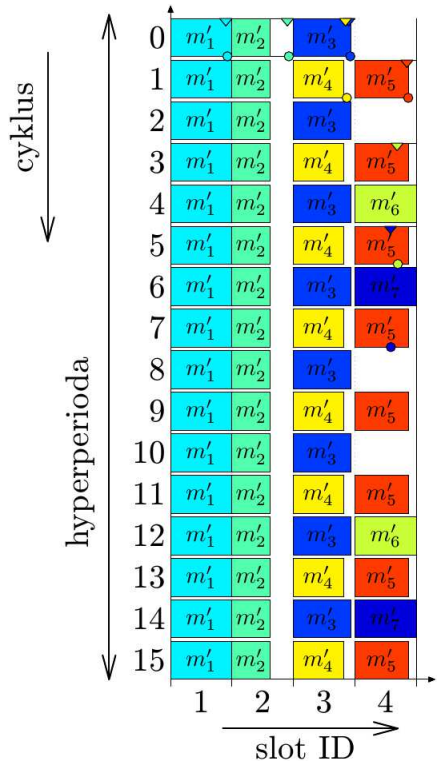
Výsledek rozvrhování optimalizovaných zpráv je zobrazen na Obrázku 17. Uvedený obrázek oproti předchozím nemá ve sloupcích zobrazeny jednotlivé signály, či zprávy, ale každý sloupec představuje slot na síti FlexRay. Z obrázku je patrné, že optimalizované zprávy, které lze rozvrhnout do jediného slotu jsou takovým způsobem naplánovány.

Tento krok je tedy závěrečným krokem algoritmu a jako takový je jeho výstupem. Vypočtený harmonogram komunikace pro jednotlivé nody se pouze spojí bez dalších výpočtů.

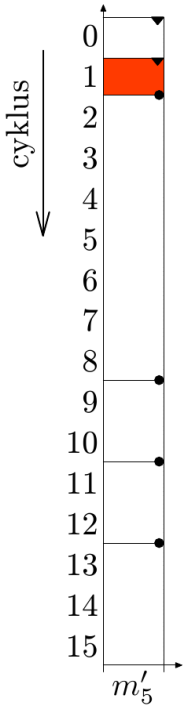
Pro detailnější představu, jak se upravují časová omezení pro zprávu během jejího vytváření, jsou na Obrázku 18 zobrazena časová omezení všech signálů sloučených do optimalizované zprávy m'_5. Vytvořená zpráva musí převzít nejpozdější release time a nejdřívější deadline, aby splňovala všechna časová omezení signálů ve zprávě obsažených.



Obrázek 16 - Grafické zobrazení vytvořených optimalizovaných zpráv



Obrázek 17 - Grafická reprezentace celkového vygenerovaného rozvrhu



Obrázek 18 - Zpřísňování časových omezení během frame packingu a optimalizace zpráv

2.3. Vylepšení algoritmu

Vzhledem k výsledkům implementovaného algoritmu bylo rozhodnuto, že bude tento postup rozvrhování signálů do statického segmentu sítě FlexRay dále vylepšován. Bylo zvoleno, že bude snaha přesunout se od heuristiky k optimálnímu řešení, pokud to bude možné.

2.3.1. Rozdělení signálů k nodům

Jelikož se v této části algoritmu neprovádí žádný výpočet, ale jedná se pouze o předzpracování signálů (převedení vstupních hodnot release time a deadline na hodnoty, se kterými algoritmus pracuje) a rozdělení vstupních dat do skupin dle nodů, které jsou zapotřebí pro navržený algoritmus a počítání bitů, které je nutné přenést za hyperperiodu pro každý node, není třeba tento krok dále zdokonalovat. Počet bitů, které jsou potřebné pro každý node za hyperperiodu, se využívá pro výpočet režie a dalších hodnocení algoritmu.

2.3.2. Frame packing

Prvním rozdílem ve vylepšené verzi frame packingu oproti původní implementaci je úvodní řazení signálů nikoliv podle deadline, ale podle velikosti okna. Velikost okna je čas mezi release time a deadline signálu. Seřazení signálů dle okna zajišťuje, že frame packing bude sdružovat především signály s podobnou velikostí okna (s respektováním časových omezení), takže vytvářené zprávy nebudou omezovány více, než je nezbytně nutné. Dále díky seřazení signálů dle velikosti jejich okna zajišťuje, že signály s nejmenšími velikostmi okna budou sdruženy (pokud to bude možné kvůli časovým omezením) a signály s velkými časovými okny nebudou omezovány signály s malými časovými okny.

Druhou důležitou změnou oproti změně původně popsané je fakt, že signály nejsou přiřazovány do první zprávy, která splňuje časová omezení. Nejprve se ohodnotí všechny zprávy dle toho, jak je vhodné signál do zprávy přidat. Signál je poté vložen do zprávy, do které je jeho přiřazení nejvhodnější. Ohodnocení každé zprávy bere v úvahu rozdílnost release time a deadline mezi signálem a zprávou, a dále také výslednou velikost zprávy. Ohodnocení zprávy m pro přidání signálu s je vypočteno následujícím způsobem:

$$ohodnoceníFP_m = 10 * \left(\left|\tilde{O}_s - \tilde{O}_m\right| + \left|\tilde{D}_s - \tilde{D}_m\right|\right) + \left(W - C_m\right)$$

Byla definována mez udávající minimální ohodnocení zprávy, kdy má ještě dojít k přidání právě zpracovávaného signálu do zprávy a kdy již nikoliv. Pokud ani jedna zpráva nedosahuje nastavené meze, nebude signál přidán do žádné existující zprávy, ale bude pro něj vytvořena zpráva nová. Tato mez zabraňuje vložení signálu do zprávy, která má velmi rozdílná časová omezení, tím pádem by byla značně omezena možnost dále optimalizovat vytvořené zprávy v dalším kroku.

Výsledkem tohoto vylepšení frame packingu je varianta, která využívá strategii Best Fit Decreasing (BFD) [6]. Uvedený postup odpovídá Pseudokódu 2.

Asymptotická časová složitost frame packingu zůstala stejná jako v původní verzi, tedy $O(s^2)$, kde s je počet signálů pro právě procházený node.

```
Vstup: signály(k) = [s(1), … s(#signály(k))]
Výstup: zprávy(k) = [m(1), …, m(#zprávy(k))]

frame_packing_mez = 20;
for (p=1 ... #signály(k))
    ohodnocení = zeros(1,#zprávy(k));
    for (q=1 ... #zprávy(k))
        ohodnocení(q) = ohodnoť přiřazení s(p) do m(q);
    end;
    if (max(ohodnocení(q)) > frame_packing_mez)
        [max_ohodnocení, index] = max(ohodnocení);
        m(index) = přiřaď s(p) do m(index);
    else
        vytvoř novou zprávu jen pro s(p);
        #zprávy(k) += 1;
    end;
end;
zprávy(k) = m;
```

Pseudokód 2 - Vylepšená verze frame packingu

Dynamické programování

Analýzou výsledků tohoto algoritmu byl frame packing identifikován jako část algoritmu, která produkuje největší režii. Kvůli tomuto zjištění byly hledány jiné způsoby, jak provádět frame packing tak, aby bylo docíleno lepších výsledků. S ohledem na časová omezení a prioritu, která je definována co možná nejmenším zpřísňováním časového omezení v tomto kroku[2], bylo zvoleno dynamické programování. To má největší potenciál vrátit výsledky v porovnatelném čase s ostatními částmi algoritmu. Stejně jako v předchozí implementaci frame packingu i zde je sdružování signálů do zpráv realizováno odděleně po periodách.

První varianty frame packingu pomocí dynamického programování vytvářely vždy jednu optimální zprávu ze signálů konkrétní periodicity. Poté byly signály, ze kterých se zpráva sestávala, odebrány a znovu se hledala optimální zpráva ze zbylých signálů. Tento postup se opakoval, dokud pro danou periodicitu zbývaly signály.

[2] To by nejpravděpodobněji vedlo k málo efektivnímu rozvrhu komunikace.

Následně byl spuštěn tentýž proces pro další periody. Tento postup byl zvolen pro ověření škálovatelnosti řešení pro velké množiny signálů, neboť nalezení kombinace optimálních zpráv pro signály jedné periodicity najednou je o několik řádů složitější problém. Ve chvíli, kdy byly vzaty v úvahu release time a deadline při slučování signálů do zpráv, se výpočet frame packingu na větších instancích množin signálů pomocí dynamického programování stal v porovnání s ostatními částmi algoritmu velmi pomalým. Popsaná metoda postupného vytváření jednotlivých optimálních zpráv však nepředstavuje celkově optimální řešení. Popsaný způsob frame packingu pomocí dynamického programování je dále označován DP.

Vzhledem k výpočetnímu času při využití dynamického programování pro frame packing bylo rozhodnuto, že možnost nalézt optimální kombinaci přiřazení signálů do zpráv nebude dále rozvíjena, jelikož přechod k optimálnímu řešení by vedl k neúměrně velkému prodloužení výpočetních časů. Díky tomu by bylo prakticky nemožné zpracovávat větší množiny signálů.

2.3.3. Optimalizace zpráv

Stejně jako u frame packingu, nejsou nyní zprávy optimalizovány s nalezením jejich první vhodné dvojice. Nejprve jsou pro každou zprávu ohodnoceny ostatní zprávy podle toho, jak jsou vhodné pro spojení s právě aktuální zprávou. Toto ohodnocení bere v úvahu rozdílnost period jednotlivých zpráv, rozdíly v release time a deadline potenciálně slučovaných zpráv a celkovou velikost potenciální nové zprávy.

$$ohodnoceníOZ_{m'} = 100 * \left(\left|T_{m1} - T_{m2}\right|\right) + 10 * \left(\left|\tilde{O}_{m1} - \tilde{O}_{m2}\right| + \left|\tilde{D}_{m1} - \tilde{D}_{m2}\right|\right) + \left(W - C_{m1} - C_{m2}\right)$$

Stejně jako u frame packingu, i zde je zavedena mez, která určuje ohodnocení, pod které se už zprávy slučovat nemají. Pokud pro aktuální zprávu žádná jiná zpráva nedosáhne této meze, nebude se zpráva v tomto kroku s žádnou jinou slučovat a to i přesto, že potenciálně existují takové zprávy, které by bylo možné s aktuální zprávou sloučit. Díky ohodnocení zpráv je zaručeno, že se nejprve budou slučovat zprávy s podobnými časovými omezeními a podobnou periodou. Zavedení meze optimalizace zpráv zajistí, že nebudou slučovány zprávy s výrazně odlišnými periodami, což by značně zvyšovalo režii algoritmu a mohlo by zabránit pozdějšímu sloučení vhodnějších zpráv.

```
Vstup: zprávy(k) = [m(1), …, m(#zprávy(k))]
Výstup: optzprávy(k) = [m’(1), …, m’(#optzprávy(k))]

optimalizace_mez = 20;
for (p=1 ... #zprávy(k))
    m’(p) = m(p);
    ohodnocení = zeros(1,#zprávy(k));
    while (max(ohodnocení) > optimalizace_mez)
        for (q=1...#zprávy(k))
            ohodnocení(q) = ohodnoť spojení m’(p) a m(q);
        end;
        if (max(ohodnocení) > optimalizace_mez)
            [max_ohodnocení, index] = max(ohodnocení);
            m’(p) = spoj m’(p) a m(index);
        end;
    end;
end;
optzprávy(k) = m’;
```

Pseudokód 3 - Vylepšená verze optimalizace zpráv

Uvedená varianta pro optimalizaci zpráv odpovídá strategii Best Fit (BF) [6]. Výsledkem vylepšení optimalizace zpráv je implementace odpovídající Pseudokódu 3.

Asymptotická časová složitost se vzhledem k uvedeným úpravám oproti původní variantě zvýšila na $O(m^3)$, kde m je počet zpráv vytvořených frame packingem pro aktuální node.

2.3.4. Tvorba rozvrhu

Díky předchozím krokům algoritmu se značně snížilo množství elementů, které je potřeba rozvrhnout (počet optimalizovaných zpráv je výrazně nižší, než počet signálů jednotlivých nodů). Díky tomu přichází v úvahu využití některého z optimálních řešení k rozvržení optimalizovaných zpráv. V této fázi je potřeba nepreemptivně rozvrhnout optimalizované zprávy do co nejmenšího počtu slotů (v terminologii rozvrhování sloty představují identické procesory). Optimalizované zprávy mohou být popsány jako jednotkové harmonické úlohy s celočíselnými hodnotami release time a deadline bez

možnosti variability v periodě (jitter).

ILP formulace

Výše popsaný problém je řešitelný pomocí následujících rovnic celočíselného lineárního programování (ILP):

x_i^j j-té možné umístění optimalizované zprávy i od jejího release time (j-tý cyklus od release time); $x \in \{0,1\}$,

${}_t y_i^j$ t-té opakování (optimalizovaná zpráva je v hyperperiodě opakována kvůli své periodě) optimalizované zprávy i pokud by první výskyt optimalizované zprávy i byl v j-tém cyklu od release time; $y \in \{0,1\}$,

z_c počet optimalizovaných zpráv, které se mají přenášet v cyklu c; $z \in \{0,1,...,\#slotů\, nodu\, k\}$,

$\sum_{j=1}^{velikost\, okna\ \ optzpravy\ \ i} x_i^j = 1$ je povolen právě jeden výskyt pro každou optimalizovanou zprávu i v rámci jejího časového okna,

$\sum_{t=1}^{|t_i|} {}_t y_i^j - |t_i| \cdot x_i^j = 0$ součet všech výskytů optimalizované zprávy i je právě takový jaký má být počet opakování zprávy v rámci hyperperiody,

$\sum {}_t y_i^j - z_c \leq 0$ počet všech výskytů optimalizovaných zpráv v cyklu c může být nejvýše roven maximálnímu počtu slotů,

$\sum_{c=1}^{cyklů\, v\, hyperperiodě} z_c = \sum_{i=1}^{\#optzpráv} |t_i|$ garantuje, že je alokován stejný počet pozic jako je celkový počet všech výskytů zpráv.

Není potřeba dále zajišťovat požadavek pro přenos instancí každé optimalizované zprávy (perioda v rámci hyperperiody) vždy ve stejném slotu, protože toho lze dosáhnout i v další fázi přiřazovaní optimalizovaných zpráv do konkrétních čísel slotů a to takovým způsobem, že nejprve budou přiřazeny optimalizované zprávy s největší periodou a optimalizované zprávy s nejnižší periodou až jako poslední.

Spolupráce ILP a heuristiky

V případě, že bychom využívali jen celočíselné lineární programování, je nutné začít s nějakým počtem přípustných slotů. Aby nebylo nutné zahájit proces na nějakém náhodně vybraném maximálním počtu slotů, je využita předchozí verze rozvrhování optimalizovaných zpráv. Tím je získán počet slotů, do kterých je v každém případě možné optimalizované zprávy rozvrhnout. Počet slotů získaný tímto způsobem je potom použit ve variantě rozvrhování pomocí celočíselného lineárního programování a to pro počáteční určení maximálního počtu použitých slotů. ILP je následně spouštěno v cyklu, ve kterém je stále snižován počet slotů, jež je možné využít pro rozvrhování. Celočíselné lineární programování je spouštěno až do doby, kdy vrací proveditelný rozvrh. Ve chvíli, kdy ILP vrátí neproveditelný rozvrh, se bere předchozí poslední proveditelný rozvrh. Díky postupnému snižování počtu použitých slotů využívá tento rozvrh jejich minimální možný počet. Uvedená spolupráce ILP a heuristiky je znázorněna na Pseudokódu 4.

```
Vstup: optzprávy(k) = [m(1), …, m(#optzprávy(k))]
Výstup: rozvrh(k)

rovrh_heuristika = vytvoř rozvrh(optzprávy(k));
UB = rozvrh_heuristika.počet_slotů;
vytvoř matice(optzprávy(k));
nastav maximum slotů na UB;
while (ILP vrací validní rozvrh)
        xmin = spusť ILP na maticích;
        UB = UB - 1;
        nastav maximum slotů na UB;
end;
rozvrh(k) = extrahuj rozvrh z validního xmin;
```

Pseudokód 4 - Spolupráce heuristiky a ILP během tvorby rozvrhu

Odlišné pořadí rozvrhovaných optimalizovaných zpráv

Jednou z dalších variant rozvrhování optimalizovaných zpráv je jejich jiné úvodní seřazení, než podle periody – tedy způsobem jakým je to provedeno v původní variantě rozvrhování. Jiné seřazení si však žádá i částečnou úpravu algoritmu tvorby rozvrhu. Postupné rozvrhování optimalizovaných zpráv podle periody totiž zajišťovalo, že pokud se ve slotu našlo volné místo pro výskyt optimalizované zprávy v první periodě, bylo ve slotu vždy garantováno místo i pro ostatní výskyty optimalizované zprávy v rámci hyperperiody.

Jako první by měly být zařazovány optimalizované zprávy, jež mají nejméně možností, na které místo v rozvrhu je lze zařadit. To odpovídá seřazení optimalizovaných zpráv podle jejich časových oken od nejmenšího po největší. U takto seřazených zpráv je dále vhodné optimalizované zprávy se stejným časovým oknem řadit dle jejich release time (pro stejná časová okna je to identické se seřazením podle deadline). V případě, kdy by byly vstupem algoritmu množiny signálů, které by neměly nastaveny release time a deadline[3], je seřazení dle velikosti časového okna a release time rovnocenné se řazením podle periody. To bylo provedeno v původní variantě rozvrhování. S ohledem na nutnost procházet všechna opakování optimalizovaných zpráv v rámci hyperperiody při hledání vhodného místa ve slotu, je tato varianta pomalejší než původní verze rozvrhování optimalizovaných zpráv. Tato varianta rozvrhování optimalizovaných zpráv je znázorněna Pseudokódem 5.

[3] Release time roven nule a deadline roven periodě pro všechny signály.

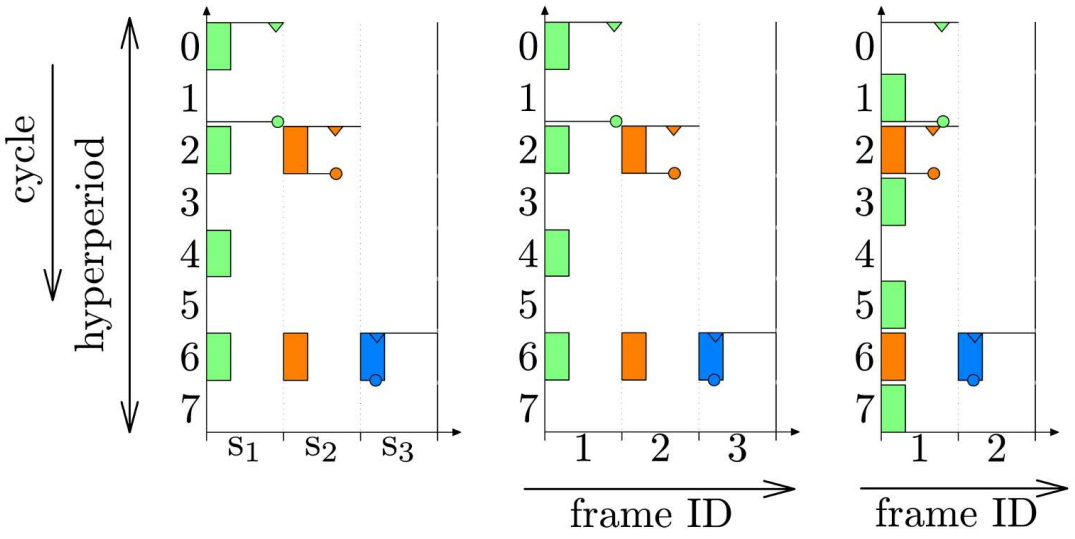
```
Vstup: optzprávy(k) = [m(1), …, m(#optzprávy(k))]
Výstup: rozvrh(k)

seřaď optzprávy(k) podle release time;
seřaď optzprávy(k) podle časového okna;
#slotů(k) = 0;
for (p=1 ... #optzpráv(k))
      for (q=1 ... #slotů(k))
            for (c=m'(p).release ... m'(p).deadline-1)
                  if (m'(p) lze přenést ve slotu q
                                          se začátkem v cyklu c)
                        přiřaď m'(p) do slotu q;
                        nastav první přenos m'(p) v cyklu c;
                        pokračuj až dalším p;
                  end;
            end;
      end;
      if (m'(p) nelze přenést v žádném aktuálním slotu)
            #slotů(k) += 1;
            přiřaď m'(p) do slotu #slotů(k);
            nastav první přenos m'(p) v cyklu m'(p).release;
      end;
end;
```

Pseudokód 5 - Tvorba rozvrhu z optimalizovaných zpráv

3. Výsledky algoritmu

Porovnáváni jednotlivých verzí částí algoritmu navzájem není možné, jelikož samotný výsledek jednotlivé části nelze objektivně ohodnotit. Jako příklad uveďme stav, kdy výsledkem sdružování signálů do zpráv je vznik určitého počtu zpráv. Není však zaručeno, že menší počet zpráv je lepší výsledek. Pokud totiž budou mít výsledné zprávy velmi malé časové okno, ve kterém mohou být posílány, může dojít k situaci, že pro výsledný rozvrh bude potřeba více slotů, než kdyby bylo zpráv více. Potom by měly větší časová okna, a tím pádem by byly snáze rozvrhnutelné. Stejná situace je u optimalizace zpráv.

Byly testovány všechny smysluplné hodnoty pro zavedené meze pro frame packing a optimalizaci zpráv a jejich kombinace na všech testovacích množinách signálů. Z těchto testů vyplynulo, že není potřeba pro každou množinu signálů testovat všechny možnosti a kombinace mezí. Určitá zvolená hodnota meze pro frame packing a jedna mez pro optimalizaci zpráv dává nejlepší výsledky pro všechny množiny signálů. Porovnání jednotlivých verzí algoritmů je zobrazeno v Tabulce 5.



Obrázek 19 - Příklad problému pro heuristiku a vyřešení ILP

Rozvrhování optimalizovaných zpráv pomocí ILP řeší některé problémy, se kterými si původní implementovaná heuristika nedokázala poradit. Příkladem uvedené situace jsou signály znázorněné na Obrázku 19. V rámci testování verzí algoritmu bylo zjištěno, že výsledky rozvrhování optimalizovaných zpráv pouze pomocí heuristiky

generuje rozvrhy se stejným počtem potřebných slotů, jako při využití celočíselného programování. Z toho vyplývá, že použitá heuristika je velmi úspěšná a není zapotřebí využívat optimální metodu ILP, která ke stejným výsledkům potřebuje nepoměrně delší čas. Vzhledem k tomuto zjištění nebyla v porovnávání výsledků verzí algoritmu použita varianta s rozvrhováním optimalizovaných zpráv pomocí celočíselného lineárního programování.

Hlavního cíle, kterým je minimalizace počtu využitých slotů v rozvrhu komunikace pro každou množinu signálů, se podařilo pomocí vylepšených verzí některých částí algoritmů dosáhnout. Dále bylo ověřeno několik alternativ, které oproti očekávání nesnižovaly počet slotů ve výsledném rozvrhu, ale doba potřebná k jejich výpočtu byla většinou značně delší. Dosažené výsledky pomocí nejlepší varianty vylepšené verze algoritmu však nejsou oproti původní verzi nijak výrazné. Podrobnou analýzou bylo zjištěno, že ve vylepšené verzi algoritmu se počet potřebných slotů pro rozvrh komunikace rovná teoretickému minimálnímu počtu slotů, do kterých je možné rozvrhnout optimalizované zprávy v naprosté většině množin signálů. U některých množin signálů dokonce výsledný rozvrh vylepšeného algoritmu potřeboval právě tolik slotů, kolik bylo jejich naprosté teoretické minimum na vstupu do algoritmu. Naprosté teoretické minimum slotů $\min_{abs}$ potřebných pro node, ve kterých lze přenést všechny signály, je vypočteno následujícím způsobem:

$$\min_{abs} = ceil\left(\frac{\sum\left(C_i \cdot \left(\frac{H}{T_i}\right)\right)}{W \cdot H}\right)$$

kde H určuje počet cyklů v hyperperiodě. Při tomto výpočtu neuvažujeme release time ani deadline jednotlivých signálů. Tento údaj určuje absolutně minimální počet slotů, do kterých by teoreticky šlo rozvrhnout vstupní signály. Po provedení sdružování signálů do zpráv a optimalizaci zpráv, je třeba výpočet aktuálního teoretického minima slotů upravit. Povolujeme totiž přenos maximálně jedné optimalizované zprávy v jednom rámci.

$$\min'_{abs} = ceil\left(\frac{\sum \frac{H}{T_i}}{H}\right)$$

kde hodnota T_i je perioda optimalizované zprávy i. Uvedené minimum je vždy větší, či stejné, než předchozí naprosté teoretické minimum slotů potřebných pro node $\min_{abs}$. Údaj $\min'_{abs}$ tedy udává nové teoretické minimum slotů potřebných pro vytvoření rozvrhu. Ani v tomto údaji se neuvažují release time a deadline.

Vylepšená verze algoritmu je schopna ve velmi krátké době rozvrhnout velké množiny signálů. Největší instance testovacích množin signálů obsahovaly jeden node s až 3000 signály. Tato vylepšená verze algoritmu byla schopna tyto množiny signálů rozvrhnout v několika vteřinách.

3.1. Výkonnostní testy

Aby bylo možno porovnat rychlost a úspěšnost výsledků jednotlivých verzí algoritmu, byly vybrány testovací množiny, na kterých byl algoritmus v několika modifikacích spuštěn. Pro každou skupinu množin signálů bylo vygenerováno 10 instancí podle stejných parametrů. Výsledek pro každou skupinu je tak tedy průměrem výsledků přes všechny její instance. Jelikož algoritmus je deterministický, jeho několikanásobné spouštění na každé skupině dává identické výsledky. Skupiny signálů mají následující parametry:

- Set1 – ani release time ani deadline nejsou nastaveny,
- Set2 – deadline roven konci periody, release time náhodně posunut do prvních pěti cyklů,
- Set3 – deadline nastaven na dvě třetiny periody, release time náhodně posunut do prvních pěti cyklů,
- Set4 – podobná konfigurace jako Set2,
- Set5 - deadline roven konci periody, release time náhodně posunut do prvních pěti cyklů,

- Set6 – stejné parametry pro generování signálů, jen vyšší zatížení sběrnice,
- Set7 – ani release ani deadline nejsou nastaveny; pro jednotlivé nody je definováno jen velmi málo signálů,
- 1ECU_500 – několik náhodně definovaných deadline, release time náhodně posunut do prvních pěti cyklů,
- 1ECU_1000 – stejná konfigurace jako 1ECU_500; větší počet signálů,
- 1ECU_3000 – stejná konfigurace jako 1ECU_1000; větší počet signálů; nastavena větší velikost slotu než u ostatních množin signálů.

Skupina signálů				Původní		Vylepšená		DP		Okno		Zprávy		Meze	
	#signálů	#nodů	min #ID	#ID	t [s]	#ID	t [s]	#ID	t [s]	#ID	t [s]	#ID	t [s]	#ID	t [s]
Set1	622	3	21,3	24,2	0,52	23,8	0,62	24,2	1,32	23,8	0,61	24	1,39	23,8	15,79
Set2	622	3	21,3	24,2	0,53	23,8	0,62	24,2	1,55	23,8	0,62	24	1,41	23,8	15,88
Set3	622	3	21,3	24,2	0,53	23,8	0,60	24,2	1,47	23,8	0,62	24	1,52	23,8	15,84
Set4	622	3	21,3	24,2	0,53	23,8	0,66	24,2	1,47	23,8	0,62	24	1,36	23,8	15,93
Set5	566,7	6	12,1	17,7	0,47	17,6	0,64	17,7	1,00	17,7	0,57	17,7	1,07	17,6	14,77
Set6	784,4	6	28,3	32,5	0,68	31,9	1,04	32,5	1,57	32	0,78	31,9	1,65	31,9	20,46
Set7	850,7	23	27,8	42,2	0,71	41,3	0,95	43,6	1,15	43,4	0,83	43,4	1,53	41,3	24,97
1ECU_500	467,3	1	20,1	20,8	0,42	20,4	0,47	21,2	5,52	20,7	0,46	20,4	1,17	20,4	11,01
1ECU_1000	973	1	41,5	42,1	0,91	41,9	0,99	43,7	11,18	42,2	0,96	41,9	3,30	41,9	24,51
1ECU_3000	2704	1	28,9	29,7	2,60	29,2	2,96	30,8	62,27	29,6	2,87	29,2	9,47	29,2	71,75

Tabulka 5 - Porovnání výsledků verzí algoritmů

V Tabulce 5 jsou zobrazeny výsledky běhu algoritmu v několika modifikacích na popsaných množinách signálů. V každém řádku tabulky jsou uvedeny výsledky pro jeden typ množiny signálů vypočtené několika rozdílnými variantami algoritmu. Součástí této tabulky jsou další popisné údaje k množinám signálů. Je zde uveden celkový počet signálů; počet nodů, mezi které jsou signály rozděleny; a naprosté teoretické minimum slotů, které jsou potřeba pro přenos těchto signálů ve statickém segmentu na síti FlexRay. Pro každou z variant algoritmu jsou výsledkem dva sloupce. První sloupec udává počet slotů potřebných pro vypočtený rozvrh, ve druhém je celkový čas běhu algoritmu pro danou množinu signálů.

	Původní	Vylepšená	DP	Okno	Zprávy	Meze
FP	FF	BFD	DP	BFD	BFD	BFD
Mez FP	-	opt	-	FP-okno	FP-min	všechny
OZ	FFD	BF	BF	BF	BF	BF
Mez OZ	-	opt	opt	OZ-okno	OZ-min	všechny

Tabulka 6 - Části algoritmu použité v jednotlivých variantách algoritmu

Použité části v jednotlivých variantách algoritmu jsou popsány v Tabulce 6. Část algoritmu rozdělující signály k jednotlivým nodům a část rozvrhující optimalizované zprávy do slotů nejsou v tabulce zobrazeny, neboť je pro všechny varianty algoritmu používána stejná varianta. Zároveň jsou pro jednotlivé varianty algoritmu v tabulce uvedeny meze, které byly v dané variantě využity. Hodnota *opt* představuje mez, která byla zvolena pro všechny množiny signálů jako nejlepší (stejná hodnota pro všechny množiny signálů dává ve všech případech nejlepší výsledky), a jedná se tedy o jednu konstantní hodnotu zvolenou při analýze a vyhodnocování výsledků algoritmu. Meze FP-okno a OZ-okno jsou takové meze, které zamezují spojování signálů či optimalizaci zpráv, pokud by mělo dojít ke zmenšení časového okna u frame packingu, respektive u optimalizace zpráv. Naopak meze FP-min a OZ-min jsou takové meze, jež nekladou žádné překážky frame packingu či optimalizaci zpráv, pokud je výsledek stále validní. Tímto způsobem se snažíme získat co nejmenší počet optimalizovaných zpráv. Speciálním případem je varianta algoritmu *Meze*. Tato varianta nepoužívá jednu přednastavenou mez, ale testuje všechny možné kombinace mezí frame packingu a optimalizace zpráv. Jako výsledek vrací variantu rozvrhu, který dosáhl nejnižšího počtu slotů. Výpočet sdružování signálů do zpráv a optimalizace zpráv ve variantách algoritmu *Okno* a *Meze* jsou částečně modifikovány. Například u varianty *Okno* není potřeba brát v úvahu signály s rozdílnými časovými omezeními, protože v této verzi bylo nastaveno, že se nebudou zmenšovat časová okna. Uvedenou změnu proto respektuje i výpočet ohodnocení.

Z Tabulky 5 je vidět vylepšení výsledků mezi původní a vylepšenou verzí algoritmu. Tabulka 5 také ukazuje, že další experimenty s jednotlivými částmi algoritmu nepřinesly zlepšení, ani zrychlení výpočtu. Využití celočíselného lineárního programování pro umisťování optimalizovaných zpráv do slotů není v tabulce zahrnuto. Při využití ILP nebyl nikdy nalezen rozvrh pro méně slotů, než pro kolik slotů byl

vytvořen harmonogram komunikace pomocí heuristiky. Taktéž je těmito výsledky ověřeno, že striktní minimalizace počtu zpráv ve frame packingu a následné optimalizaci zpráv nevede k rozvrhu, který potřebuje minimum slotů, stejně tak jako velmi důsledné zachovávání velikosti časových oken.

Přesto, že algoritmus je možné paralelizovat, byl ve všech uvedených případech spouštěn na jednom počítači s jedním jednojádrovým procesorem pro jednodušší možnost porovnání doby běhu s již dostupnými algoritmy. Využitým běhovým prostředím byl Matlab 2009b na operačním systému Microsoft Windows XP. Počítač byl vybaven procesorem Intel Core 2 Solo a 1GB operační paměti.

3.2. Srovnání s existujícími metodami

Nejlepší verze vytvořeného algoritmu popsána v této diplomové práci byla v testech schopna rozvrhovat množiny signálů obsahující téměř 3000 signálů pro jeden node za méně než 3 vteřiny. Žádná z doposud dostupných možností rozvrhování statického segmentu sítě FlexRay s použitím všech časových omezení, ve které je uvedena doba výpočtu, se této hodnotě ani nepřibližuje.

Existuje jen velmi málo možností jak rozvrhovat signály se všemi časovými omezeními (release time, deadline a perioda). Rozvrhování respektující tato omezení se začínají objevovat teprve v poslední době.

Přestože už v dnešních luxusních vozech je až 70 ECU, které mezi sebou komunikují a předávají si až 2500 signálů[1], dostupné metody rozvrhování signálů ve statickém segmentu počítají s desítkami, maximálně stovkami signálů. Navíc výpočet rozvrhu komunikace i pro takové množství signálů je časově náročný.

Pomocí heuristiky v [25] byl vypočten rozvrh pro 28 signálů v řádu jednotek minut. V této práci je však uvažováno opakování přenosu signálu pro zajištění spolehlivosti i při výpadcích komunikace. Algoritmus využívající MILP v [27] respektuje všechna časová omezení vstupních signálů. Výpočet rozvrhu pro 196 signálů trvá přibližně hodinu, ale prokázání optimality pro tyto signály trvá déle než jeden den. Stejně jako algoritmus vyvinutý v této diplomové práci, i autoři [22] se rozhodli pro rozvrhování signálů separátně pro každý node. Motivací v uvedeném článku byla

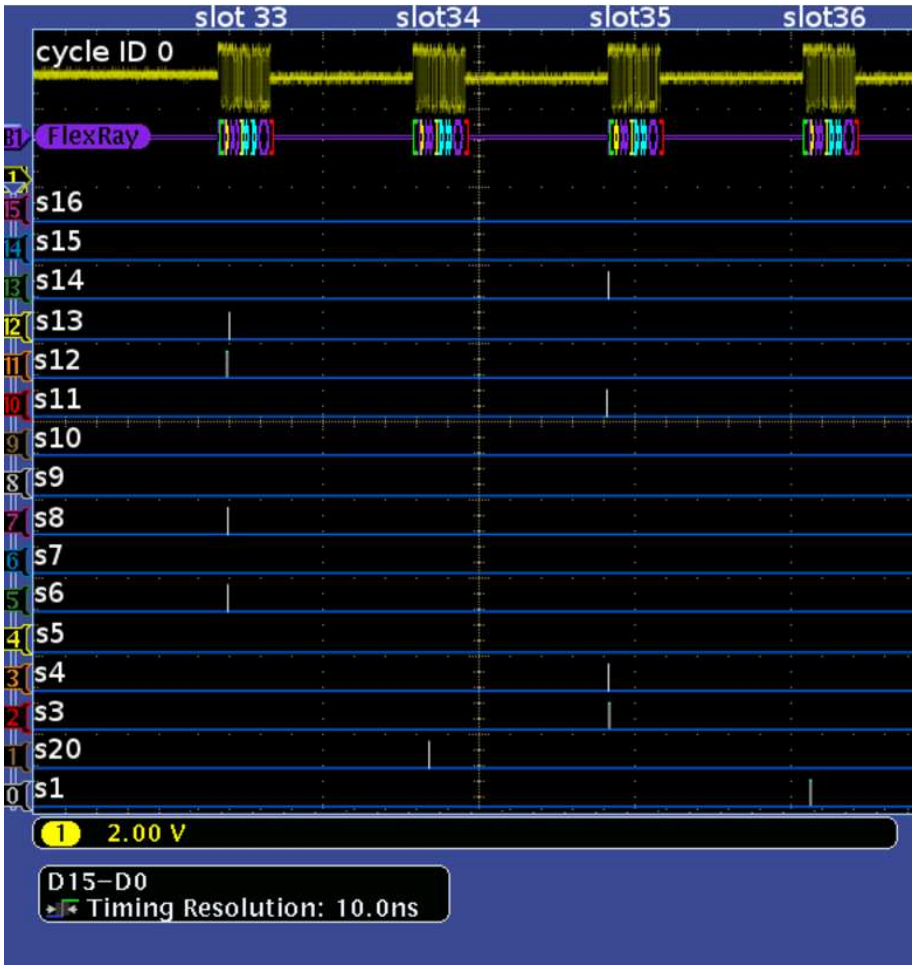
potřeba rozvrhování velkého množství signálů, které jsou v současnosti popsány pro sběrnici CAN.

Srovnání vyvinutého algoritmu na stejných množinách signálů jako využívali autoři existujících metod rozvrhování nebylo provedeno. Jedním z důvodů bylo to, že oslovení autoři již neměli testované množiny signálů k dispozici. Další důvodem byla skutečnost, že většina využívaných způsobů rozvrhování signálů ve statickém segmentu nepočítá s časovými omezeními jednotlivých signálů, a proto by ani při použití stejných množin signálů nebylo porovnání výsledků relevantní.

3.3. Ověření algoritmu na fyzické síti

Pro ověření a demonstraci proveditelnosti vygenerovaných rozvrhů byl realizován praktický test na síti FlexRay. K náhodně vybrané množině signálů byly přidány ukázkové signály z kapitoly 3.2.3. jako signály dalšího nodu a takto vyrobená nová množina signálů byla požita jako vstup algoritmu. Výsledkem algoritmu byl rozvrh na 36 slotů, z nichž poslední 4 sloty byly určeny pro námi přidaný node. Jednotlivá zařízení (založená na mikrokontrolérech Freescale MC9S12XF) na testovací síti FlexRay sestavené na katedře měření (viz Obrázek 20) byla nakonfigurována podle vygenerovaného algoritmu. Ke sběrnici byl připojen osciloskop Tektronix řady MSO/DPO4000B, který je schopen přímo dekódovat provoz na síti FlexRay. Výstup z osciloskopu je zachycen na Obrázku 3, kde je zobrazena část jednoho komunikačního cyklu. Záznam obrazovky osciloskopu ukazuje čtyři statické sloty, do kterých byly rozvrhnuty signály použité pro ukázku fungování algoritmu. V řádcích označených stejně jako ukázkové signály jsou ve chvílích, kdy je přenášen ten signál, podle kterého je řádek pojmenován, zobrazeny jednotkové impulsy. Pokud porovnáme rozvrh vygenerovaný vyvinutým algoritmem z Obrázku 17 (za pomoci Tabulky 4 pro převod optimalizovaných zpráv zpět na jednotlivé signály) a zachycené komunikace na síti FlexRay pomocí osciloskopu na Obrázku 21, zjistíme, že je při reálné komunikaci opravdu dodržen vypočtený rozvrh.

Obrázek 20 - Sestava testovací sítě FlexRay



Obrázek 21 - Komunikace na sběrnici při praktickém testu zobrazena na osciloskopu [11]

4. Popis a požadavky algoritmu

Tato kapitola popisuje jak zprovoznit implementovaný algoritmus na vlastním počítači.

Přiložená verze algoritmu není závislá na operačním systému. Tvorbu rozvrhu lze spustit na všech operačních systémech, které jsou podporované prostředím Matlab.

4.1. Předpoklady pro běh algoritmu

Pro spuštění algoritmu bez generování grafického znázornění výsledného rozvrhu není mimo prostředí Matlab potřeba žádných dalších komponent. Pouze pro možnost využít celočíselného lineárního programování je vyžadováno rozšíření TORSCHE pro Matlab. S tímto rozšířením je totiž mimo jiné dodáván ILP solver GLPK.

Grafická reprezentace výsledných rozvrhů komunikace potřebuje pro svou funkcionalitu několik komponent. Jedná se o dvě rozšíření pro Matlab, TORSCHE a XML Toolbox, které jsou volně dostupné. V této konfiguraci je možné vytvářet grafické rozvrhy pomocí komunikace s webovou službou běžící na serveru Katedry řídící techniky FEL. Pro možnost tvorby grafických výstupů bez nutnosti komunikovat s webovou službou je nutno nainstalovat lokálně program Psgantt.

Pro korektní fungování programu Psgantt musely být zdrojové soubory tohoto programu částečně upraveny. Psgantt v upravené verzi je proto přiložen k elektronické verzi této diplomové práce. Provedená úprava sestává jednak z malé změny generování šipek (jejich velikost je v programu pevně definována, a proto se pro tuto aplikaci zobrazovaly šipky příliš malé), a dále z důležitější změny a to úpravy spouštění programu Metapost z Psgantt (jeden z parametrů používaných při volání Metapostu není v současné době podporován, a proto v nezměněné verzi Psgantt končil svou činnost chybou bez validního výstupu).

Zobrazování vlastností jednotlivých signálů a zpráv a následně i výsledných rozvrhů v této diplomové práci je vytvořeno pomocí výše zmíněného programu Psgantt.

4.2. Popis jednotlivých souborů

Implementovaný algoritmus sestává z následujících souborů. Odsazení jednotlivých položek znázorňuje, jak jsou navzájem jednotlivé soubory volány při běhu algoritmu.

- signal.m - struktura pro signály a zprávy (jelikož ty jsou identické se zprávami), nutná pro funkcionalitu řazení signálů (i zpráv) podle různých parametrů,
- run_2.m - spuštění algoritmu na větším množství setů signálů najednou (postupně) a vygenerování výstupu,
    - v1.m-v7.m - jedná se o postupný vývoj algoritmu od nejnižších čísel k nejvyšším,
    - vClanek.m, vEnhance.m, vBounds.m, vMessage.m, vWindow.m, vDP.m - jedná se o varianty algoritmu porovnávané v článku,
        - rozdel*.m - implementace rozdělení signálů na skupiny dle nodů a přizpůsobení hodnot signálů ze vstupů na taková čísla, se kterými algoritmus počítá,
        - zpracuj_node.m - zapouzdření jednotlivých operací (viz. níže) pro jeden node,
            - frame_packing*.m - implementace frame packingu = sdružování zpráv se stejnou periodou do zpráv - ve výsledku snížení rozvrhovaných elementů,
            - optimalizuj*.m - implementace optimalizace zpráv = sdružování zpráv mezi sebou i s rozdílnými periodami - další snižování rozvrhovaných elementů,
            - rozvrhni*.m - samotné rozvrhování = nalezení místa, kde (první cyklus a slot) se má příslušná zpráva přenášet,
        - vygeneruj_vystup.m - zpětný přepočet z vygenerovaného rozvrhu na podobnou strukturu, jaká byla na vstupu,
        - otestuj.m - test, zda ve výsledném rozvrhu platí pro každý signál

podmínky zadané na vstupu, tj. že signál není ve výsledku přenášen méněkrát, než by měl být atd.,

- performance.m - vygenerování několika hodnot o rozvrhu, které by mohly být zajímavé při jeho hodnocení,
- zobraz.m - grafické zobrazení rozvrhu (vyžaduje předpoklady uvedené v předchozí části kapitoly),
    - psgantt.bat, psgantt.sh - využito při volání Psgantt z prostředí Matlab; využito jen při používání lokální instalace Psgantt pro generování grafického výstupu; zajišťuje nastavování příznaků a čekání po dobu, po kterou je Psgantt spuštěn,

- gcdm.m - výpočet GCD pro vektor hodnot najednou,
- adresář old - obsahuje předchozí vývojové verze jednotlivých částí, které již nejsou využívané.

4.2.1. Další pomocné soubory a soubory nevyužívané přímo algoritmem

Součástí archivu se zdrojovými soubory vyvinutého algoritmu jsou také další pomocné soubory, které nejsou přímo využity samotným algoritmem, ale jsou určeny pro další činnosti potřebné například pro generování výstupních obrázků, zobrazování průběhu algoritmu a podobně. Jedná se o následující soubory:

- ganttlab.m - kopie souboru z TORSCHE pro potřeby volání Psganttu jako webové služby (existující soubor z TORSCHE nelze využít, neboť je veden pouze jako privátní),
- base64*.m - zkopírované soubory z TORSCHE, jelikož jsou vedeny pouze jako privátní,
- example* - několik velmi jednoduchých příkladů signálů, na kterých jsou vidět určité rozdíly a chyby jednotlivých variant algoritmu,
- zobraz_signaly, zobraz_zpravy, zobraz_optimalizovane_zpravy - grafické zobrazení jednotlivých částí průběhu algoritmu (vyžaduje předpoklady uvedené v předchozí části kapitoly),

- article* - vygenerování obrázků do článku (vyžaduje předpoklady uvedené v předchozí části kapitoly),
- spoj - přidání nodu (a jeho signálů) k již existujícímu setu signálů,
- zmen_slot_size - změna velikosti slotu pro sety signálů,
- adresář data - obsahuje množiny signálů pro testování a hodnocení algoritmu.

4.3. Potřebné úpravy před spuštěním algoritmu

Pro spuštění některých operací je nutná triviální úprava zdrojových souborů. Jedná se převážně o nastavení systémových cest.

K využití lokálního generování grafických znázornění vypočtených rozvrhů komunikace na síti FlexRay je nutné upravit cestu k programu Psgantt ve skriptu volajícím tento program. Jedná se o třetí řádek ve skriptu psgantt.bat respektive psgantt.sh, dle použitého operačního systému, kde je třeba upravit hodnotu proměnné PSGANTT_PATH. Zároveň je třeba tento skript zkopírovat do adresáře, který se nachází v systémové cestě.

Pokud je potřeba spustit algoritmus pro více množin signálů (více vstupních souborů), spustí se soubor run2.m. Kód v tomto souboru zajistí, že bude algoritmus spuštěn postupně na všech dostupných množinách signálů v definované složce a následně vytvoří soubor s výsledky pro každou množinu signálů ve formátu CSV. Aby byl program schopen identifikovat cestu, ve které jsou uloženy soubory s množinami signálů, je nutné upravit hodnotu proměnných DATA_DIRECTORY a RESULT_DIRECTORY na řádcích 10 a 11. První proměnná definuje adresář, ve kterém se hledají vstupní soubory pro algoritmus ve formátu datových souborů Matlab *.mat. Ve složce, která je definována druhou proměnnou, bude po spuštění algoritmu na všech dostupných množinách signálů vytvořen soubor vysledky.csv. Proto musí mít uživatel právo zápisu do této složky. V případě potřeby spuštění jiné verze algoritmu, než je jeho vylepšená varianta, je nutné změnit jméno volané funkce na řádku 25. Vytvořený soubor s výsledky ve formátu CSV je možné pro lepší čitelnost otevřít v libovolném tabulkovém kalkulátoru (například Microsoft Excel či OpenOffice Calc). Soubor je uložen ve znakové sadě Unicode (UTF–8), jako oddělovač je použit tabulátor

a pro textové hodnoty jsou použity dvojité uvozovky.

Využití celočíselného lineárního programování pro tvorbu rozvrhu komunikace z optimalizovaných zpráv lze povolit změnou proměnné ILP na hodnotu 'true' v souborech jednotlivých verzí algoritmů. Například v souboru vEnhance.m na řádku 34. Pro povolení ILP bude řádek vypadat takto:

```
if (not(exist('ILP','var'))); ILP = true; end;
```

4.4. Očekávaný vstupní formát souborů pro algoritmus

Vstupem algoritmu je Matlab soubor množiny signálů. Tento soubor musí mít následující proměnné, které popisují vlastnosti signálů:

- **nc_ear_slot** řádkový vektor hodnot release time jednotlivých signálů uváděný v počtu slotů od začátku komunikace pro první výskyt signálu,
- **nc_lat_slot deadline** řádkový vektor hodnot deadline jednotlivých signálů uváděný v počtu slotů od začátku komunikace pro první výskyt signálu,
- **nc_n_nodes** počet nodů, mezi které jsou signály rozděleny,
- **nc_frame_len** řádkový vektor délek jednotlivých signálů uváděný v počtu přenášených bitů,
- **nc_frame_names** sloupcový vektor buněk, ve kterých se nachází názvy jednotlivých signálů,
- **nc_l_cycle** řádkový vektor délek cyklu v počtu slotů pro každý signál,
- **nc_l_slot** délka slotu v bitech,
- **nc_node_of_signals** řádkový vektor čísel nodů, ke kterým patří signál na dané pozici,
- **nc_periods** řádkový vektor period signálů uváděných v počtu slotů pro každý signál.

Všechny vstupní vektory musí mít stejnou velikost.

4.5. Generovaný výstupní formát souboru

Výstupem algoritmu je Matlab soubor pojmenovaný stejně jako vstupní soubor, navíc však má přídomek '_result'. Tento soubor obsahuje následující proměnné:

- **nc_n_slots_per_node** řádkový vektor počtu slotů potřebných pro jednotlivé nody ve výsledném rozvrhu,
- **nc_slot_of_signals** řádkový vektor udávající číslo (ID) slotu, ve kterém je posílán každý signál,
- **nc_start_cycles** řádkový vektor s údajem v kolikátém cyklu od začátku komunikace (též každé hyperperiody) se každý signál poprvé posílá,
- **nc_offsets** řádkový vektor posunu v bitech od začátku slotu, kdy se má každý signál odesílat,
- **nc_periods** řádkový vektor period jak často je každý signál posílán, může se lišit od původní periody, pokud v rámci algoritmu bylo rozhodnuto, že se signál bude posílat častěji.

Závěr

Cílem této diplomové práce bylo seznámit se s komunikačním systémem FlexRay a navrhnout způsob rozvrhování signálů statického segmentu na této síti. Následovat měla implementace navrženého řešení, její reálné ověření na síti FlexRay a konečné porovnání dosažených výsledků s ostatními již existujícími možnostmi rozvrhování.

Všechny definované části byly splněny, tj. bylo navrženo řešení tvorby rozvrhu pro signály, které mají být přenášeny ve statickém segmentu sítě FlexRay, tento návrh byl implementován v prostředí Matlab a jeho výsledky byly ověřeny praktickým testem na fyzické síti FlexRay. Navržený způsob rozvrhování se od existujících možností liší především respektováním všech časových omezení periodických signálů. Dále je schopen zpracovávat mnohem větší množství signálů v krátkém čase, než je možné dosavadními metodami. Zároveň dosahuje algoritmus velmi dobrých výsledků především v situacích, kdy každé zařízení posílá větší množství signálů. Pokud některé zařízení posílá málo signálů, nemusí být algoritmus natolik efektivní, nicméně stále poskytuje dobré výsledky.

Souběžně s touto diplomovou prácí byl vytvořen příspěvek, který byl přijat na konferenci The 10th International Workshop on Real-Time Networks v Portugalsku, jež popisoval první verzi navrženého algoritmu a ukazoval jeho úspěchy. Tento příspěvek byl na konferenci prezentován vedoucím diplomové práce. Představený způsob rozvrhování signálu byl posluchači hodnocen kladně. Dále je v současné době v přípravě článek do IEEE Transactions on Industrial Informatics, který popisuje vylepšenou verzi tohoto algoritmu.

Seznam literatury a použitých zdrojů

[1] Amos Albert. Comparison of Event-Triggered and Time-Triggered Concepts with Regard to Distributed Control Systems. In Embedded World 2004, strany 235-252, 2004.

[2] AUTOSAR Development Partnership. Specification of FlexRay Transport Layer. Version 2.0.1, červen 2006.

[3] AUTOSAR Development Partnership. Specification of Module FlexRay Interface. Version 2.0.0, červen 2006.

[4] Josef Berwanger, Martin Peteratzinger, a Anton Schedl. Flexray startet durch – FlexRay-Bordnetz für Fahrdynamik und Fahrerassistenzsysteme. In Elektronik automotive: Sonderausgabe 7er BMW, 2008.

[5] Christelle Braun, Lionel Havet, a Nicolas Navet. Netcarbench: A benchmark for techniques and tools used in the design of automotive communication systems. In The 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems (FeT‘2007), listopad 2007.

[6] Edward G. Coffman, Michael R. Garey and David S. Johnson. Approximation Algorithms for Bin-Packing – An Updated Survey. Algorithm Design for Computer Systems Design, G. Ausiello, M. Lucertini, and P. Serafini (eds.), pp. 49-106, Springer-Verlag, 1984.

[7] FlexRay Consortium. Flexray Communication System Protocol Specification Version 3.0.1, 2010.

[8] FlexRay Consortium. FlexRay Communication System Electrical Physical Layer Specification Version 3.0.1, 2010

[9] FlexRay Consortium. Webová prezentace komunikačního systému FlexRay, dostupné online na <http://www.flexray.com>

[10] Mathieu Grenier, Lionel Havet, a Nicolas Navet. Configuring the communication on FlexRay: the case of the static segment. In Embedded Real Time Software, Francie, 2008.

[11] Zdeněk Hanzálek, David Beneš a Denis Waraus. Time Constrained FlexRay Static Segment Scheduling. 10th International Workshop on Real-Time Network, Portugalsko, 2011

[12] Antonín Karolík. Nástroj na kreslení Ganttových diagramů s využitím Metapostu. Diplomová práce. Praha, ČVUT, 2007

[13] Hermann Kopetz. A solution to an automotive control system benchmark. In Real-Time Systems Symposium, 1994., Proceedings., strany 154-158, 7-9 1994.

[14] Viktor Pokorný. Metody měření vybraných parametrů komunikačního standardu FlexRay a jejich implementace. Diplomová práce. Praha, ČVUT, 2007

[15] Traian Pop, Paul Pop, Petru Eles, Zebo Peng, and Alexandru Andrei. Timing analysis of the FlexRay communication protocol. Real-Time Systems, 2006. 18th Euromicro Conference on, strana 11 pp. -216, 2006.

[16] SAE Technical Report J2056/1. Class C Application Requirement Considerations, červen 1993

[17] Ece Guran Schmidt a Klaus Schmidt. Message scheduling for the FlexRay protocol: The dynamic segment. Vehicular Technology, IEEE Transactions on, 58; 58(5):2160-2169, 2009.

[18] Klaus Schmidt a Ece Guran Schmidt. Message scheduling for the FlexRay protocol: The static segment. Vehicular Technology, IEEE Transactions on, 58; 58(5):2170-2179, 2009.

[19] School of Engineering Sciences, University of Southampton. Rozšíření prostředí Matlab XML Toolbox, dostupné online na <http://www.geodise.org/toolboxes/generic/xml_toolbox.htm>

[20] Byungseok Seo a Dongik Lee. Determining the length of static message for efficient use of FlexRay network. SICE Annual Conference 2010, Proceedings of, strany 563-566, srpen 2010.

[21] Jonathan Smith. Hop on the FlexRay; dostupný na <http://jcwinnie.biz/wordpress/?p=1633>, staženo dne 13.12.2011

[22] Young Hun Song, Man Ho Kim, Suk Lee, Kyung Chang Lee. Node-based FlexRay Scheduling Method for Reducing the Scheduling Komplexity. Proceedings of 2011 International Conference on Mechanical, Industrial, and Manufacturing Engineering, 2011

[23] Piotr Swedrowski a Raphael Guerra. Optimal scheduling approaches for FlexRay bus. Proceedings of 5th Real-Time Systems Seminar, 2011

[24] Přemysl Šůcha, Michal Kutil, Michal Sojka, Zdeněk Hanzálek. TORSCHE Scheduling Toolbox for Matlab. In IEEE International Symposium on Computer-Aided Control Systems Design. Mnichov, Německo, 2006

[25] Bogdan Tanasa, Unmesh Dutta Bordoloi, Petru Eles a Zebo Peng. Reliability-aware frame packing for the static segment of flexray. Proceedings of the ninth ACM international conference on Embedded software, strany 175-184, 2011

[26] Bogdan Tanasa, Unmesh Dutta Bordoloi, Petru Eles a Zebo Peng. Scheduling for Fault-Tolerant Communication on the Static Segment of FlexRay. Real-Time Systems Symposium (RTSS), 2010 IEEE 31st, 2011

[27] Haibo Zeng, Marco Di Natale, Arkadeb Ghosal, a Alberto Sangiovanni-Vincentelli. Schedule Optimization of Time-Triggered Systems Communicating Over the FlexRay Static Segment. Industrial Informatics, IEEE Transactions on, 7(1):1-17, 2011.

Příloha A: Seznam použitých zkratek

AUTOSAR – AUTomotive Open System ARchitecture

CAN – Controller Area Network

CRC – Cyclic Redundancy Check

CSV – Comma Separated Values

DP – Dynamic Programming

ECU – Electronic Control Unit

FP – Frame Packing

GCD – Greatest Common Divisor

GLPK – GNU Linear Programming Kit

IEEE – Institute of Electrical and Electronics Engineers

ILP – Integer Linear Programming

ISO – International Organization for Standardization

MILP – Mixed-Integer Linear Programming

NIT – Network Idle Time

OZ – Optimalizace Zpráv

TDMA – Time Division Multiple Access

XML – Extensible Markup Language

Příloha B: Obsah přiloženého CD

/data: adresář obsahuje používané množiny signálů pro testování výsledků vyvinutého algoritmu.

/docs: adresář obsahuje tuto diplomovou práci ve formátu PDF.

/source: adresář obsahující zdrojové kódy vyvinutého algoritmu.