Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

WirelessHART

Plzeň 2014 Lukáš Kopáček

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 26. června 2014 Lukáš Kopáček

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce Ing. Jiřímu Ledvinovi CSc. z Katedry informatiky a výpočetní techniky FAV ZČU za jeho vřelý přístup, cenné rady a poskytnuté materiály. Dále bych rád poděkoval svým nejbližším za čas strávený nad korekturou práce.

Abstrakt

Tato diplomová práce je zaměřena na návrh a realizaci programového vybavení pro uzel sítě komunikující dle specifikace WirelessHART.

Klíčová slova

WirelessHART, HART, CC2430, Cortex-M3, LM3S9B92, EVALBOT, časová synchronizace, bezdrátové senzorické sítě, ZigBee, IEEE 802.15.4.

Abstract

WirelessHART

This master thesis focuses on design and implementation of software for node communicating by WirelessHART specification.

Keywords

WirelessHART, HART, CC2430, Cortex-M3, LM3S9B92, EVALBOT, time synchronization, wireless sensor network, ZigBee, IEEE 802.15.4.

Obsah

1 Úvod

S rostoucím zájmem o automatizaci a monitorování řídících procesů roste zájem o sítě, které umožní spolehlivé snímání parametrů řízených procesů. Pro tyto účely byl vyvinut komunikační protokol HART. Nevýhodou tohoto protokolu je nutnost rozvedení vodičů a vysoká spotřeba měřící stanice. Proto byla vyvinuta bezdrátová varianta tohoto prokolu s názvem WirelessHART.

Cílem této práce je navrhnout a realizovat odpovídající programové vybavení umožňující bezdrátovou komunikaci mezi několika uzly pomocí specifikace WirelessHART.

2 Teoretická část

2.1 HART

Protokol HART (Highway Addressable Remote Transducer) byl vyvinut v polovině 80. let společností Rosemount Inc. pro použití s inteligentními měřícími přístroji. Původně proprietární protokol byl brzy zveřejněn pro volné použití, a v roce 1990 byla vytvořena HART User Group. V roce 1993 byla všechna práva a registrovaná ochranná známka společnosti převedena na HART Communication Foundation (HCF). Protokol zůstal otevřený pro volné použití bez licenčních poplatků. [7]

Protokol HART je celosvětový standard pro přenos digitálních informací mezi inteligentními zařízeními a monitorovacím systémem pomocí analogové proudové smyčky 4 - 20 mA.

HART je protokol typu master - slave, zařízení typu master komunikuje se zařízením typu slave systémem dotaz - odpověď.

Tento protokol také podporuje režim „burst“, při kterém může jedno zařízení typu slave periodicky vysílat zprávu s odpovědí.

2.1.1 Fyzická vrstva

Fyzická vrstva je založena na standardu Bell 202. Používá frekvenční modulaci a rychlost přenosu 1200 Bd/s. Frekvence pro reprezentaci logické 0 a logické 1 jsou 2200 a 1200 Hz. Tento signál je modulován na analogový měřící signál 4 - 20 mA. K jedné proudové smyčce může být připojeno až 15 nebo i více zařízení typu slave. [7]

2.1.2 Linková vrstva

Linková vrstva definuje protokol typu master - slave. Systém může pracovat v jednom ze dvou režimů, v normálním řežimu nebo burst režimu.

V normálním režimu odpovídá zařízení jen když je vyslán požadavek. V síti mohou být dvě zařízení typu master - primární (např. řídící systém) a sekundární (např. ruční HART komunikátor). Okamžiky, kdy zařízení typu master může zahájit komunikaci jsou definovány speciálním algoritmem pro předávání pověření.

V burst režimu přebírá funkci řízení vybraná slave stanice. V jednu chvíli může být v burst režimu pouze jedna stanice. Algoritmus předávání pověření je pak modifikován tak, že se ve vysílání může střídat burst stanice, primární i sekundární master.

2.1.3 Aplikační vrstva

Aplikační vrstva definuje příkazy, odpovědi, datové typy a stavová hlášení podporovaná protokolem HART. Příkazy protokolu jsou rozděleny do několika hlavních skupin:

1. univerzální příkazy (Universal Commands) - příkazy, které musí být podporovány ve všech zařízeních,
2. provozní příkazy (Common Practice Commands) - příkazy společné pro většinu zařízení,
3. specifické příkazy (Device Specific Commands) - příkazy jedinečné pro konkrétní zařízení, nebo určitý typ zařízení.

2.2 WirelessHART

WirelessHART je robustní bezdrátový protokol navržený pro měřící a řídící procesy v průmyslu. Je založen na osvědčené a známé technologii HART. Umožňuje uživatelům rychle a snadno využít výhody bezdrátové technologie a zároveň zachovává zpětnou kompatibilitu s existujícími zařízeními protokolu HART.

2.2.1 Fyzická vrstva

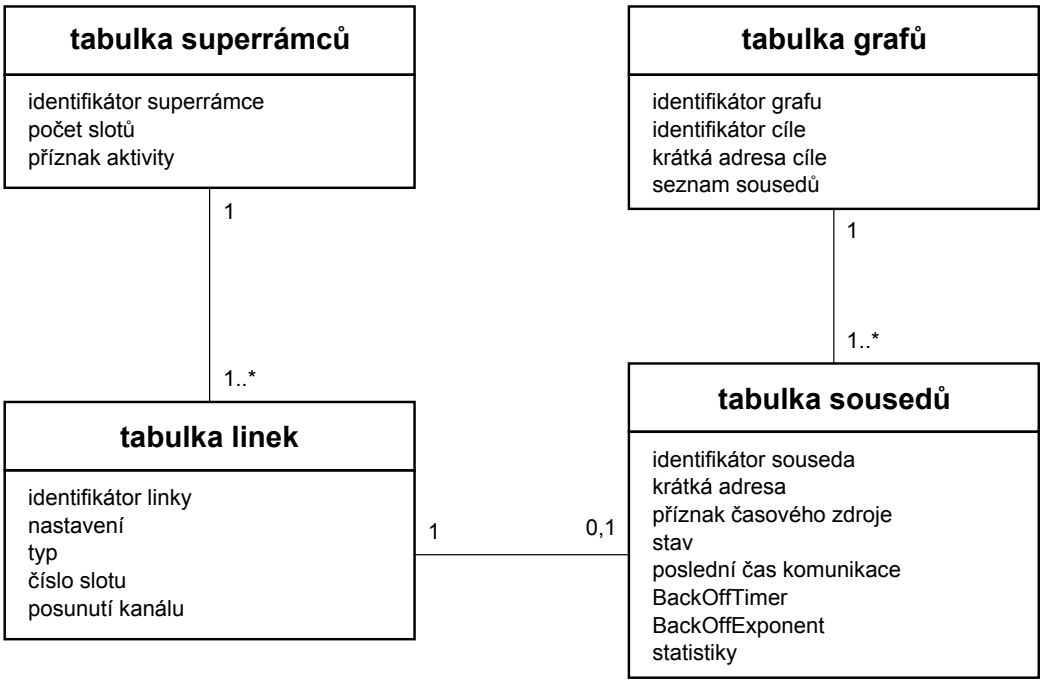
Fyzická vrstva protokolu WirelessHART [8] je založena na standardu IEEE STD 802.15.4-2006 2.4GHz DSSS, který definuje charakteristiky radiového přenosu jako jsou způsob signalizace, modulace, rychlost přenosu, síla vysílaného signálu, šířka pásma jednotlivých kanálů a citlivost zařízení.

Stejně jako IEEE 802.15.4 pracuje i WirelessHART v nelicencovaném ISM pásmu 2400 - 2483.5MHz s maximální přenosovou rychlostí 250 kb/s. Kanály jsou číslované od 11 do 26 po 5 MHz. [1]

2.2.2 Linková vrstva

Protokol WirelessHART používá na linkové vrstvě pojmy:

- soused - dosažitelný sousední uzel, se kterým lze komunikovat,
- slot - časový interval pro plánování provozu sítě,
- graf - orientovaný sled sousedů s možnými redundantními cestami,



Obrázek 2.1: Vztahy mezi tabulkami linkové vrstvy

- linka - komunikační kanál mezi dvěma (nebo více) sousedními uzly sítě, který má určený slot, kanál, sousední uzel a typ,
- superrámec - daný počet linek s předem určenou posloupností

Tabulky linkové vrstvy

Pro řízení linkové vrstvy používá protokol WirelessHART čtyři tabulky [2]. Tabulku linek, tabulku grafů, tabulku sousedů a tabulku superrámců. Tyto tabulky obsahuje každý uzel sítě. Vztahy mezi tabulkami jsou znázorněny na obrázku 2.1.

Tabulka grafů obsahuje číslo grafu a seznam sousedních uzlů. Velikost seznamu sousedních uzlů udává počet redundantních cest. Identifikátor cíle a krátká adresa cíle jsou nepovinné atributy.

Tabulka superrámců osahuje identifikátor superrámce, délku superrámce a příznak, je-li aktivní. Délka superrámce se určuje v počtu časových slotů.

Tabulka linek obsahuje identifikátor linky, nastavení linky, typ linky, číslo slotu, posunutí kanálu a odkazy na superrámec a na souseda. Podle tabulky linek se určuje s kterým sousedem má uzel komunikovat, na jakém kanálu a v jakém slotu.

Tabulka sousedů obsahuje dlouhou i krátkou adresu souseda, příznak zdroje pro časovou synchronizaci, stav souseda, čas poslední komunikace, hodnoty časovače a statistická data. Dlouhá adresa je jednoznačným identifikátorem sousedního uzlu. Hodnoty časovače se využívají pro vyhnutí se kolizi na sdílených linkách.

Vzájemná komunikace

Pro vzájemnou komunikaci uzly sítě využívají linek. Linky, které uzel využívá přiděluje centrálně jediný uzel - network manager. Network manager plánuje provoz celé síti. Má možnost naplánovat i paralelně probíhající přenosy bez vzájemného rušení.

Pro přístup ke sdílenému médiu je použita metoda TDMA. TDMA je přístupová metoda pro bezkolizní deterministickou komunikaci. Tato metoda je založena na rozdělení času do časových slotů. V každém slotu spolu mohou komunikovat uzly sítě, kterým je slot přiřazen. Časové sloty jsou plánovány a mají pevnou délku 10 ms.

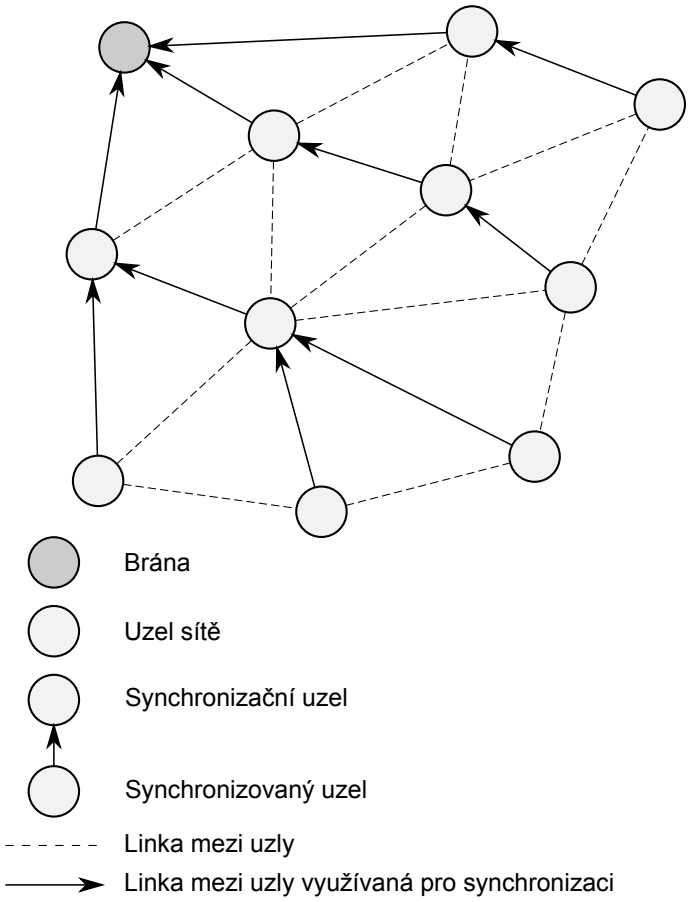
Přeskakování mezi kanály, tzv. „channel hopping“ zajišťuje komunikaci na rozdílných kanálech. Pro určení na kterém kanálu se v daném slotu vysílá se používá absolutní číslo slotu (absolute slot number), posunutí kanálu (channel offset) a seznam aktivních kanálů (channel map). Absolutní číslo slotu je identické pro všechna zařízení v síti. Posunutí kanálu je uvedeno v tabulce linek. Seznam aktivních kanálů je pro všechny uzly stejný a spravuje jej centrální uzel sítě - network manager.

Pro zvýšení spolehlivosti je TDMA používáno v kombinaci s přeskakováním kanálů a CCA (clear channel assessment). CCA je způsob ověření, že na daném kanálu nikdo nevysílá. Podstatou CCA je naslouchání na dané frekvenci po určitou dobu. Pokud během naslouchání není zachycena komunikace, je kanál považován za volný. Z kombinace TDMA a přeskakování kanálů vyplývá, že obě komunikující zařízení musejí přijímat resp. vysílat ve stejný čas a na stejné frekvenci. Pro zvýšení spolehlivosti přenosu obsahuje protokol WirelessHART i možnost zakázat použití příliš zarušených kanálů (tzv. channel blacklisting).

Použitím těchto mechanismů minimalizuje protokol WirelessHART rušení od ostatních bezdrátových systémů.

2.2.3 Časová synchronizace

Pro chod sítě je velice důležitá přesná synchronizace časových slotů mezi sousedními uzly. Pro organizování časových slotů používá protokol WirelessHART superrámce (tzv. superframe). Pro časovou synchronizaci jsou uzly



Obrázek 2.2: Precedenční graf pro časovou synchronizaci

uspořádány do precedenčního grafu, vytvářeného Network managerem (viz obr. 2.2). Kořenem grafu je vstupní uzel sítě - brána. Nadřazený uzel je označován jako synchronizační, podřízený jako synchronizovaný.

Každý synchronizovaný uzel synchronizuje své vnitřní hodiny s vybranými sousedy (synchronizační uzly). Network manager přidělí každému synchronizovanému uzlu množinu synchronizačních uzlů.

Synchronizovaný uzel sítě se synchronizuje podle rámců přijatých od synchronizačních sousedů. Když synchronizovaný uzel přijme paket od synchronizačního uzlu, je čas přijmutí uložen a porovnán s ideálním časem příjmu. Diference mezi časy je poslána v potvrzovacím rámci (ACK). Pokud komunikaci zahájí synchronizovaný uzel, synchronizuje se synchronizovaný uzel podle hodnoty uvedené v potvrzovacím rámci. Pokud komunikaci zahájí synchronizační uzel, synchronizuje se synchronizovaný uzel podle času přijetí rámce od synchronizačního uzlu.

Synchronizace probíhá posunutím začátku následujícího časového slotu, případně změnou délky časového slotu.

2.2.4 Síťová a transportní vrstva

Základní prvky sítě WirelessHART jsou [6]

- koncová zařízení (Field Devices) - zařízení připojená ke sledovanému procesu,
- mobilní zařízení (Handhelds) - přenosná zařízení používaná při údržbě jako například spouštění diagnostiky, vyvolání kalibrace a konfigurace zařízení,
- brány (Gateways) - propojují koncová zařízení s hostitelskou aplikací,
- network manager - je zodpovědný za konfiguraci sítě, správu směrovacích tabulek a plánování komunikace mezi zařízeními,
- security manager - je zodpovědný za distribuci šifrovacích klíčů.

WirelessHART vytváří sítě typu mesh s redundantními cestami mezi uzly. Pakety tedy mohou být přenášeny i při výpadku některých linek. Všechna zařízení v síti musejí být schopna směrovat pakety. Protokol definuje dva mechanismy směrování (graph routing a source routing)[3], které musejí podporovat všechna zařízení.

Směrování podle grafu (graph routing) je směrování podle seznamu hran, které propojují uzly v síti. Hrany v každém grafu jsou konfigurovány network managerem a jsou uloženy v každém zařízení. Zařízení směruje paket podle graph ID (uloženo v síťové hlavičce paketu). Zasílá jej vybranému sousedovi. Soused je vybrán podle záznamů z tabulky grafů linkové úrovně.

Směrování podle zadané cesty (source routing) je směrování mezi zdrojovým a cílovým uzlem podle zadaného seznamu mezilehlých uzlů. V paketu je uveden seznam krátkých adres uzlů obsahující 4 nebo 8 položek.

Směrování podle grafu má výhodu v tom, že mezi dvěma uzly sítě může existovat více cest. Pokud je některá cesta přerušena, může být použita cesta náhradní. Směrování podle zadané cesty naopak vyžaduje dodržení nadefinované cesty. Při výpadku uzlu nebo při ztrátě spojení neexistuje žádná náhradní cesta.

Směrovací informace pro jednotlivé uzly spravuje centralizovaně network manager. Informace o topologii získává network manager postupně díky algoritmu pro přidávání uzlů do sítě.

2.2.5 Aplikační vrstva

Aplikační vrstva protokolu WirelessHART [4] je v principu stejná jako u protokolu HART. Oba protokoly obsahují množinu univerzálních příkazů (Universal Commands), provozních příkazů (Common Practice Commands) a množinu specifických příkazů (Device Specific Commands) pro aplikaci. Aplikační vrstva protokolu WirelessHART je navíc rozšířena o skupinu příkazů pro konfiguraci bezdrátové sítě. Aplikace jsou navrhovány pro řešení rozdílných úkolů, ale základ zůstává stejný: přijmout příkaz, na jeho základě provést akci, vygenerovat odpověď a poslat ji zpět žadateli. (viz [6])

3 Návrh systému

3.1 Použité technické prostředky

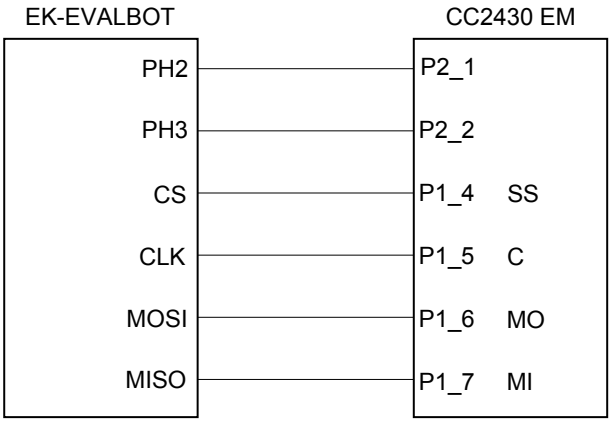
Systém je vyvíjen na vývojovém kitu Stellaris® LM3S9B92 EVALBOT Robotic Evaluation Board (`EVALBOT`) od firmy Texas Instruments. Tento vývojový kit obsahuje 2 USB konektory (USB host, USB device), konektor pro RJ45 pro připojení do sítě Ethernet, konektor pro MicroSD kartu, 2 motory, display, 4 tlačítka, 2 LED diody a konektory pro připojení radiového modulu.

Mikrokontroler LM3S9B92 obsahuje procesorové jádro ARM Cortex-M3, 256 KB Flash paměti, 96 KB paměti SRAM, periferie pro komunikaci (UART, SSI, I2C, I2S, CAN, Ethernet MAC a PHY, USB), analogové a digitální komparátory, analogově-digitální převodníky (ADC) a regulátor napětí. Pro ladění je k dispozici rozhraní JTAG a SWD (Serial Wire Debug - rozhraní pro ladění po seriovém portu).

Jako radiový modul byl použit `CC2430 EM`. Tento modul obsahuje jen procesor, konektor pro připojení antény a konektory pro připojení k vývojovému kitu `EVALBOT`. Pro komunikaci s vývojovým kitem `EVALBOT` je využito 6 pinů, z toho jsou 4 piny použité pro komunikaci přes SPI rozhraní. Schematické propojení modulů je na obrázku 3.1. Tento modul neobsahuje žádné rozhraní pro ladění pomocí debuggeru, proto bylo pro ladění použito radiové rozhraní. Pro naprogramování modulu `CC2430 EM` je použita deska `SmartRF 04EB` od Texas Instruments.

Pro ladění programového vybavení radiového modulu byly použity ladící výpisy přes radiové rozhraní a indikace stavu prostřednictvím dvou pinů. Během ladění tedy nebylo možné modul zastavit pomocí debuggeru nebo jej restartovat. Pro restart modulu bylo potřeba vypnout a zapnout napájení pro celý vývojový kit. Tato skutečnost znemožnila použití debuggeru pro ladění komunikace s radiovým modulem i pro ladění programového vybavení vývojového kitu `EVALBOT`.

Pro vytvoření programového vybavení kitu `EVALBOT` bylo použito vývojové prostředí Code Composer Studio verze 4 a operační systém FreeRTOS. Pro radiový modul byl použit volně šiřitelný překladač SDCC a vývojové prostředí Eclipse. Programové vybavení radiového modulu neobsahuje operační systém.



Obrázek 3.1: Propojení `EVALBOT` s `CC2430 EM`

3.2 Operační systém FreeRTOS

Operační systém FreeRTOS^{TM}od Real Time Engineers Ltd. má vedoucí postavení na trhu s operačními systémy reálného času. Podporuje 34 architektur a zaznamenává více než 100 000 stažení za rok. Je vyvíjen profesionálně s přísnou kontrolou kvality. Tento systém je robustní, podporovaný a zdarma i pro komerční použití bez nutnosti zveřejňovat své zdrojové kódy. Kromě neplacené verze existují i placené verze tohoto operačního systému. Je používán ve všech možných odvětvích od hraček až po navigaci letadel. [5]

Operační systém FreeRTOS^{TM}podporuje rozdělení do úloh, preemptivní plánování, zpracování událostí, softwarové časovače, synchronizaci a komunikaci mezi úlohami pomocí front zpráv, semaforů a mutexů.

Základem operačního systému FreeRTOS je jádro s podporou preemtivního plánování a zpracování událostí. Úlohy lze rozdělit na vlákna (označované jako tasky) a přidělit jim prioritu. Jednotlivé tasky jsou jádrem převáděny mezi stavy připraven, běží, čeká na událost a neaktivní. Tasky mají vlastní zásobník, mohou být vytvořeny i za běhu aplikace.

K realizaci tasků, semaforů a front zpráv slouží systémová volání, jejichž součástí je i rezervace potřebné vnitřní paměti (memory management).

3.3 Návrh komunikačního programového vybavení

Fyzická vrstva je implementována na modulu `CC2430EM`. Všechny vyšší vrstvy protokolu jsou implementovány v kitu `EVALBOT`.

3.3.1 Fyzická vrstva

Protokol WirelessHART používá fyzickou vrstvu protokolu IEEE 802.15.4-2006. Programové vybavení fyzické vrstvy je realizováno v modulu `CC2430 EM`.

Fyzická vrstva funguje jako stavový automat viz obr. 3.2. Přechody mezi stavy se řídí aktuálním vnitřním stavem fyzické vrstvy a přijatým znakem přes rozhraní SPI. Při změně stavu se vyvolá přerušení v SPI masteru (EVALBOT), který tak může zaznamenat okamžik přijetí začátku rámce a reagovat na aktuální stav fyzické vrstvy. Pro správnou funkci fyzické vrstvy a časové synchronizace je nutné signalizovat přijetí začátku rámce vyšší vrstvě. Jelikož komunikace přes rozhraní SPI vyžaduje 4 piny, zbývají na signalizaci stavu pouze 2 piny. Fyzická vrstva tedy může signalizovat celkem 4 stavy (`IDLE`, `BUSY`, `RX_SFD`, `ERROR`).

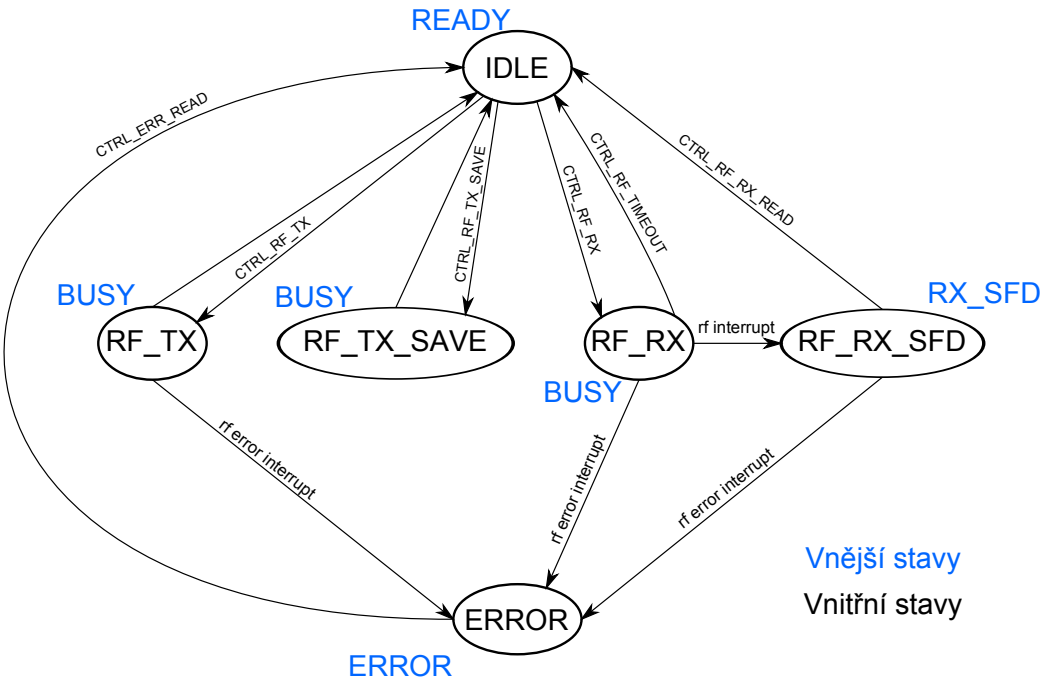
Na obrázku 3.3 je zobrazen časový průběh komunikace mezi moduly EVALBOT a CC2430 pro vyslání zprávy. V první fázi vyšle EVALBOT po SPI zprávu `CTRL_RESET` a čeká dokud není stav fyzické vrstvy `READY`. Poté vyšle příkaz `CTRL_RF_TX_SAVE`, fyzická vrstva přejde do vnitřního stavu `RF_TX_SAVE`, což signalizuje změnou vnějšího stavu na stav `BUSY`. V tomto stavu fyzická vrstva očekává předání zprávy pro vysílání ve formátu uvedeném na obrázku 3.4. Po předání zprávy přejde fyzická vrstva do vnitřního stavu `IDLE`, což signalizuje přechodem do vnějšího stavu `READY`. Pak už jen EVALBOT vyšle příkaz `CTRL_RF_TX`. Na tento příkaz fyzická vrstva zareaguje přechodem do vnitřního stavu `RF_TX` a zároveň přechodem do vnějšího stavu `BUSY`. Po odvysílání zprávy přejde fyzická vrstva zpět do vnitřního stavu `IDLE` a vnějšího stavu `READY`.

Obrázek 3.5 zobrazuje průběh komunikace mezi moduly pro příjem zprávy. Pro zahájení příjmu vyšle EVALBOT příkaz pro resetování, kterým dostane fyzickou vrstvu do vnitřního stavu `IDLE` resp. do vnějšího stavu `READY`, poté vyšle příkaz `CTRL_RF_RX`, kterým uvede fyzickou vrstvu do vnitřního stavu příjmu (`RF_RX`), což signalizuje vnějším stavem `BUSY`.

Pokud před příchodem začátku rámce vyšle EVALBOT příkaz `CTRL_RF_TIMEOUT`, ukončí fyzická vrstva příjem, přejde do vnitřního stavu `IDLE` resp. do vnějšího stavu `READY`.

Při přijmutí znaku začátku rámce (start of frame delimiter) přejde fyzická vrstva do vnitřního stavu `RF_RX_SFD` resp. do vnějšího stavu `RF_RX`. Po přijetí příkazu `CTRL_RX_READ` pošle po SPI jeden znak zprávy. Odeslání celé zprávy signalizuje přechodem do vnitřního stavu `IDLE` resp. do vnějšího stavu `READY`.

Pokud se při příjmu zprávy vyskytne chyba, signalizuje to fyzická vrstva přechodem do stavu `ERROR`. V tomto stavu čeká na příkaz `CTRL_ERR_READ`, na který odpoví odesláním chybového kódu a přechodem do stavu `IDLE` resp.



Obrázek 3.2: Konečný automat s vnitřními a vnějšími stavy fyzické vrstvy

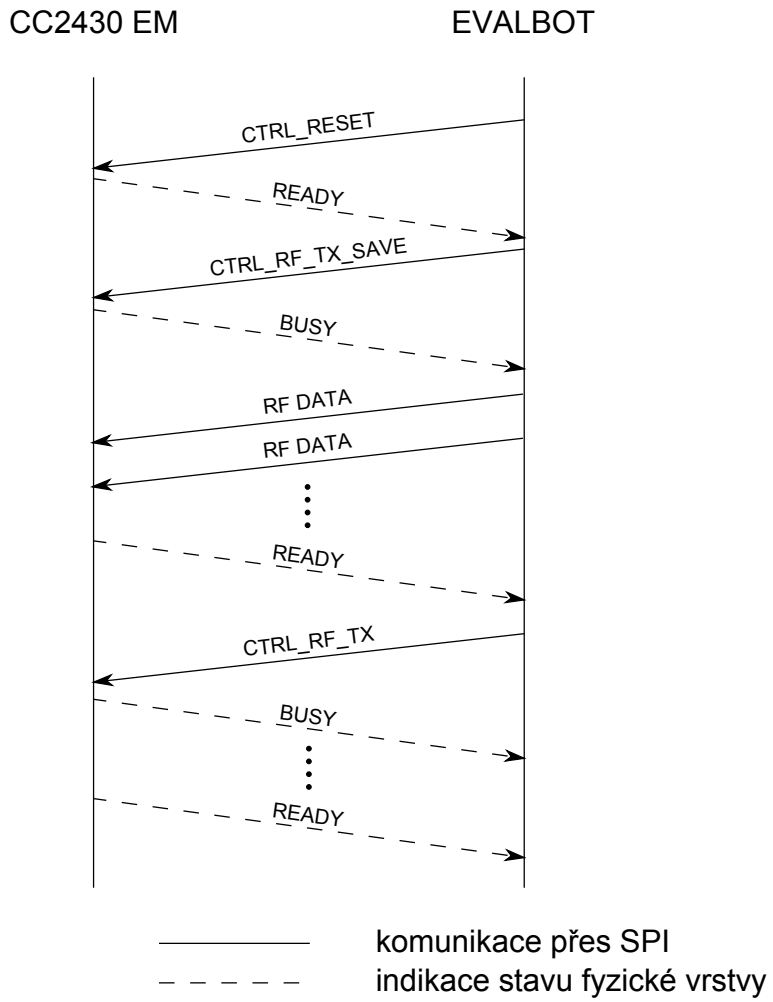
`READY`.

Na fyzické vrstvě definuje specifikace protokolu WirelessHART následující servisní primitiva:

- `ENABLE_request` pro uvedení fyzické vrstvy do stavu vysílání, příjmu nebo vypnutí,
- `ENABLE_confirm` pro potvrzení vykonání příkazu,
- `ENABLE_indicate` pro detekci příjmu značky začátku rámce,
- `DATA_request` pro předání dat k vysílání fyzické vrstvě,
- `DATA_confirm` pro potvrzení odeslání dat na fyzickou vrstvu,
- `DATA_indicate` pro detekci přijetí paketu na fyzické vrstvě.

3.3.2 Linková vrstva

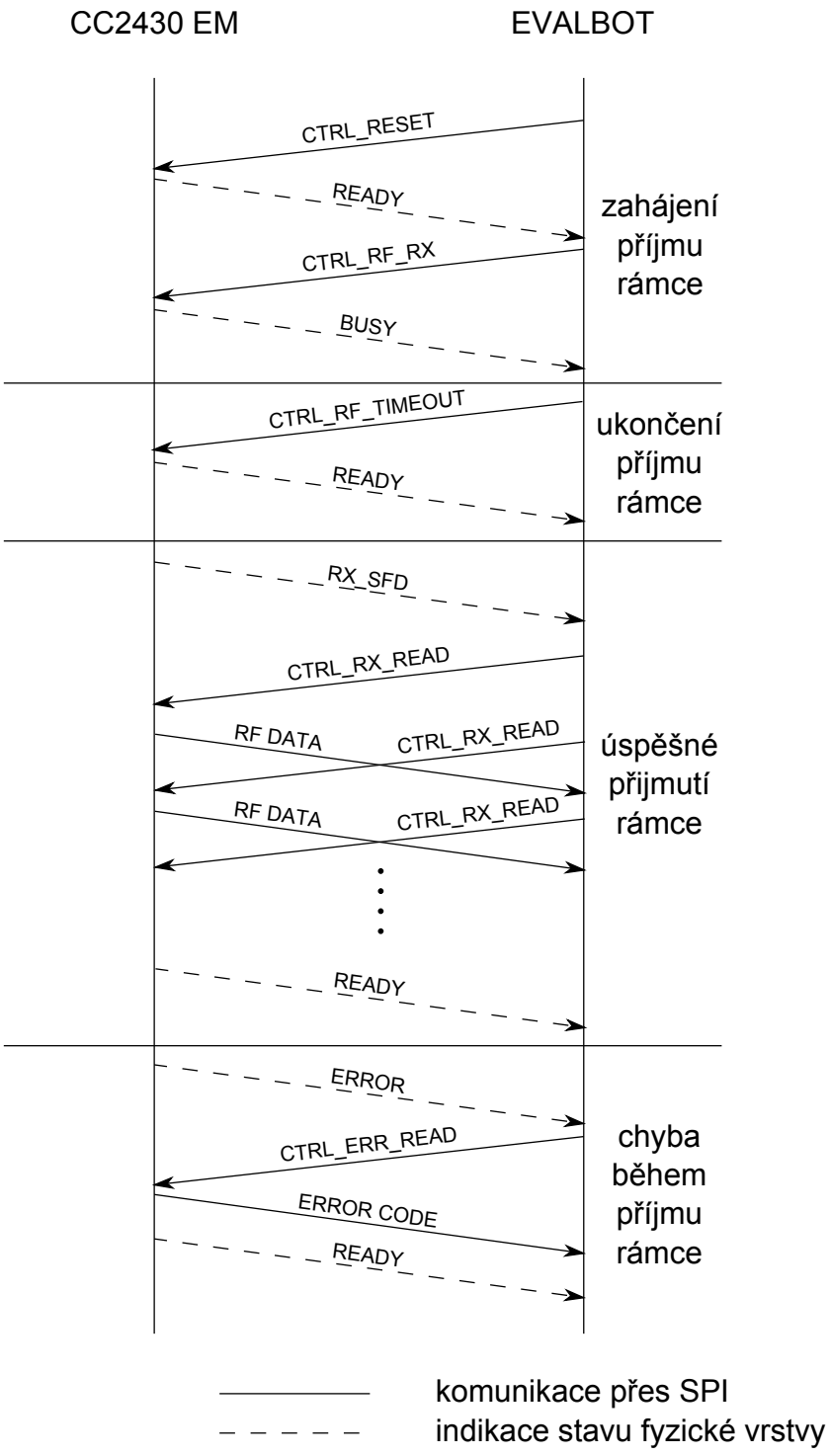
Programové vybavení linkové vrstvy je realizováno v kitu `EVALBOT`. Linková vrstva se vnitřně dělí na přístupovou a řídící podvrstvu. Přístupová podvrstva



Obrázek 3.3: Komunikační diagram pro vyslání zprávy na fyzickou vrstvu

velikost:	1B	proměnná
obsah:	délka	zpráva

Obrázek 3.4: Formát zprávy fyzické vrstvy



Obrázek 3.5: Komunikační diagram pro přijmutí zprávy z fyzické vrstvy

poskytuje přístup na médium, řídící podvrstva poskytuje potvrzovaný přenos zpráv.

Na linkové vrstvě jsou specifikací WirelessHART definována následující servisní primitiva:

- `TRANSMIT_request` pro přenos dat na ostatní zařízení,
- `TRANSMIT_confirm` pro informaci o úspěšnosti přenosu dat,
- `TRANSMIT_indicate` pro předání informace o nově přijatém paketu síťové vrstvě,
- `FLUSH_request` pro odstranění paketu na linkové vrstvě,
- `FLUSH_confirm` pro informování, zda-li byl paket odstraněn a kdy,
- `DISCONNECT_indicate` pro oznámení o odpojování jiného zařízení ze sítě,
- `PATH_FAILURE_indicate` pro oznámení o selhání cesty k připojenému zařízení,
- `ADVERTISE_indicate` pro indikaci přijmutí paketu typu advertise,
- `NEIGHBOR_indicate` pro informování o připojování nového sousedního uzlu,
- `RECEIVE_indicate` pro přepnutí linkové vrstvy do promiskuitního režimu.

Linková vrstva protokolu WirelessHART je řízena časovými sloty (časovými okénky) pro organizaci komunikace. Tyto časové sloty jsou organizovány do jednotlivých superrámců (superframe). Termín superframe označuje logické spojení slotů do jednoho celku, který se periodicky opakuje. Pro realizaci komunikace mezi sousedními uzly definuje protokol WirelessHART na linkové vrstvě linky, grafy, sousedy a superrámce a k nim příslušné tabulky. Každý superrámec má určitou délku (počet slotů).

Přístupová podvrstva protokolu WirelessHART se skládá z přístupové metody TDMA v kombinaci s mechanismem přeskakování kanálů (channel hopping) a zakazování kanálů (channel blacklisting). [6]

Tabulky linkové vrstvy

Tabulka linek (link table) je určena pro ukládání záznamů o jednotlivých linkách. Každá linka má záznam, kde jsou uvedeny následující informace:

- ▷ LinkId - unikátní identifikátor linky,
- ▷ RefNeighborId - reference do tabulky sousedů,
- ▷ LinkType - typ linky (normal, broadcast, join, discovery),
- ▷ TxLink - indikace, lze-li použít linku pro vysílání,
- ▷ RxLink - indikace, lze-li použít linku pro příjem,
- ▷ SharedLink - indikace, je-li linka sdílena více zařízeními,
- ▷ Slot - číslo slotu v superrámci
- ▷ ChannelOffset - posunutí (offset) kanálu pro výpočet aktivního kanálu v mechanismu přeskakování kanálů (channel hopping)
- ▷ RefSuperframeId - reference do tabulky superrámců pro zjištění, kterému superrámci linka náleží.

Na základě informací uvedených v tabulce linek se realizuje vysílání. Nejprve se určí aktuální časový slot. Ten se poté vyhledá v tabulce linek a ze záznamu v tabulce se rozhodne, bude-li se vysílat nebo přijímat, na kterém kanálu bude slot realizován, s kterým sousedním uzlem a ve kterém superrámci. V záznamu linky byl přidán index do tabulky superrámců (RefSuperframeId) pro zjednodušení implementace.

Tabulka superrámců (superframe table) uchovává informace o jednotlivých superrámcích. Záznam o superrámci obsahuje tyto informace:

- ▷ SuperframeId - unikátní identifikátor superrámce,
- ▷ NumSlots - počet slotů v superrámci (velikost superrámce),
- ▷ ActiveFlag - příznak určující, je-li superrámec aktivní,
- ▷ bCastQueue - fronta zpráv, které mají být vyslány na linkách typu broadcast náležejících tomuto superrámci

Ze záznamu superrámce byla vynechána položka Links, která představovala seznam linek umístěných v superrámci a byla přidána položka bCastQueue popsaná výše.

Tabulka sousedů (neighbor table) obsahuje informace o sousedních uzlech sítě. Záznam tabulky sousedů obsahuje následující informace:

- ▷ NeighborId - unikátní identifikátor sousedního uzlu (pětibajtová adresa),
- ▷ NeighborNickname - dvoubajtová adresa sousedního uzlu,
- ▷ JoinPriority - schopnost připojovat další uzly do sítě,
- ▷ TimeSourceFlag - označení sousedního uzlu, podle kterého se má uzel synchronizovat,
- ▷ Status - informace o dosažitelnosti sousedního uzlu,
- ▷ queue - fronta zpráv pro sousední uzel čekající na potvrzení

V záznamu tabulky sousedů byly vynechány položky BOExp a BOCntr používané pro komunikaci po sdílených linkách a také položky LastTimeComunicated, PathFailureTimer, AvgRSL, PacketsTransmitted, MissedAckPackets, PacketReceived, BroadcastsReceived používané pro zaznamenání statistických údajů.

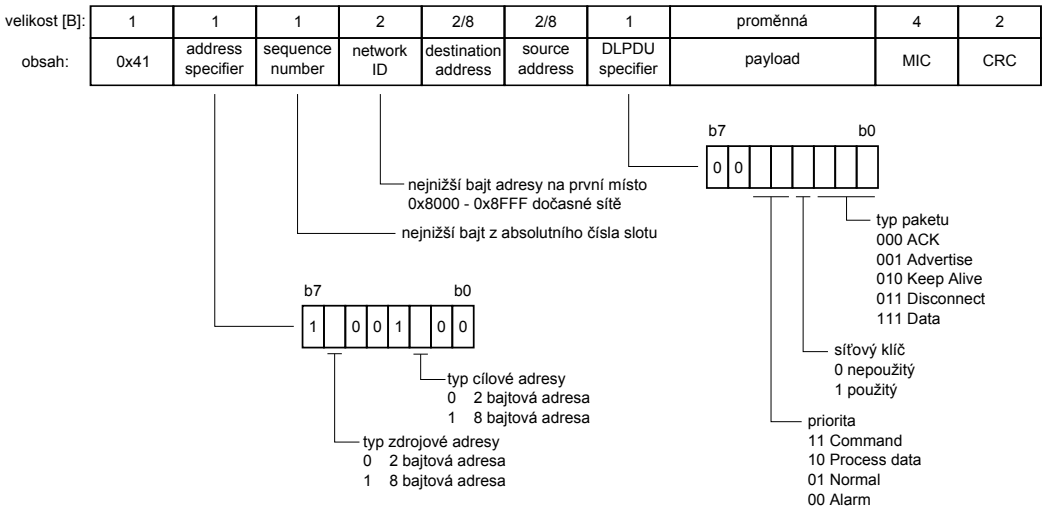
Tabulka grafů (graph table) obsahuje informace o grafech. Záznam v tabulce grafů obsahuje následující informace:

- ▷ GraphId - unikátní identifikátor grafu,
- ▷ RefNeighborId - index do tabulky sousedů

Byly vynechány volitelné položky destUniqueId a destNickname a byla přidána položka RefNeighborId, která je použita pro identifikaci sousedního uzlu pro předání zprávy.

Formát paketů linkové vrstvy

Linková vrstva používá 5 typů paketů. Základem je obecný paket linkové vrstvy uvedený na obrázku 3.6.



Obrázek 3.6: Obecný formát paketu linkové vrstvy

Obecný paket linkové vrstvy začíná znakem s hexadecimální hodnotou `0x41`.

Za tímto znakem následuje jednobajtová specifikace adresy (address specifier), kde se bitem b2 určuje typ cílové adresy a bitem b6 typ zdrojové adresy. V obou případech značí logická nula adresu dvoubajtovou, logická jednička adresu osmibajtovou.

Následuje jednobajtové sekvenční číslo (sequence number), které je odvozeno z nejnižšího bajtu absolutního čísla slotu.

Další položkou paketu je dvoubajtový identifikátor sítě (network ID). Nejnižší bajt tohoto identifikátoru je v paketu uložen jako první, nejvyšší jako druhý.

Následuje adresa cílového uzlu (destination address) a adresa zdrojového uzlu (source address). Podle specifikace adresy jsou tyto adresy dvoubajtové nebo osmibajtové. Nejnižší bajt adresy je vždy první.

Paket dále pokračuje jednobajtovým specifikátorem obsahu paketu (DLPDU specifier). Bity b6 a b7 jsou rezervovány pro budoucí použití a musejí být nastaveny na logickou nulu. Bity b5 a b4 jsou vyhrazeny pro určení priority. Hodnoty priority jsou následující:

- 11 - Command,
- 10 - Process data,
- 01 - Normal,
- 00 - Alarm.

Bit b3 značí použití síťového klíče, kde hodnota 0 znamená, že klíč není použit. Hodnota 1 znamená, že klíč použit je. Bity b2, b1 a b0 jsou použity pro identifikaci typu obsahu paketu. Hodnoty těchto bitů jsou následující:

- 000 - ACK (potvrzení)
- 001 - Advertise
- 010 - Keep Alive
- 011 - Disconnect
- 111 - Data

Další částí paketu je jeho obsah (payload) proměnné délky. Po tomto obsahu následuje čtyřbajtový kód pro ověření integrity zprávy (MIC - message integrity code) a dvoubajtový kontrolní součet (CRC - cyclic redundancy check) pro ověření neporušenosti zprávy.

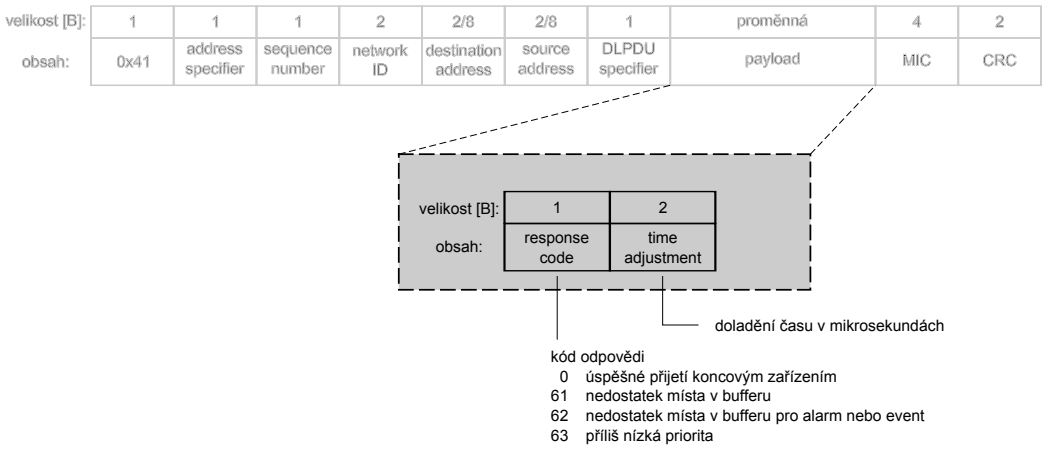
Datový paket linkové vrstvy má v obsahu data vyšších vrstev, která linková vrstva předává vyšší vrstvě.

Potvrzovací paket (ACK - acknowledgment) obsahuje jednobajtový kód odpovědi, který může nabývat následujících hodnot:

- 0 - koncové zařízení v pořádku přijalo paket,
- 61 - zařízení nemá dostatek místa pro uložení přijatého paketu do bufferu, paket je zahozen,
- 62 - zařízení nemá dostatek volného místa v bufferu pro alarm nebo pro event, paket je zahozen,
- 63 - paket má příliš nízkou prioritu, je zahozen.

Další položkou potvrzovacího paketu je dvoubajtové doladění času. Odpovídající čas v μs je uložen jako dvoubajtové celé číslo ve dvojkovém doplňku. Hodnota je kladná resp. záporná, je-li paket přijat dříve resp. později než je očekáváno. Formát tohoto paketu je na obrázku 3.7.

Keep alive paket neobsahuje žádné informace a je používán pro časovou synchronizaci a udržování sítě.



Obrázek 3.7: Formát potvrzovacího paketu linkové vrstvy

Advertise paket je používán pro předání informací potřebných pro zahájení procesu připojení uzlu do sítě.

Na prvních pěti bajtech je absolutní číslo slotu (absolute slot number) uložené od nejnižšího bajtu po nejvyšší.

Následuje jeden bajt pro řízení připojování (join control), který obsahuje na bitech b0 - b3 informaci o prioritě připojení a na bitech b4 - b7 informaci o podporované úrovni zabezpečení. Priorita připojení značí schopnost zařízení připojit další zařízení do sítě.

Další položkou advertise paketu je počet bitů mapy kanálů (number of bits of channel map). Tato položka má velikost jednoho bajtu. Pro fyzickou vrstvu IEEE 802.15.4 je hodnota této položky 16.

Následuje obecně několikabajtová mapa kanálů (channel map). Pro fyzickou vrstvu IEEE 802.15.4 je tato mapa dvoubajtová. Každý bit mapy určuje, je-li daný kanál používán (logická jednička) nebo nikoliv (logická nula).

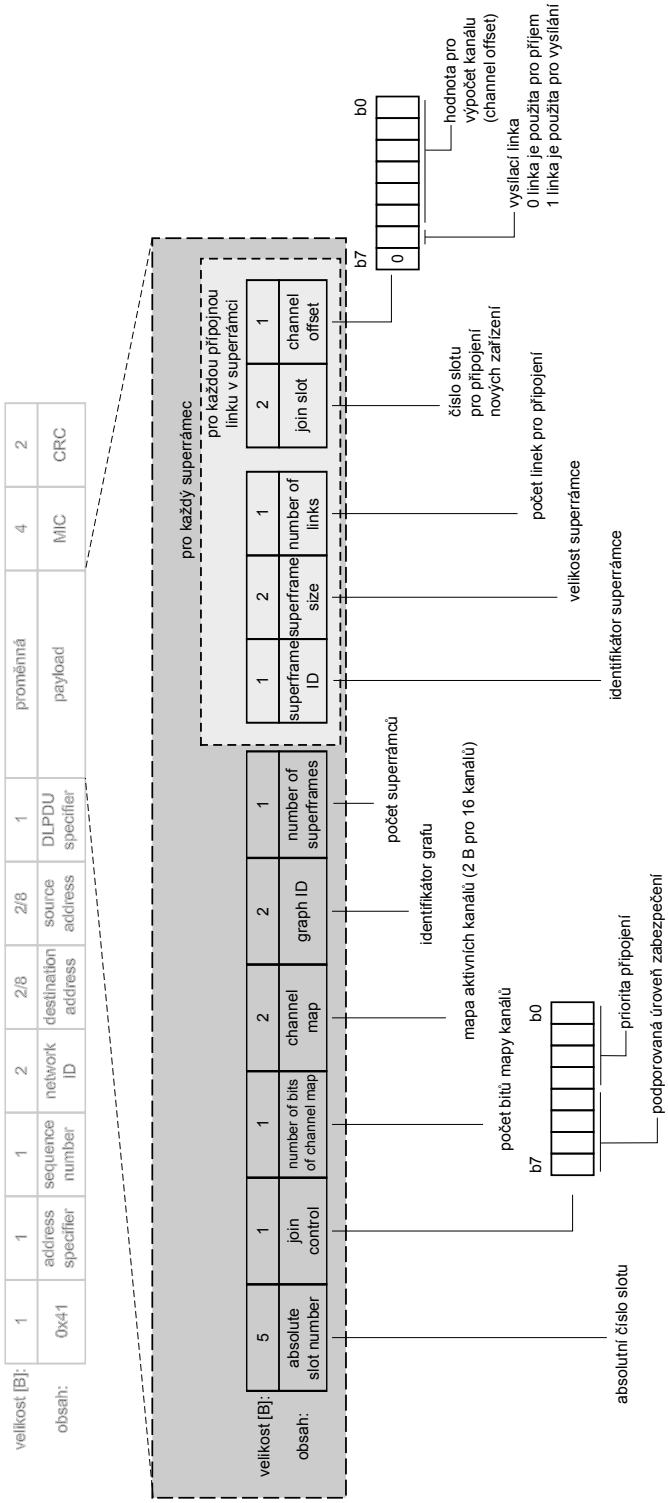
Další položkou paketu je identifikátor grafu (graph ID). Tento identifikátor určuje číslo grafu, které použije připojující se zařízení pro komunikaci s network managerem. Identifikátor grafu je dvoubajtový.

V paketu je dále uveden jednobajtový počet superrámců (number of superframes).

Pro každý superrámec je v paketu záznam, který obsahuje jednobajtový identifikátor superrámce (superframe ID), dvoubajtovou velikost superrámce v počtu slotů (superframe size) a jednobajtový počet linek typu join (number of links).

Pro každou linku typu join je v paketu záznam, který obsahuje dvoubajtovou identifikaci slotu (join slot) a jednobajtové posunutí (channel offset).

Formát advertise paketu je na obrázku 3.8



Obrázek 3.8: Formát advertise paketu

Disconnect paket neobsahuje žádné informace. Je používán pro odpojování uzlů ze sítě. Po přijetí tohoto paketu musí uzel odstranit záznamy spojené se zdrojem uvedeným v paketu, tj. odstranit zdrojový uzel z tabulky sousedů a odstranit všechny linky používané pro komunikaci s tímto uzlem.

Sestavení sítě

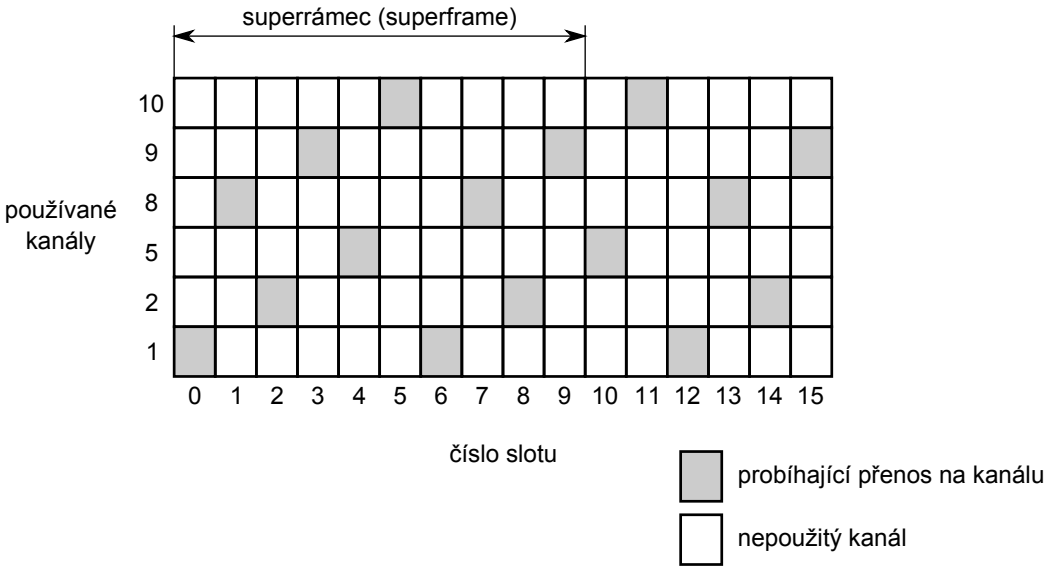
Základním prvkem sítě je network manager. S network managerem naváže spojení brána (gateway) a také přístupový bod (access point) za účelem získání konfigurace. Okamžik, kdy si přístupový bod nastaví *absolutní číslo slotu* (absolute slot number) na nulu, je na výrobci přístupového bodu. Je však potřeba, aby byl tento časový okamžik zaznamenán network managerem, který jej spojí s příslušným časem v reálném světě. Přístupový bod spustí časovač s periodou 10 ms. S každým tikem časovače inkrementuje absolutní číslo slotu. Absolutní číslo slotu obsahuje 5 bajtů, tudíž nastane jeho přetečení přibližně až po 348 letech provozu (viz 3.1). Jakmile vyšle přístupový bod zprávu pro připojení do sítě (advertise), mohou se připojovat další uzly.

$$t_{asn} = 2^{5 \cdot 8} \cdot 10^{-2} = 10995116277,76[s] \doteq 348,4[let] \tag{3.1}$$

Připojení uzlu do sítě

Jednou z nejsložitějších věcí je u protokolu WirelessHART připojení nového zařízení do sítě. Zařízení musí nejprve odchytit zprávu typu advertise, ve které je uvedeno celé absolutní číslo slotu, seznam aktivních kanálů, číslo grafu vedoucího k network manageru a také seznam linek, které jsou použity pro připojení nových zařízení. Kvůli mechanismu přeskakování kanálů (channel hopping) a zakazování kanálů (channel blacklisting) musí zařízení postupně zkoušet přijímat pakety na všech kanálech. Po přijmutí paketu typu advertise již má zařízení dostatek informací pro připojení do sítě. Připojení do sítě probíhá ve slotu typu join, kde zařízení v síti očekává pakety od připojujících se zařízení. Připojující se zařízení vyšle takto zprávu network manageru, který je zodpovědný za přidání linek pro připojující se zařízení a také za konfiguraci příslušných tabulek linkové úrovně a klíče pro šifrování komunikace. Tím je zařízení připojeno do sítě. Pro připojení do sítě může zařízení používat předem nakonfigurovaný klíč nebo známý klíč (well known key).

Tento mechanismus připojování uzlů do sítě umožňuje network manageru udržovat zařízení v karanténě a neprozradit tak žádné další informace o síti.



Obrázek 3.9: Ukázka metody přeskakování kanálů v kombinaci se zakázáním několika kanálů

Přeskakování a zakazování kanálů

Protokol WirelessHART je pro odolnost proti rušení vybaven metodou přeskakování kanálů (channel hopping) a zakazování kanálů (channel blacklisting) (viz obr. 3.9).

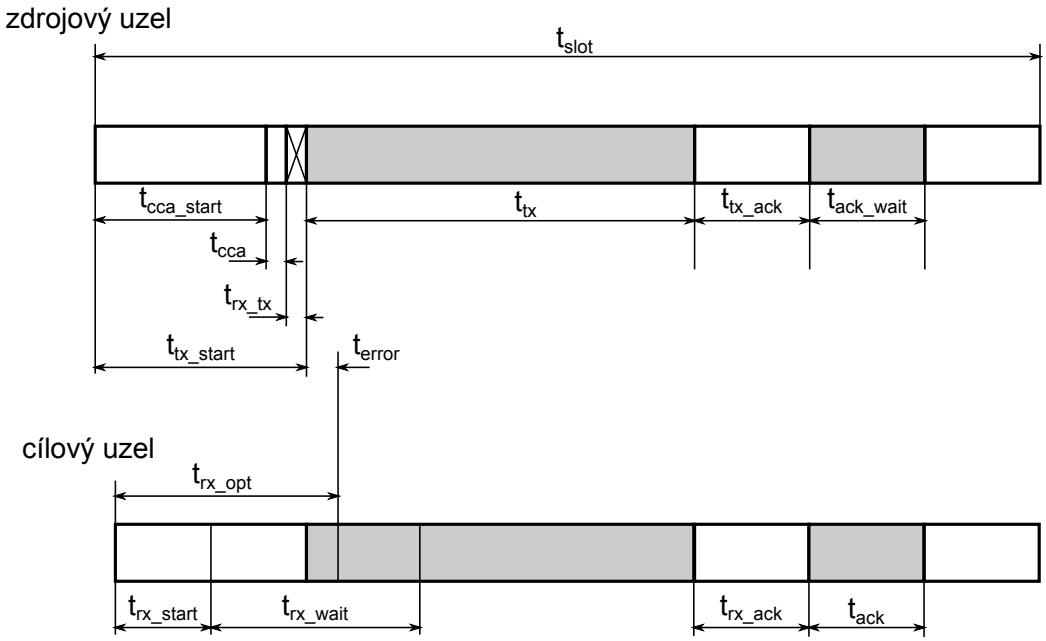
Podstata zakazování kanálů je následující. Každý uzel si udržuje seřazený seznam aktivních kanálů. Tento seznam může network manager měnit příkazy protokolu HART a tak zakázat nebo povolit v síti používání určitých kanálů.

Přeskakování kanálů se řídí vzorcem 3.2. Kde k_i představuje index do pole aktivních kanálů, k_o představuje posunutí (offset) kanálu, A představuje absolutní číslo slotu (absolute slot number) a k_n představuje počet aktivních kanálů.

$$k_i = (k_o + A) \pmod{k_n} \tag{3.2}$$

Časování slotu

Časování slotu je na obrázku 3.10. Čas t_{slot} označuje časový úsek jednoho slotu. V případě protokolu WirelessHART je délka tohoto slotu 10 ms. Čas t_{cca_start} je doba, kterou vysílací uzel čeká na zjištění, je-li kanál volný. Čas t_{cca} je doba, po kterou vysílací uzel naslouchá než začne vysílat. Čas t_{rx_tx} je prodleva nutná pro přechod z příjmu na vysílání. Čas t_{tx_start} označuje



Obrázek 3.10: Časování slotu

dobu od počátku slotu, po které vysílací uzel začne vysílat paket. Doba t_{tx} je doba, po kterou zdrojový uzel vysílá zprávu. Doba t_{tx_ack} je doba, po které musí vysílací uzel naslouchat a doba t_{ack_wait} je doba, kterou vysílací uzel čeká na potvrzení.

U cílového uzlu označuje doba t_{rx_start} dobu, po které má začít naslouchat, t_{rx_wait} dobu, po kterou má čekat na příchozí začátek rámce, t_{rx_ack} dobu po které má začít vysílat potvrzení a t_{ack} dobu vysílání potvrzení. Čas t_{rx_opt} označuje čas, kdy by měl být přijat začátek rámce, pokud by byla síť perfektně synchronizována. Čas t_{error} označuje chybu synchronizace.

Časová synchronizace

Přístupová metoda TDMA protokolu WirelessHART definuje striktní 10ms časové sloty. Během jednoho slotu musí být uzel sítě schopen vyslat zprávu a přijmout potvrzení resp. přijmout zprávu a vyslat potvrzení. Aby spolu mohly jednotlivé uzly komunikovat, je nutné synchronizovat časové sloty v sousedních uzlech a tudíž v celé síti. Časová synchronizace je stěžejní pro správnou funkci celého systému a má velký vliv na spotřebu energie jednotlivých uzlů.

Na časovou synchronizaci jsou kladeny následující požadavky:

- vysoká přesnost,

- minimální spotřeba energie,
- rychlá konvergence.

Potřeba časové synchronizace je dána výrobní tolerancí krystalových oscilátorů, které se obvykle vyrábí s tolerancí ± 10 *ppm*. Pokud má procesor frekvenci oscilátoru 16 MHz je skutečná frekvence oscilátoru 16 MHz ± 160 Hz. Nejhorší možný případ je tedy ten, že jeden uzel sítě bude mít frekvenci oscilátoru 16 MHz -160 Hz a druhý 16 MHz $+160$ Hz. Maximální časový rozdíl $t_{slotDiff}$ během 10ms časového slotu je tedy přibližně $0,2$ μs (viz výpočet 3.3). Podle výpočtu 3.4 tak k posunutí časových slotů o délku jednoho slotu dojde za 500 s.

$$t_{slotDiff} = (\frac{1}{16000000 - 160} - \frac{1}{16000000 + 160}) \cdot 160000 \doteq 2 \cdot 10^{-7}\ s = 0,2\ \mu s \tag{3.3}$$

$$t_{desync} = \frac{t_{slot}}{t_{slotDiff}} \cdot t_{slot} = 500\ s \tag{3.4}$$

Z výše uvedeného plyne, že je nutné uzly sítě synchronizovat.
Pro popis časové synchronizace je nutné vymezit následující pojmy:

- synchronizační uzel - uzel, který je označen jako zdroj časové synchronizace pro jiný uzel sítě,
- synchronizovaný uzel - uzel, který se synchronizuje podle synchronizačního uzlu,
- clock skew - chyba způsobená vzájemným posunutím začátku periody dvou časovačů,
- clock drift - chyba způsobená rozdílnou frekvencí dvou oscilátorů.

Cílem časové synchronizace je naladění časovače synchronizovaného uzlu na frekvenci časovače synchronizačního uzlu s minimální chybou. Časovou synchronizaci lze realizovat doladovacími registry oscilátoru nebo změnou hodnoty čítače.

Uzly sítě si vzájemně vyměňují pakety. Uzly si vždy vyměňují zprávu a potvrzení. Pokud zprávu odesílá synchronizační uzel, odvozuje synchronizovaný uzel synchronizační odchylku od času přijetí paketu. Odesílá-li zprávu synchronizovaný uzel, zašle mu synchronizační uzel odchylku v potvrzovacím paketu.

Pro správnou synchronizaci celé sítě je nutné, aby synchronizační uzly tvořili stromovou strukturu. Tím má každý synchronizovaný uzel určen právě jednoho souseda pro časovou synchronizaci. Synchronizační uzly sítě volí centrálně network manager.

Nejjednodušší možností synchronizace dvou uzlů je posunem začátku periody časovače. Tato metoda je zobrazena na obrázku 3.11. Uzel `A` je synchronizační, uzel `B` je synchronizovaný. Uzel `A` má pevně danou periodu hodnotou čítače V_A. Tato hodnota odpovídá času t_A. Uzel `B` má na začátku synchronizace periodu čítače také nastavenu na hodnotu V_A. Tato hodnota, kvůli rozdílným frekvencím oscilátorů na obou uzlech, odpovídá času t_B. Paket vyslaný uzlem `A` je zachycen uzlem `B` v čase t_{RX_B}, kterému odpovídá hodnota časovače V_{RX_B}. Odchylka od ideální hodnoty časovače (V_{RX_I}) v čase příjmu se použije pro určení hodnoty přetečení časovače (V_{B_c}) podle vztahu 3.5. Tento postup se opakuje každou periodu časovače.

$$V_{B\_c} = V_A \cdot \frac{V_{RX\_B}}{V_{RX\_I}} \tag{3.5}$$

Výše uvedený postup má následující nevýhody:

- nepřesnost a chyby při výpočtech v plovoucí řádové čárce,
- vyšší spotřeba energie pro výpočty s plovoucí řádovou čárkou,
- delší čas potřebný pro výpočet.

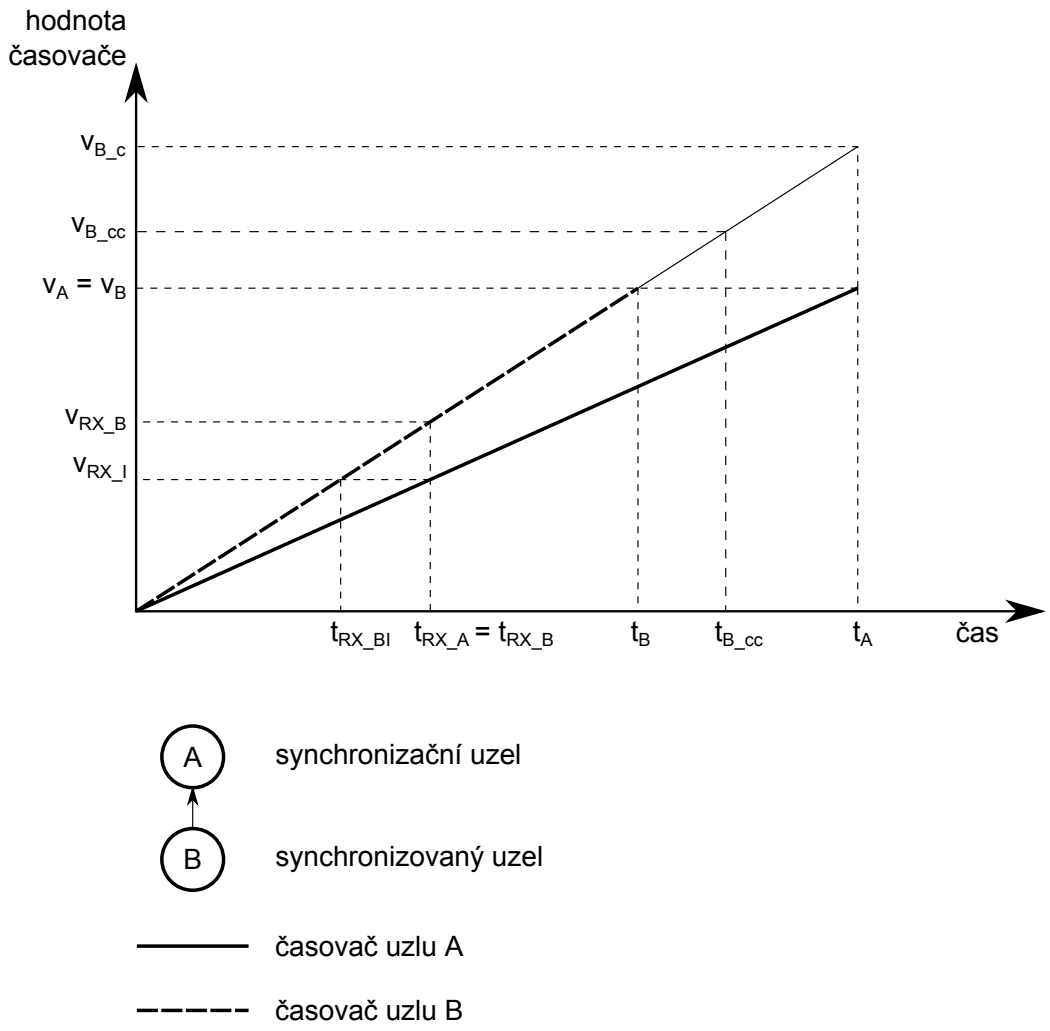
Nepřesnost a chyby při výpočtech jsou zapříčiněny malým rozdílem hodnot V_{RX_B} a V_{RX_I} u téměř synchronizovaných uzlů. Potom poměr $\frac{V_{RX\_B}}{V_{RX\_I}}$ vyjde číslo velmi blízké 1. Při výpočtech s plovoucí řádovou čárkou může nastat situace, že výše zmíněný poměr vyjde 1 i v případě že se ve skutečnosti 1 nerovná. Tato situace je zapříčiněna zaokrouhlovacími chybami při výpočtech s plovoucí řádovou čárkou.

Na obrázku 3.11 je zobrazen celočíselný posun začátku periody čítače. Hodnota čítače V_{B_cc} je vypočítána podle vztahu 3.6.

$$V_{B\_cc} = V_B + (V_{RX\_B} - V_{RX\_I}) \tag{3.6}$$

Celočíselný posun se pouze přibližuje ideální periodě čítače, ale přibližuje se i v případě synchronizování dvou téměř synchronizovaných uzlů. Operace sčítání a odčítání celých čísel jsou rychlejší a méně náročné na energii než operace násobení a dělení čísel s plovoucí řádovou čárkou.

V reálné síti probíhá přenos paketů jednou za několik časových slotů. Rozdíl period časovačů se tak zvětšuje úměrně počtu slotů, kdy uzly nekomunikují. Proto je nutné k posunu periody čítače přidat ještě změnu velikosti



Obrázek 3.11: Celočíselný posun začátku periody časovače

periody. Opravenou periodu T_B časovače synchronizovaného uzlu vypočítáme podle vztahu 3.7.

$$T_B = V_B + \frac{(V_{RX\_B} - V_{RX\_I})}{N_{ASN}} \tag{3.7}$$

V_B je původní hodnota časovače, V_{RX_B} je hodnota časovače v okamžiku přijmutí paketu od synchronizačního uzlu, V_{RX_I} je ideální hodnota časovače v okamžiku přijmutí paketu, N_{ASN} je počet slotů od poslední synchronizace.

3.3.3 Síťová vrstva

Programové vybavení síťové vrstvy je realizováno v kitu `EVALBOT`.

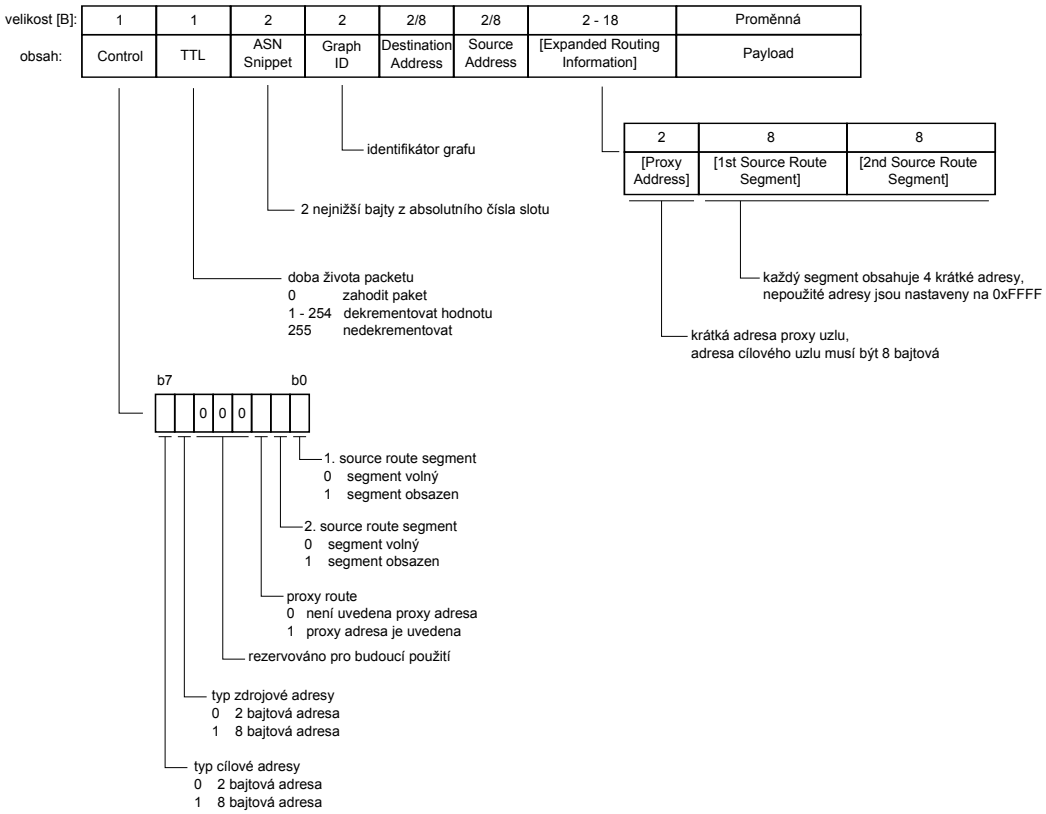
Na síťové vrstvě jsou definována následující servisní primitiva:

- `TRANSMIT_request` pro přenos dat na jedno nebo více zařízení v síti,
- `TRANSMIT_confirm` pro vrácení odpovědi na zaslaný paket,
- `TRANSMIT_indicate` pro předání informace o nově přijatém paketu aplikační vrstvě,
- `TRANSMIT_response` pro zaslání odpovědi koncového zařízení na příslušný paket,
- `FLUSH_request` pro odstranění paketu,
- `FLUSH_confirm` pro indikaci odstranění paketu,

Síťová vrstva musí reagovat na události z linkové vrstvy i na události z aplikační vrstvy. Linková vrstva generuje událost přijmutí paketu (`TRANSMIT_indicate`), aplikační vrstva generuje požadavky na odeslání paketu (`TRANSMIT_request` a `TRANSMIT_response`).

Pro nezávislou obsluhu všech tří událostí obsahuje síťová vrstva frontu událostí. Při příchodu události od linkové nebo aplikační vrstvy je příslušná událost vložena do fronty událostí, kde čeká na obsluhu. Task síťové vrstvy čeká na událost ve frontě událostí. Pokud není ve frontě žádná událost, je tento task pozastaven. Podle přijmuté události task vyvolá obsluhu dané události.

Při události `TRANSMIT_indicate` vyvolané z linkové vrstvy je podle pole Transport Byte rozhodnuto, jedná-li se o odpověď na zadaný příkaz nebo o požadavek na vykonání příkazu. Jedná-li se o požadavek na vykonání příkazu, je příkaz vložen do fronty příkazů, kterou zpracovává task aplikační vrstvy.



Obrázek 3.12: Obecný formát paketu síťové vrstvy

Jedná-li se o odpověď na požadavek, je příkaz i s jeho datovou částí předán na aplikační vrstvu prostřednictvím servisního primitiva `TRANSMIT_confirm`.

Při události `TRANSMIT_request` je k datové části přidána hlavička transportní a síťové vrstvy. Tento paket je předán na linkovou vrstvu k vyslání.

Pro událost `TRANSMIT_response` je třeba v poli Transport Byte nastavit bit pro indikaci odpovědi. Takto upravený paket se již může opatřit hlavičkou síťové vrstvy a odeslat na linkovou vrstvu.

Formát paketu síťové vrstvy

Síťová vrstva používá formát paketu uvedený na obrázku 3.12.

První bajt paketu (Control) obsahuje řídící informace. Je rozdělen po bitech a význam jednotlivých bitů je následující:

- b7 - informace o délce cílové adresy,
- b6 - informace o délce zdrojové adresy,

- b5 - b3 - rezervováno pro budoucí použití,
- b2 - informace, je-li uvedena proxy adresa,
- b1 - informace, je-li obsažen v poli expanded routing information druhý seznam adres (source route segment),
- b0 - informace, je-li obsažen v poli expanded routing information první seznam adres (source route segment).

Druhý bajt paketu obsahuje dobu života paketu (Time To Live). Tato hodnota je snižována o jedna v každém zařízení, kterým paket prochází. Pokud je hodnota doby života nula, je paket zahozen. Je-li hodota 255 (`0xFF` v šestnáctkovém zápisu čísla), nesmí se tato hodnota snižovat.

V paketu následuje dvoubajtové pole ASN Snippet, které obsahuje nejnižší dva bajty z absolutního čísla slotu. Toto pole plní funkci časového razítka. Je nastaveno odesílatelem. Příjemce si tak v čase příjmu paketu může spočítat dobu, kterou paket strávil v síti.

Další položkou paketu je dvoubajtový identifikátor grafu (Graph ID). Podle tohoto identifikátoru se realizuje směrování podle grafu.

Dvě další položky paketu jsou adresa cíle (Destination Address) a adresa zdroje (Source Address). Tyto adresy mohou být dvoubajtové nebo osmibajtové. O délce adres rozhoduje první bajt paketu (Control).

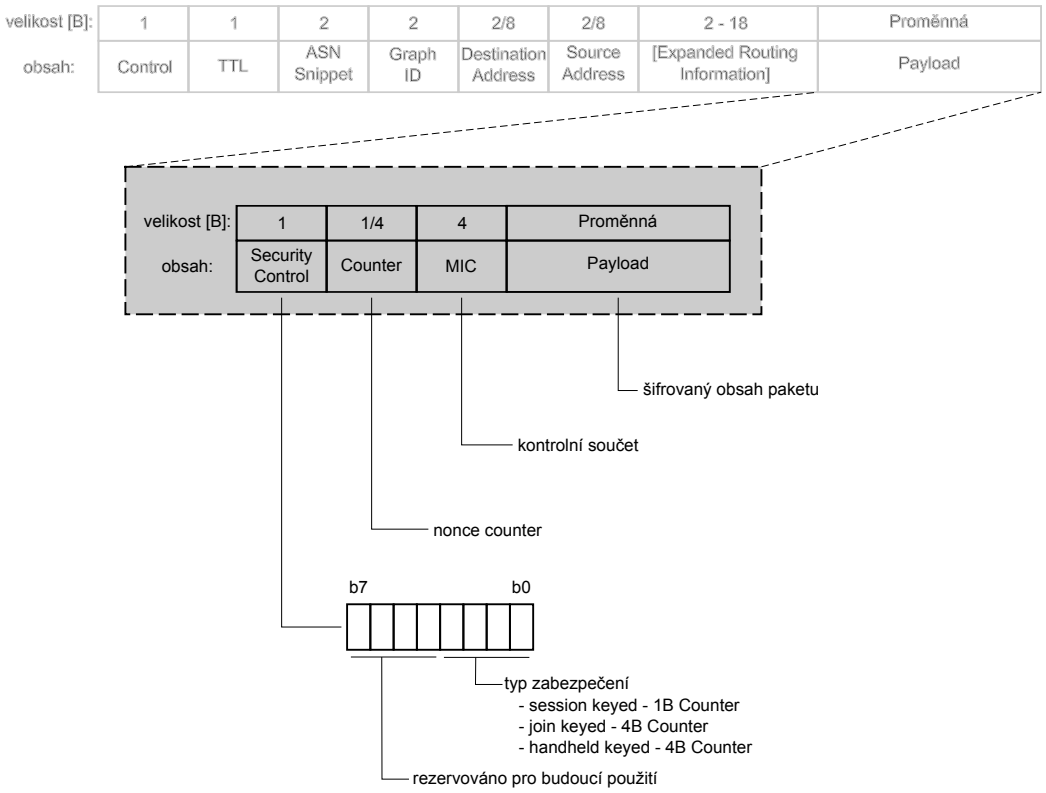
Následuje položka rozšířené směrovací informace (Expanded Routing Information). Délku této položky určuje první bajt paketu (Control). Položka může obsahovat dvoubajtovou adresu proxy zařízení a dva segmenty dvoubajtových (krátkých) adres pro směrování podle zadané cesty. Směruje-li se pomocí proxy uzlu, musí být adresa cíle osmibajtová. Každý segment pro směrování podle zadané cesty obsahuje čtyři dvoubajtové adresy, kde jsou nepoužité adresy nastaveny na hodnotu `0xFFFF`.

Formát paketu bezpečnostní podvrstvy

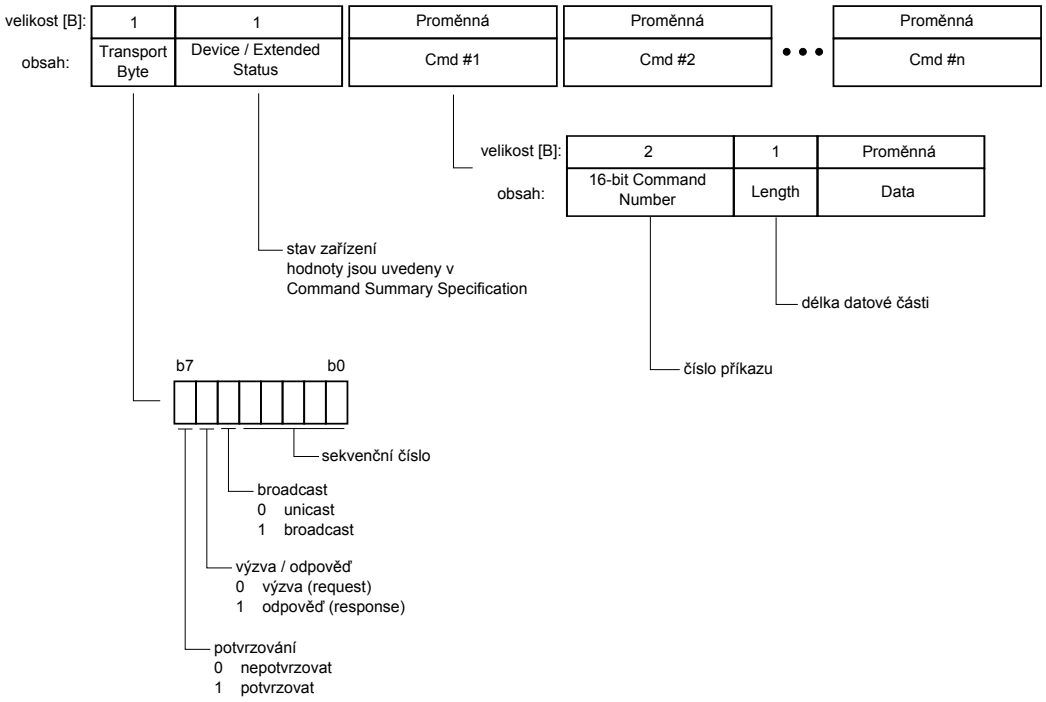
Bezpečnostní podvrstva síťové vrstvy používá formát paketu uvedený na obrázku 3.13.

Na prvním místě je v paketu jednobajtová položka Security Control. Nižší čtyři bity slouží k určení klíče pro dešifrování zprávy. Vyšší čtyři bity jsou rezervovány pro budoucí použití.

Další položka může být jednobajtová nebo čtyřbajtová. Délka této položky závisí na typu zabezpečení zvoleném v položce Security Control. Je-li zvolena možnost zabezpečení relačním klíčem (session keyed), má tato položka velikost jednoho bajtu a obsahuje nejnižší bajt čítače noncí (nonce



Obrázek 3.13: Formát paketu bezpečnostní podvrstvy



Obrázek 3.14: Formát paketu přenášeného síťovou vrstvou

counter). Nonce je jednorázová hodnota používaná v kryptografii pro obranu proti útokům založeným na přehrání zprávy odeslané dříve. Je-li zvoleno zabezpečení klíčem pro připojení nebo klíčem pro komunikaci s příručním zařízením, je položka Counter čtyřbajtová a obsahuje hodnotu čítače noncí.

Poslední položkou je šifrovaný obsah paketu. Tento obsah je zašifrován klíčem vybraným v položce Security Control.

Formát paketu transportní vrstvy

Formát paketu transportní vrstvy je zobrazen na obrázku 3.14.

První položkou paketu transportní vrstvy je Transport Byte. Tato položka je rozdělena po bitech následovně:

- b7 - bit indikující potvrzení paketu,
- b6 - bit pro rozpoznání, jedná-li se o výzvu (request) nebo odpověď (response),
- b5 - bit pro zjištění, jedná-li se o paket typu broadcast nebo unicast,
- b4 - b0 - bity rezervované pro sekvenční číslo.

Druhou položkou paketu transportní vrstvy je Device / Extended Status. Obsah této položky je uveden ve specifikaci *Command Summary Specification*. Tato položka slouží pro indikaci stavu koncového zařízení. Tuto položku obsahuje každý paket transportní vrstvy.

Následuje seznam příkazů. Každý příkaz obsahuje dvoubajtové číslo příkazu, jednobajtovou informaci o délce datové části a datovou část určené délky.

Směrování

Na linkové vrstvě existují dva druhy směrování - směrování podle zadané cesty a směrování podle grafu.

Pro směrování podle zadané cesty je seznam dvoubajtových adres uveden v hlavičce paketu síťové vrstvy v poli 1st Source Route Segment, případně 2nd Source Route Segment.

Pro směrování podle grafu je v hlavičce paketu uvedeno číslo grafu.

3.3.4 Aplikační vrstva

Aplikační vrstvu tvoří dva tasky - obslužný a klientský. Obslužný task se stará o vykonávání příkazů a generování odpovědí na ně. Úkolem klientského tasku je obsluha událostí od uživatele, generování požadavků pro obslužný task koncového zařízení a zobrazení odpovědí od koncového zařízení.

Aplikační vrstva obsahuje dvě fronty příkazů - frontu příkazů čekajících na vykonání a frontu vykonaných příkazů čekajících na odeslání sítí.

Obslužný task čeká na příkazy z fronty příkazů čekajících na vykonání. Po odebrání příkazu z fronty task podle čísla příkazu (Command Number) vyvolá obsluhu daného příkazu. Vygenerovanou odpověď uloží do fronty příkazů čekajících na odeslání sítí a vyvolá servisní primitivum `TRANSMIT_response`. Vyvoláním tohoto servisního primitiva se vloží událost do fronty událostí síťové vrstvy.

Klientský task na základě uživatelského vstupu vygeneruje příkaz a předá jej servisním primitivem `TRANSMIT_request` síťové vrstvě. Data potřebná pro odeslání příkazu na zadanou adresu jsou předávána v datové struktuře události. Po inicializaci události je tato událost vložena do fronty událostí síťové vrstvy.

4 Realizace navrženého systému

Navržený systém obsahuje celkem čtyři tasky, které spolu vzájemně komunikují. Navržený systém tasků je na obrázku 4.1. Pro správnou funkci systému je nutné tasku linkové vrstvy (`dataLinkLayerTask`) přiřadit nejvyšší prioritu, tasku síťové vrstvy (`networkLayerTask`) nižší prioritu a aplikačním taskům (`clientTask` a `applicationLayerTask`) prioritu nejnižší.

4.1 Fyzická vrstva

Kvůli nedůvěře v překladač je programové vybavení fyzické vrstvy realizováno v jediném zdrojovém souboru. Program běží v nekonečné smyčce, kde se řídí aktuálním stavem globální proměnné `control_flag`, která je nastavena v obsluze přerušení od SPI na aktuálně zpracovávaný příkaz. Při hodnotě `CTRL_RF_RX` spustí rutinu pro radiový příjem a po dokončení rutiny nastaví `control_flag` na hodnotu `CTRL_NONE`. Při hodnotě `CTRL_RF_TX` spustí rutinu pro vyslání dat a při hodnotě `CTRL_SLEEP` spustí rutinu pro přechod do režimu nízké spotřeby. Z režimu nízké spotřeby je procesor vzbuzen vnějším přerušením od SPI (pin SS - slave select).

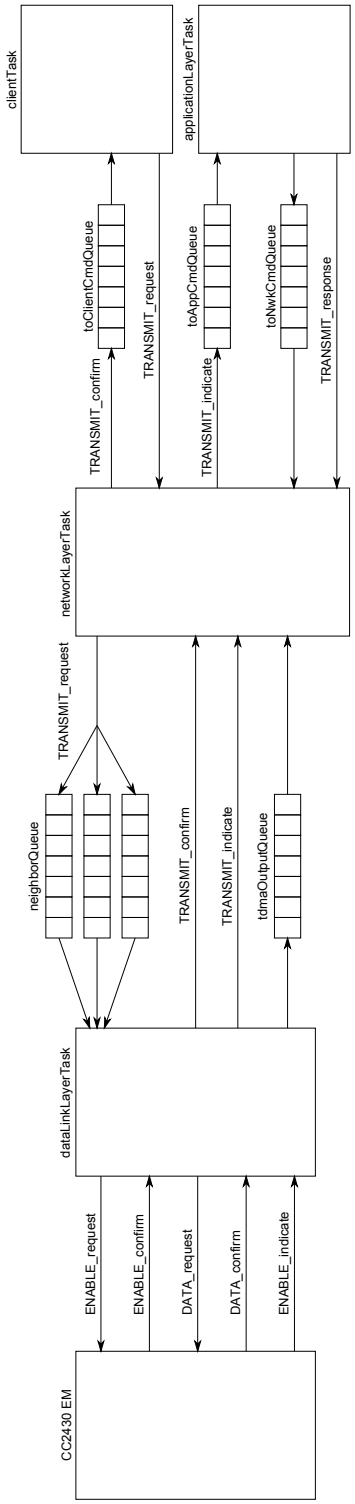
Při příjmutí příkazu `CTRL_RF_TX_SAVE` je uložena hodnota prvního znaku zprávy (délka zprávy) do proměnné `saved_len` a zároveň je uložena do bufferu vysílacího koprocesoru. Dokud nejsou přečteny všechny znaky zprávy jsou přijaté znaky ukládány do bufferu vysílacího koprocesoru. Po příjmu posledního znaku se nastaví `control_flag` na hodnotu `CTRL_NONE` a vnější stav na `READY`.

Při příjmu zprávy zůstává přijatá zpráva taktéž v bufferu koprocesoru, odkud je postupně čtena přes SPI. Tímto bylo docíleno malých nároků na paměť procesoru a možnosti překládat programové vybavení v „memory model small“.

4.2 Linková vrstva

Pro ukládání paketů byl vytvořen buffer (packet pool), který uchovává pakety a informace o nich. Struktura uchovávaných dat obsahuje následující informace:

- status - informace, je-li místo pro paket volné nebo obsazené,
- phyHdr - pointer na hlavičku fyzické vrstvy,



Obrázek 4.1: Navržený systém

- dllHdr - pointer na hlavičku linkové vrstvy,
- nwkHdr - pointer na hlavičku síťové vrstvy,
- payload - pointer na obsah paketu,
- packet - uložený paket.

Buffer je implementován jako staticky alokované pole záznamů s pevnou velikostí. Přístup k paketům je realizován prostřednictvím funkcí:

- `packetPoolInit` - pro inicializaci bufferu (paket poolu),
- `packetPoolIsFull` - pro zjištění, je-li buffer plný,
- `acquirePacket` - pro rezervování místa v bufferu pro nově vytvářený paket,
- `savePacket` - pro uložení paketu přijmutého z fyzické vrstvy,
- `getPacket` - pro přístup k paketu,
- `releasePacket` - pro uvolnění místa pro nový paket.

4.2.1 Přístupová podvrstva

Implementace přístupové podvrstvy je v souboru `w-hart_dll_mac.c`. Jsou zde vytvořeny funkce:

- `macInit` - pro inicializaci přístupové vrstvy,
- `macTimerIsr` - obslužná rutina pro přerušení od časovače,
- `waitForSlot` - rutina pro čekání na začátek časového slotu,
- `macTimerPrepare` - pro inicializaci časovače,
- `macTimerEnable` - pro spuštění časovače,
- `sfdReceived` - pro obsluhu události přijmutí začátku rámce,
- `dataReceived` - pro obsluhu události přijatého paketu,
- `isPacketReceived` - pro zjištění, je-li přijat paket,
- `resetPacketReceivedFlag` - pro resetování příznaku přijatého paketu,

proměnná	hodnota	význam
`txPreambleAndSfdTicks`	2560	doba potřebná pro odvysílání preambule a značky začátku rámce
`idealTxTimeTicks`	18190	ideální čas pro vyslání paketu
`macTimerPeriod`	160000	perioda časovače odpovídající 10 ms
`phyTxDelay`	9250	zpoždění fyzické vrstvy na pokyn vyslání paketu
`maxRxWaitDelay`	60000	maximální čas po který se čeká na příjem paketu
`rxToAckDelay`	1000	časové zpoždění pro vyslání potvrzení
`lastAckRxTime`	140000	doba, do kdy se čeká na potvrzení paketu

Tabulka 4.1: Proměnné pro časování slotu

- `getLastPacketIndex` - pro zjištění, na jakém místě v bufferu paktů je poslední přijatý paket,
- `waitForAdvertise` - pro periodický příjem paketů, dokud není přijat advertise paket,
- `parseAdvertise` - pro zpracování přijatého advertise paketu,
- `waitForKeepAlive` - pro periodický příjem paketů, dokud není přijat keep-alive paket,
- `syncPacketReceived` - pro realizaci časové synchronizace,
- `waitForTxInSlot` - pro čekání na okamžik vyslání paketu,
- `macTimerSynchronizePrepare` - pro inicializaci časovače před připojením do sítě,
- `macTimerPeriodSet` - pro změnu periody časovače,
- `synchronizeTimerToNetwork` - pro synchronizaci časovače se sítí.

Časování slotu

Pro časování slotu jsou na přístupové podvrstvě definované proměnné uvedené v tabulce 4.1.

Délka slotu je určena proměnnou `macTimerPeriod`. Po spuštění časovače je v obslužné rutině přerušení nastavena hodnota přetečení na

`macTimerPeriod`. Aktuální hodnota přetečení časovače se mění funkcí `macTimerPeriodSet`.

Časová synchronizace

Časová synchronizace je realizována na přístupové podvrstvě. Kořen synchronizačního stromu má pevně danou periodu časovače. Tato perioda je určena proměnnou `macTimerPeriod` staticky při překladu. Po zapnutí je nastavena hodnota přetečení a je spuštěn časovač. Tento časovač generuje přerušení s periodou 10 ms. Zdrojem hodinového signálu pro časovač je externí krystalový oscilátor. Frekvence tohoto oscilátoru je 16 $MHz \pm 50$ ppm.

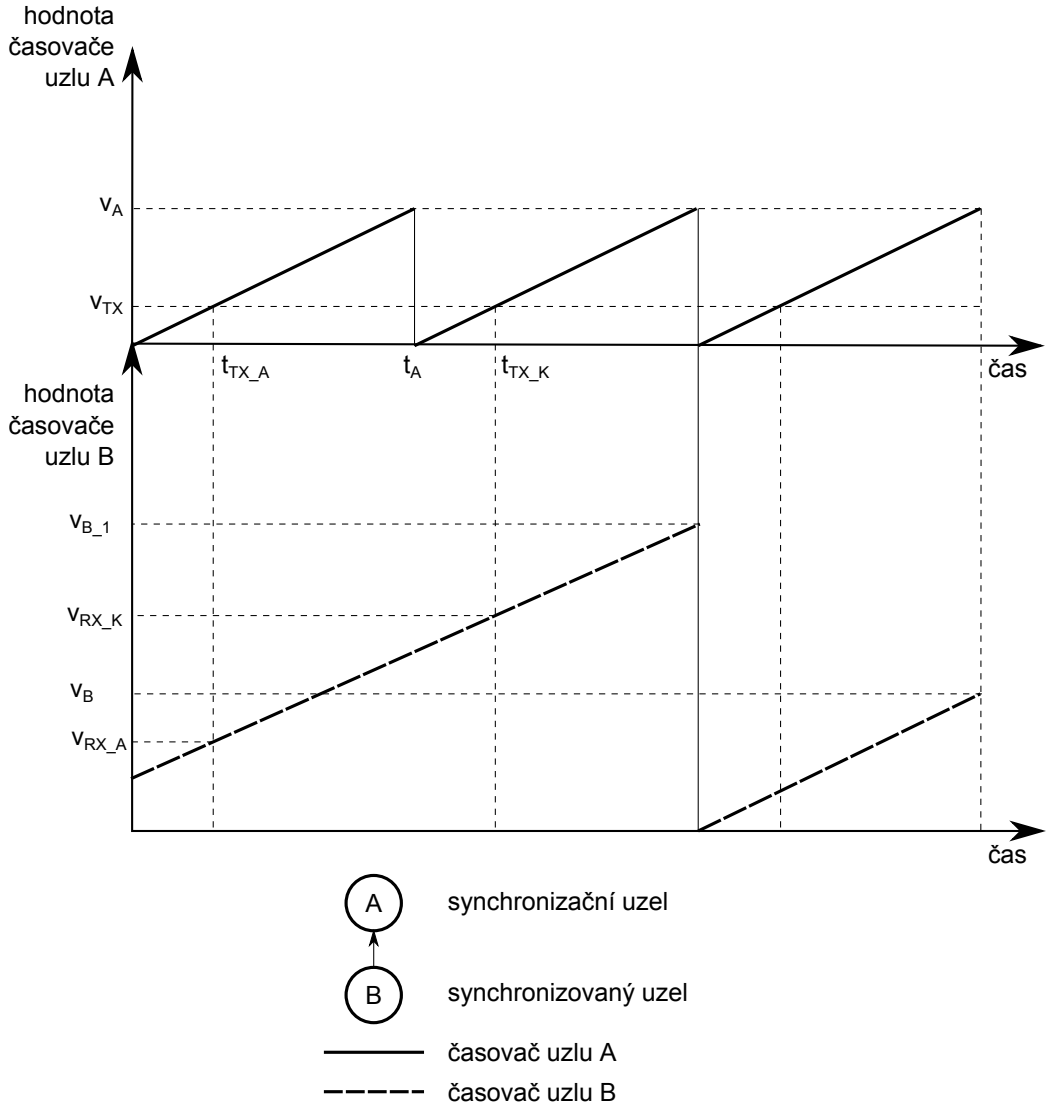
Synchronizace uzlu k dané síti vychází z předpokladu, že ve slotu bezprostředně následujícím po odeslání advertise packetu je odeslán keep alive paket. Způsob prvotní synchronizace je na obrázku 4.2. Uzel spustí časovač, naslouchá a čeká na přijmutí advertise paketu. Čas příjmu advertise paketu (V_{RX_A}) si uloží do proměnné `tAdvertise`. Podle absolutního čísla slotu v přijatém advertise paketu nastaví absolutní číslo slotu na řídící podvrstvě. Poté naslouchá a čeká na přijmutí keep alive paketu. Čas přijmutí keep alive paketu (V_{RX_K}) si uloží do proměnné `tKeepAlive`. Z těchto dvou hodnot určí periodu časovače `macTimerPeriod`, která bude použita v obslužné rutině přerušení pro nastavení periody časovače. Aktuální periodu časovače sníží na hodnotu V_{B_1} podle vztahu 4.1.

$$V_{B\_1} = V_{RX_K} + (V_{RX_K} - V_{RX_A}) - V_{TX} - V_{txPreSfd} - V_{phyTxDelay} \qquad (4.1)$$

V_{RX_K} je hodnota časovače při přijmutí keep alive paketu, V_{RX_A} je hodnota časovače při přijmutí advertise paketu, V_{TX} je hodnota časovače pro okamžik vyslání paketu, $V_{txPreSfd}$ představuje dobu nutnou pro přijmutí preambule a značky začátku rámce, $V_{phyTxDelay}$ představuje dobu reakce fyzické vrstvy na pokyn vyslání paketu.

Tímto je uzel připojen k síti a musí udržovat synchronizaci se synchronizačním uzlem. Pro udržování synchronizace je na přístupové podvrstvě implementována funkce `syncPacketReceived`, která je volána z řídící podvrstvy vždy po přijetí packetu od synchronizačního uzlu. V této funkci je realizována časová synchronizace podle vztahu 3.7. Funkce má dvě lokální proměnné:

- `absError` - pro výpočet chyby synchronizace,
- `absAsnDiff` - počet slotů od poslední synchronizace,
- `idealRxTimeTicks` - hodnota časovače v ideaálním čase příjmu.



Obrázek 4.2: Způsob prvotní synchronizace uzlu k síti

V globální proměnné `sfdRxTime` je uložena hodnota čítače v okamžiku přijmutí značky začátku rámce. Ideální čas přijmutí začátku rámce (`idealRxTimeTicks`) je vypočítán sečtením ideálního času pro vyslání paketu (`idealTxTimeTicks`), času potřebného pro vyslání preambule a značky začátku rámce (`txPreambleAndSfdTicks`) a zpoždění fyzické vrstvy (`phyTxDelay`).

Hodnota ideálního času přijmutí začátku rámce (`idealRxTimeTicks`) je porovnána s hodnotou čítače v okamžiku přijetí značky začátku rámce (`sfdRxTime`).

Pokud je paket přijmut dříve než by měl být (`sfdRxTime` je menší než `idealRxTimeTicks`), musí se perioda časovače zkrátit. Hodnota `absError` je vypočítána odečtením `sfdRxTime` od `idealRxTimeTicks`, aktuální perioda časovače je zkrácena o `absError` a hodnota `macTimerPeriod` je snížena o `absError`/`absAsnDiff`.

Pokud je paket přijmut později (`sfdRxTime` je větší než `idealRxTimeTicks`), musí se perioda časovače prodloužit. Hodnota `absError` je vypočítána odečtením `idealRxTimeTicks` od `sfdRxTime`, aktuální perioda časovače je prodloužena o `absError` a hodnota `macTimerPeriod` je zvýšena o `absError`/`absAsnDiff`.

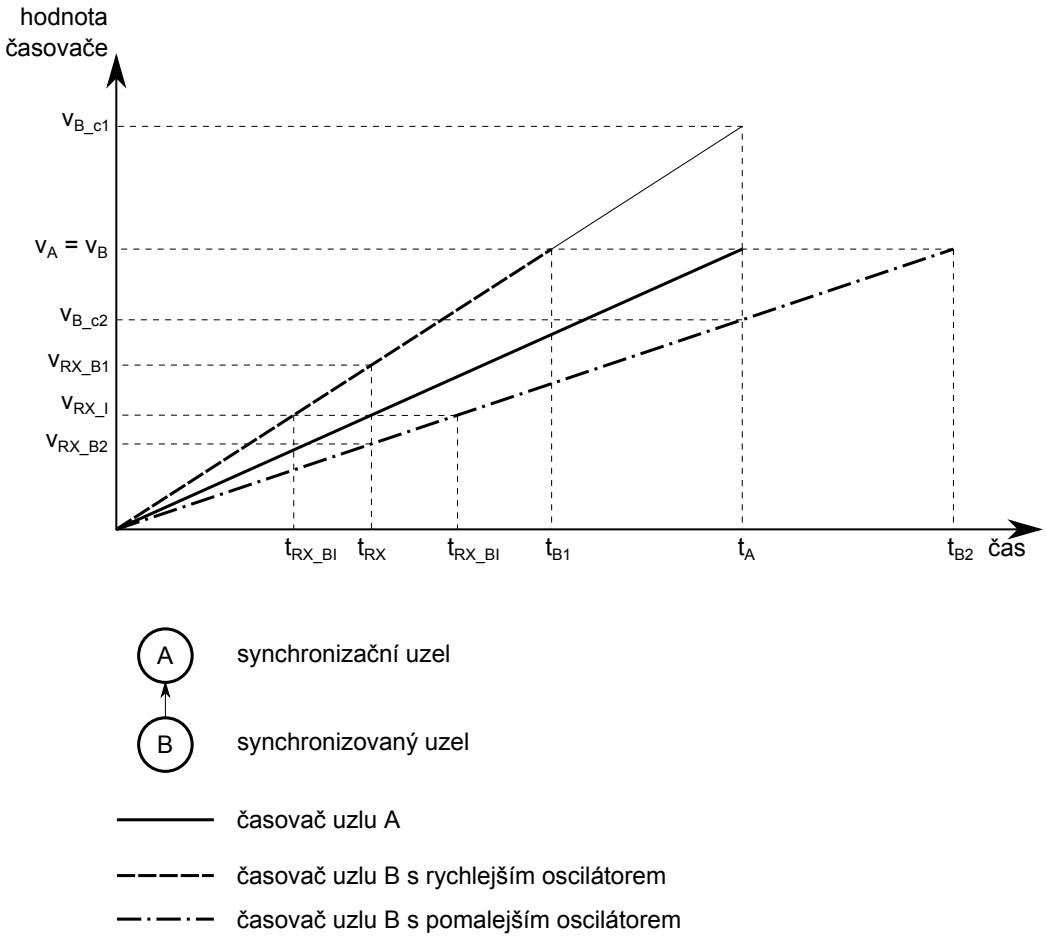
Logika změny periody na základě naměřené odchylky je vidět na obrázku 4.3, kde v čase t_{RX} zachytí synchronizovaný uzel zprávu od synchronizačního.

Tento čas reprezentuje synchronizovaný uzel s rychlejším oscilátorem jako hodnotu čítače V_{RX_B1}, která je větší než hodnota ideální (V_{RX_I}). Z pohledu uzlu B se tato situace jeví tak, že přijmul paket později, přesto však musí periodu čítače zvětšit, aby vykompenzoval chybu synchronizace.

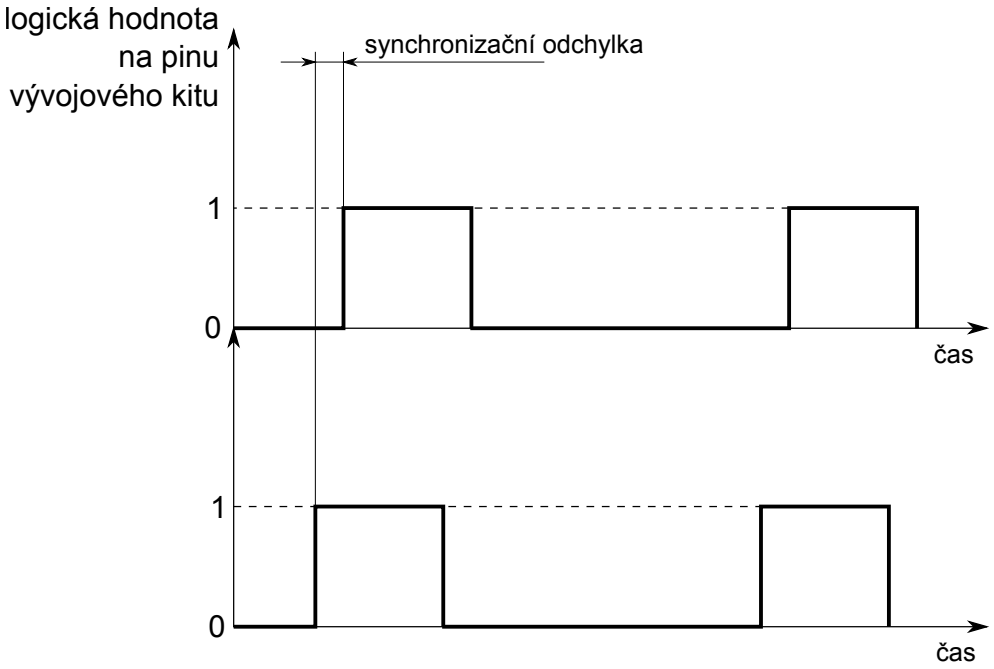
U synchronizovaného uzlu s pomalejším oscilátorem je tomu naopak. Časový okamžik t_{RX} reprezentuje hodnotou čítače V_{RX_B1}, která je menší než ideální hodnota V_{RX_I}. Tudíž se z pohledu uzlu B jeví tato situace tak, že byl paket přijat dříve, přesto je potřeba periodu čítače zmenšit pro vykompenzování chyby synchronizace.

Pro měření časové synchronizace na vytvořené síti je použit logický analyzátor připojený k oběma uzlům. Každý uzel signalizuje vykonávání kódu v obslužné rutině přerušení od časovače logickou jedničkou na jednom z vyvedených pinů (viz obr. 4.4). Záznam se signalizací je exportován programem do formátu `csv`. Takto vytvořený soubor je použit pro vygenerování grafů programem Microsoft Excel. Rozdíl časů začátku vykonávání kódu je v grafu na obrázku 4.5 a detailněji na obrázku 4.6.

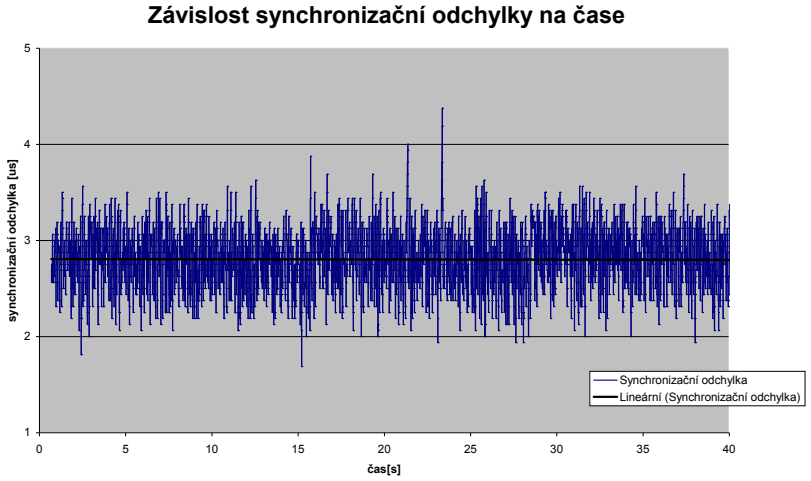
Možnou příčinou zákmitů by mohla být nekonstantní odezva fyzické vrstvy. Graf průběhu odezvy fyzické vrstvy je na obrázku 4.7. Pro zjištění průběhu odezvy fyzické vrstvy jsem také použil logický analyzátor, kterým jsem měřil čas od pokynu vyslání paketu až po přijmutí značky začátku rámce. V



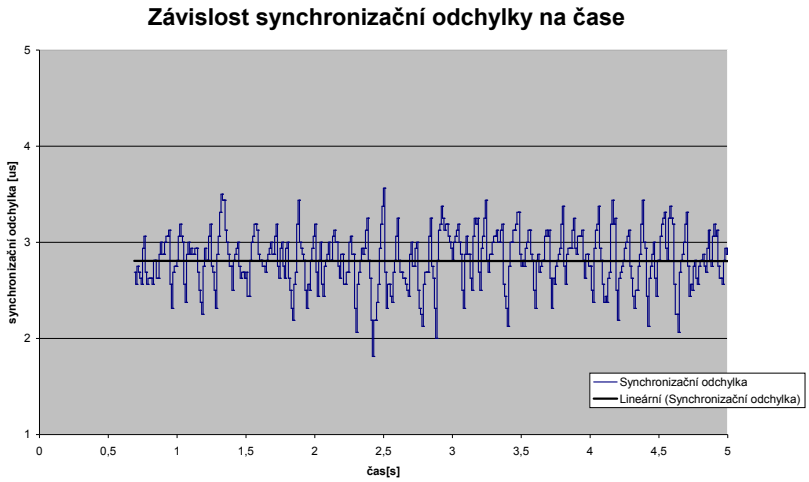
Obrázek 4.3: Logika změny periody v závislosti na čase přijmutí paketu



Obrázek 4.4: Zpracování výstupu logického analyzátoru



Obrázek 4.5: Graf závislosti synchronizační odchylky na čase



Obrázek 4.6: Graf závislosti synchronizační odchylky na čase

ideálním případě by tento čas měl být konstantní.

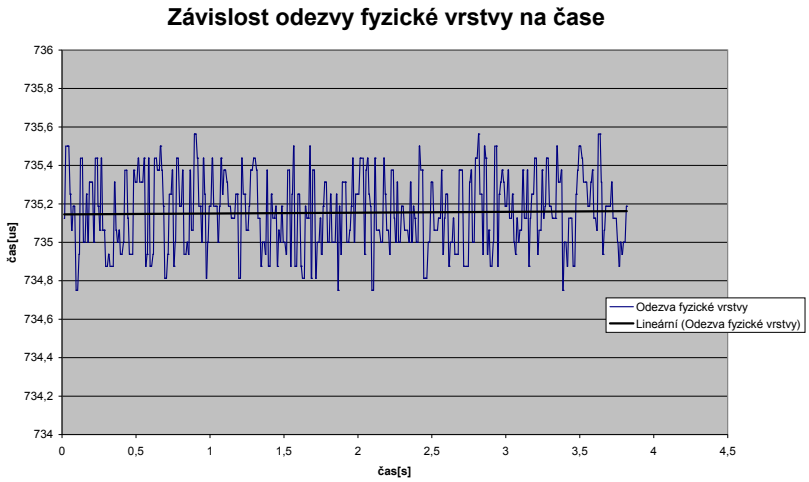
U logického analyzátoru je použita vzorkovací frekvence 16 MHz, tudíž snímá s chybou $\pm 0,0625\ \mu s$ (délka periody hodinového signálu). Nemůže se tudíž jednat o chybu měření v případě měření odezvy fyzické vrstvy.

Další možnou příčinou zákmitů a nepřesnosti v synchronizaci je krystalový oscilátor na kitu `EVALBOT`, který má udávanou frekvenci 16 MHz $\pm$ 50 ppm. Tato nepřesnost může při periodě 10 ms vytvořit chybu až $\pm 0,5\ \mu s$.

Díky těmto chybám může docházet k nepravidelnosti při vysílání paketů a tím i k zákmitům při synchronizaci uzlu. Přesto uvedený způsob synchronizace vykazoval velmi dobrou stabilitu i při dlouhodobých testech. Synchronizace byla testována po dobu až 24 hodin. I po této době se synchronizační odchylka nelišila od výše uvedené.

4.2.2 Řídící podvrstva

Implementace řídící podvrstvy je v souboru `w-hart_dll_llc.c`. Řídící podvrstva je implementována jako samostatná úloha (task), která spolupracuje s přístupovou podvrstvou a stará se o vyslání a příjem paketů pro fyzickou vrstvu.



Obrázek 4.7: Graf závislosti odezvy fyzické vrstvy na čase

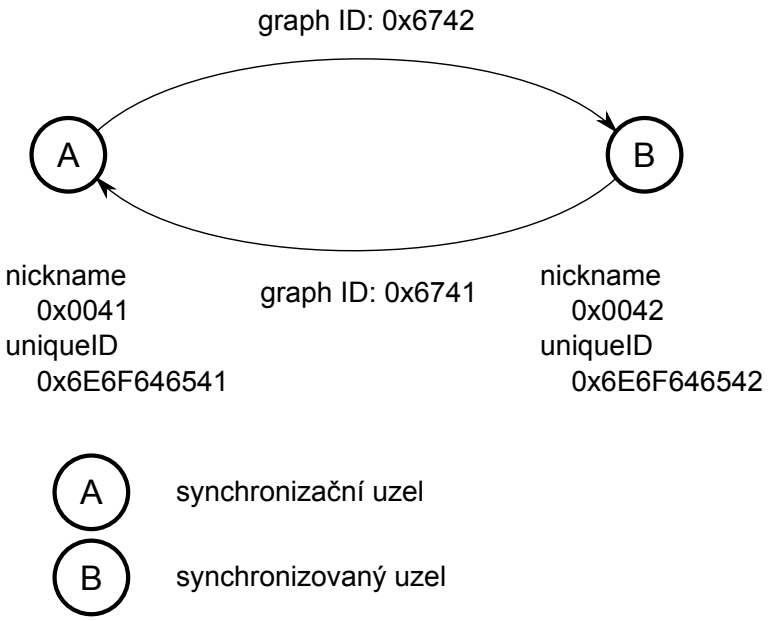
Na řídící podvrstvě jsou implementovány následující funkce:

- `dllInit` - pro inicializaci linkové vrstvy,
- `incrementAsn` - pro zvýšení počítadla slotů,
- `setAsn` - pro nastavení počítadla slotů,
- `getAsn` - pro zjištění aktuální hodnoty počítadla slotů,
- `getMyNickname` - pro zjištění dvoubajtové adresy uzlu,
- `getMyLongAddr` - pro zjištění osmibajtové adresy uzlu,
- `addGraphEntry` - pro přidání záznamu o grafu,
- `removeGraphEntry` - pro odebrání záznamu o grafu,
- `sendGraphMsg` - pro odeslání zprávy podle grafu,
- `addNeighborEntry` - pro přidání záznamu o sousedním uzlu,
- `removeNeighborEntry` - pro odebrání záznamu o sousedním uzlu,

- `sendShortNeighborMsg` - pro odeslání zprávy sousedovi podle dvoubajtové adresy,
- `sendLongNeighborMsg` - pro odeslání zprávy sousedovi podle osmibajtové adresy,
- `addLinkEntry` - pro přidání záznamu o lince,
- `removeLinkEntry` - pro odebrání záznamu o linkce,
- `addSuperframeEntry` - pro přidání záznamu o superrámci,
- `removeSuperframeEntry` - pro odebrání záznamu o superrámci,
- `sendSuperframeMsg` - pro odeslání skupinové (broadcast) zprávy,
- `receivedAckCheck` - pro kontrolu přijatého potvrzení,
- `realizeSlot` - pro realizaci slotu,
- `prepareNormalLink` - pro připravení zařízení na linku typu `NORMAL`,
- `prepareBcastLink` - pro připravení zařízení na linku typu `BROADCAST`,
- `prepareJoinLink` - pro připravení zařízení na linku typu `JOIN`,
- `prepareDiscoveryLink` - pro připravení zařízení na linku typu `DISCOVERY`,
- `prepareForSlot` - pro zahájení přípravy zařízení na nadcházející slot,
- `dataLinkLayerTask` - task pro obsluhu linkové vrstvy,
- `setPacketDllHeader` - pro nastavení ukazatelů na hlavičku linkové a síťové vrstvy.

Tato podvrstva je implementována jako nekonečná smyčka, která vyvolává funkce pro připravení na slot (`prepareForSlot`), čekání na začátek slotu (`waitForSlot`) a pro realizaci slotu (`realizeSlot`).

Funkce `prepareForSlot` má jako parametr absolutní číslo slotu, na který se má linková vrstva připravit. Podle uvedeného parametru vypočítá číslo slotu, který vyhledá v tabulce linek a dále se již řídí záznamem z tabulky linek. Je-li linka typu `NORMAL`, použije ze záznamu linky index souseda, pro zjištění, je-li ve frontě daného souseda uložena nějaká zpráva. Je-li linka typu `BROADCAST` zjistí, jestli je ve frontě k příslušného superrámce zpráva k odeslání. Pokud žádnou nenajde, vyšle paket typu keep-alive. Je-li linka typu



Obrázek 4.8: Topologie sestavené sítě

`DISCOVERY`, zajistí vyslání discovery paketu. Pro linku typu `JOIN` se připraví na příjem.

Funkce `waitForSlot` je funkce přístupové podvrstvy a slouží k realizaci čekání na začátek slotu.

Funkce `realizeSlot` podle čísla slotu vyšle příkaz pro vysílání resp. příjem fyzické vrstvy. Po vyslání paketu přepne fyzickou vrstvu na příjem a čeká na potvrzení. Po příjmu paketu zkontroluje jeho obsah a vyšle potvrzení.

Sestavení sítě

Na obrázku 4.8 je topologie sítě, na které bylo testováno programové vybavení. Bohužel nebylo k dispozici více uzlů sítě, nemohla tak být ověřena přesnost synchronizace pro více než jeden podřízený uzel. Na testované topologii nemohla být ověřena funkčnost směrování, proto je směrování implementováno pouze statickým přeposíláním paketů.

Komunikační linky, dvoubajtové adresy (nickname) i osmibajtové adresy (uniqueID) jsou nastaveny na začátku běhu programu. Pro rozpoznání, které adresy se mají nastavit je použito tlačítko (`SWITCH 1`) na vývojovém kitu `EVALBOT`. Při stisknutém tlačítku je uzel inicializován jako uzel `B`. Stejným způsobem jsou inicializovány tabulky linkové vrstvy.

Uzel `A` představuje bránu sítě, uzel `B` představuje připojovaný uzel. Uzel `A` vyšle paket advertise následovaný v dalším slotu paketem keep alive. Paket

advertise je zachycen uzlem `B`, který z něj zjistí aktuální číslo slotu a synchronizuje periodu svého časovače podle přijatého keep alive paketu. Ostatní údaje již nezjišťuje, neboť jsou inicializovány na začátku běhu programu.

4.3 Síťová vrstva

Síťová vrstva je implementována v souboru `w-hart_nwk.c`, kde jsou vytvořeny funkce:

- `nwkTableInit` - pro inicializaci tabulky podle, které probíhá směrování,
- `nwkInit` - pro vytvoření fronty událostí a úvodní inicializaci síťové vrstvy,
- `processPacketFromTDMAQueue` - pro obsluhu paketu z výstupní fronty linkové vrstvy,
- `addTransportHeader` - pro přidání hlavičky transportní vrstvy,
- `addNwkHeader` - pro přidání hlavičky síťové vrstvy,
- `sendNwkPacket` - pro odeslání síťového paketu na linkovou vrstvu,
- `sendResponsePacket` - pro odeslání odpovědi na příkaz,
- `networkLayerTask` - task pro obsluhu událostí,
- `setPacketPayloadPointer` - pro inicializaci ukazatele na data paketu.

Na síťové vrstvě byla vytvořena fronta událostí, nad kterou čeká task síťové vrstvy. Struktura události obsahuje tyto položky:

- `event` - typ události,
- `pktIndex` - index paketu v bufferu paketů (packet poolu),
- `nickname` - krátká adresa cíle,
- `payload` - ukazatel na obsah paketu.

Jednotlivé položky jsou inicializovány a používány podle typu události. Typ události je výčtový typ a obsahuje hodnoty:

- `NWK_EVENT_TRANSMIT_INDICATE`,
- `NWK_EVENT_TRANSMIT_REQUEST`,

- `NWK_EVENT_TRANSMIT_RESPONSE`.

Při příchodu události `NWK_EVENT_TRANSMIT_INDICATE` zpracuje task síťové vrstvy jeden paket z výstupní fronty linkové vrstvy. Nejprve zkontroluje adresu cíle. Pokud je paket určen jinému uzlu, vyvolá přeposlání paketu servisním primitivem `TRANSMIT_request` sousednímu uzlu. Je-li paket určen pro aktuální uzel, rozhodne se podle bitu pro indikaci požadavku/odpovědi (request/response) (v položce paketu Transport Byte) jaké servisní primitivum vyvolá. Pokud se jedná o požadavek (request), vyvolá se servisní primitivum `TRANSMIT_indicate`. Při odpovědi (response) je vyvoláno servisní primitivum `TRANSMIT_confirm`.

Událost `NWK_EVENT_TRANSMIT_REQUEST` zpracuje task vyvoláním funkce `sendNwkPacket`, která využije nastavené hodnoty `pktIndex`, `nickname` a `payload` pro odeslání síťového paketu na linkovou vrstvu.

Událost `NWK_EVENT_TRANSMIT_RESPONSE` je obsloužena přijmutím provedeného příkazu z fronty od aplikačního tasku. Tento příkaz je poté prostřednictvím funkce `sendResponsePacket` opatřen hlavičkou transportní a síťové vrstvy a je odeslán na linkovou vrstvu.

4.4 Aplikační vrstva

Aplikační vrstva je realizována v souboru `w-hart_app.c`, kde jsou vytvořeny funkce:

- `appInit` - pro inicializaci aplikační vrstvy,
- `applicationLayerTask` - obslužný task aplikační vrstvy.

Funkce `appInit` slouží k vytvoření front aplikační vrstvy. Fronta `toAppCmdQueue` je fronta příkazů čekajících na obsluhu. Fronta `toNwkCmdQueue` je fronta obsloužených příkazů čekajících na odeslání. Fronta `toClientCmdQueue` slouží pro předání zpracovaných příkazů.

Aplikační vrstvu tvoří dva tasky:

- obslužný - pro obsluhu příkazů zaslaných po síti,
- klientský - pro komunikaci s uživatelem, zobrazování a zasílání příkazů.

Obslužný task aplikační vrstvy v nekonečné smyčce čeká na příkazy z fronty `toAppCmdQueue`. Po příchodu příkazu se podle čísla příkazu (Command Number) rozhodne, jakou obsluhu vyvolá. Po vygenerování datové části příkazu pošle tento příkaz do fronty `toNwkCmdQueue` a zavolá servisní primitivum `TRANSMIT_response`.

Klientský task je realizován v souboru `app_tasks.c`. Tento task komunikuje s uživatelem a vytváří příkazy. Tyto příkazy poté odesílá prostřednictvím servisního primitiva `TRANSMIT_request` reprezentovaného funkcí `TRANSMIT_request_nickname`. Odpovědi na zaslané příkazy obsahuje po vykonání a doručení fronta `toClientCmdQueue`. Příkazy z této fronty klientský task zobrazuje uživateli.

4.5 Použité prostředky

Pro vývoj programového vybavení fyzické vrstvy byl použit volně šiřitelný překladač `SDCC` a vývojové prostředí `Eclipse`. Pro naprogramování modulu fyzické vrstvy byla použita deska `SmartRF 04 EB` od Texas Instruments. Programové vybavení bezdrátového modulu `CC2430 EM` zabralo 128 B paměti RAM a 2427 B paměti FLASH.

Pro vývoj programového vybavení linkové, síťové a aplikační vrstvy bylo použito vývojové prostředí `Code Composer Studio` verze 4 a operační systém `FreeRTOS`. Linková, síťová a aplikační vrstva je realizována na vývojovém kitu Stellaris® LM3S9B92 EVALBOT Robotic Evaluation Board (`EVALBOT`) od Texas Instruments. Programové vybavení kitu `EVALBOT` zabralo přibližně 28 KB paměti FLASH a 37 KB paměti SRAM.

Pro ladění byl použit modul `CC2430 DB` od Texas Instruments, který sloužil jako jednokanálový sniffer. Dále byl pro ladění použit osmikanálový logický analyzátor `Saleae Logic Analyzer`, kterým bylo možné snímat časové poměry při časové synchronizaci obou uzlů.

Pro vytváření obrázků byl použit program `InkScape`. Text práce byl vytvořen nástrojem TEX.

5 Závěr

Seznámil jsem se s architekturou protokolu WirelessHART a s operačním systémem FreeRTOS. Navrhl a realizoval jsem programové vybavení pro uzel sítě složený z bezdrátového modulu `CC2430 EM` a vývojového kitu Stellaris® LM3S9B92 EVALBOT Robotic Evaluation Board (`EVALBOT`).

Důsledkem takto složeného uzlu bylo několik problémů při realizaci. Prvním problémem bylo velice obtížné ladění programového vybavení bezdrátového modulu i ladění komunikace s modulem `EVALBOT`. Další problém nastal při časové synchronizaci uzlů důsledkem nekonstantní doby reakce bezdrátového modulu na pokyn vyslání paketu.

I přes výše uvedené problémy byla realizována bezdrátová síť umožňující vzájemnou komunikaci mezi uzly pomocí specifikace WirelessHART. Součástí realizace vzájemné komunikace byla i realizace časové synchronizace s poměrně vysokou stabilitou, rychlostí synchronizace a nízkou výpočetní náročností. Bohužel nebylo možné ověřit tyto vlastnosti na větší síti, protože jsem měl k dispozici pouze dva uzly.

Programové vybavení kitu `EVALBOT` je za daných podmínek přenositelné a realizovatelné na jiném hardwaru s procesorovým jádrem Cortex-M3.

Za největší přínos této práce považuji realizaci algoritmu pro synchronizaci uzlů sítě a návrh programového vybavení umožňující komunikaci uzlu sítě pomocí specifikace WirelessHART.

6 Přehled zkratek

Zkratka	Význam	Vysvětlení zkratky
AES	advanced encryption standard	standard pro symetrické šifrování
CCA	clear channel assessment	zjištění, je-li komunikační kanál volný
MAC	medium access control	přístupová vrstva
O-QPSK	offset quadrature phase-shift keying	způsob modulace fázovým posunem
RAM	random access memory	paměť s přímým přístupem
RSSI	received signal strength indication	síla přijímaného signálu
SDCC	small device C compiler	volně šiřitelný překladač jazyka C (viz `http://sdcc.sourceforge.net/`)
SFD	start of frame delimiter	značka začátku rámce na fyzické úrovni
SPI	serial peripheral interface	sériové periferní rozhraní
TDMA	time division multiple access	plánovaný komunikační protokol
USART	universal synchronous / asynchronous receiver and transmitter	zařízení, které je schopno komunikovat jako SPI i jako UART

Literatura

[1] HCF_SPEC-065. 2.4GHz DSSS O-QPSK Physical Layer Specification. Austin: HART Communication Foundation, 2007.

[2] HCF_SPEC-075. TDMA Data Link Layer Specification. Austin: HART Communication Foundation, 2008.

[3] HCF_SPEC-085. Network Management Specification. Austin: HART Communication Foundation, 2009.

[4] HCF_SPEC-290. WirelessHART Device Specification. Austin: HART Communication Foundation, 2008.

[5] FreeRTOS. Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions. [online]. © 2010-2013 [cit. 2014-04-13]. Dostupné z: http://www.freertos.org/.

[6] GUSTAFSSON, David. WirelessHART- Implementation and Evaluation on Wireless Sensors. KTH Electrical Engineering, 2009. Diplomová práce. KTH Electrical Engineering..

[7] HART Specifications. HART Communication Foundation. [online]. © 2014 [cit. 2014-03-24]. Dostupné z: http://www.hartcomm.org/protocol/about/aboutprotocol_specs.html.

[8] JIANPING, Song, Aloysius MOK, Deji CHEN, Mike LUCAS, Mark NIXON a Wally PRATT. WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control. IEEE Real-Time and Embedded Technology and Applications Symposium. 2008, 377 - 386.