České vysoké učení technické v Praze
Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

Dálkové sledování vozu GSM technologiemi

Praha, 2006 **Autor: Martin Vacula**

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 26. května 2006

Poďakovanie

V úvode by som chcel poďakovať pánovi Ing. Janovi Krákorovi za pomoc a trpezlivé vedenie diplomovej práce.

Ďalej by som chcel menovite podakovať:

- Ing. Ondrejovi Špinkovi za pomoc a cenné rady s programovaním riadiaceho modulu
- Ing. Tomášovi Fenclovi za cenné rady ohladne použitia PPP stacku

Ďalej by som rád poďakoval mojej rodine a priatelom za pomoc a podporu počas štúdia na FEL ČVUT v Praze.

Abstrakt

Cielom diplomovej práce je vytvorit system na dialkové sledovanie vozu pomocou GSM technológii. Teoretická časť popisuje systém a spôsoby jeho realizácie. Dalej sa vysvetlujú protokoly ktoré sú potrebné v našej realizácii na prenos údajou pomocou datoveho spojenia v sieti GSM pomocou GPRS. Pozornosť je venovaná aj modulu prenosovej jednotky a jej softwarovému rozšíreniu. Praktická časť popisuje realizáciu softwarového riešenia stackov pre protokoly PPP, IP a UDP. Popísané sú najdôležitejšie funkcie a ich algoritmy pomocou vývojových diagramov. Ďalej je popísané fyzické riešenie prijímania, ukladania a vysielania dat.

Abstract

The goal of this thesis project is to design a system for vehicle monitoring utilizing the GSM cellular technology. Theoretical part deals with the system itself and the possible implementations. This part also includes a description of communication protocols used in the system for data exchange through GPRS connection in the GSM network. A special attention is given to the data transfer unit and its software exstension.

The functional part describes the software implementation of stacks for PPP, IP and UDP protocols. The most important functions and their algorithms are described using the state diagrams. Furthermore, the hardware solution for data reception, saving and transmission is presented.

České vysoké učení technické v Praze - Fakulta elektrotechnická

Katedra řídicí techniky *Školní rok:* 2004/2005

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Martin V a c u l a

Obor: Technická kybernetika

Název tématu: Dálkové sledování vozu GSM technologiemi

Z á s a d y p r o v y p r a c o v á n í :

1. Seznamte se s s technologiemi GSM, GPRS a komunikačním protokolem CAN.
2. Prostudujte vlastnosti mikroprocesoru Motorola HC12 a modulu XT55 firmy Siemens s GPS, GSM a GPRS technologiemi.
3. Realizujte aplikaci pro čtení dat z automobilu Škoda a aplikaci pro čtení dat z polohovacího modulu s Motorolou HC12 a s modulem XT55.
4. Naprogramujte aplikaci pro příjem stavových dat automobilu do PC.
5. Realizujte aplikaci pro databázový sběr stavových dat a jejich presentaci skrze WWW technologie.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí diplomové práce: Ing. Jan Krákora

Termín zadání diplomové práce: zimní semestr 2004/2005

Termín odevzdání diplomové práce: leden 2006

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

v z.p. prof. Ing. Vladimír Kučera, DrSc.
děkan

V Praze dne 09.03.2005

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

Cieľom diplomovej práce je vytvoriť systém na sledovanie pohybu a stavu automobilu. Práca nadväzuje na predchádzajúce diplomové prace, pri ktorých bol realizovaný hardware a základne ovládače pre prácu s CAN zbernicou. Základnou úlohou je vytvorenie spojenia pomocou GPRS pre prenos dát o stave automobilu a jeho polohe. S toho dôvodu je potrebná realizácia základných stackou pre protokoly použite pre prenose ako sú PPP, IP a UDP. Dáta o stave vozidla sa získavajú zo zbernica CAN automobilu škoda octávia a dáta o polohe prostredníctvom satelitného navigačného systému GPS. Tieto dáta musia byť prijaté, spracovane a vhodným spôsobom vyslané čo kladie nároky na použité riadiaci algoritmus, buffery a spôsob vytvárania paketov.

Kapitola 2

Systém sledovania automobilu

S rozvojom automobilového priemyslu nastala potreba jeho efektívneho riadenia. A to hlavne v prípade autodopravcov a firiem s veľkým vozovým parkom. Reakciou na tuto potrebu vznikli systémy na sledovanie stavu a polohy automobilu. Jedným z najrozšírenejších je systém elektronickej knihy jázd.

Pôvodne kniha jázd bol záznam o stave a pohybu vozidla zapísaný v papierovej podobe. Jeho efektívnosť bola značne obmedzená, pretože záznam bol vytvorený človekom, čím sa do neho mohli zaniesť chyby. Taktiež prenositeľnosť dát v tejto podobe je veľmi obmedzená. S nástupom informačných technológií do automobilového priemyslu sa tieto nedostatky odstránili. V súčasnosti sa jedná o plne automatický systém, ktorý splňuje základné požiadavky:

- Prehľad o polohe a stave automobilu
- Automatické vytváranie knihy jázd
- Znížinie počtu volaní medzi dispečerom a vodičom vozidla
- Znížinie výdajov na provoz systému
- Nerušinie vodiča behom riadenia

2.1 Popis systému

Systém sa skladá z dvoch základných častí. Jednotky, ktorá je umiestnená v automobile, ktorej úlohou je získavať informácie o polohe a stave vozidla. Následne musí byť schopný

tieto dáta spracovať a ukladať do svojej pamäti. V prípade požiadavku alebo naplnenia pamäte musí byť schopná prenášať zvolením spôsobom získané dáta na dispečerské stanovisko.

Na strane dispečera je aplikácia, ktorá je schopná prichádzajúce dáta spracovať a uložiť. Najčastejšie sa používa forma databáze a v pokročilejších systémoch taktiež vykresľovanie polohy vozidla na mape.

Podľa požiadavku na prenos dát sa delia systémy elektronickej knihy jázd na online a offline.

2.1.1 Offline

Pri tejto variante sú dáta ukladané do pamäte jednotky v automobile a ich prenos nastáva až po ukončení jazdy vozidla, tzn. po návrate vozidla do parkovacích priestorov firmy. Počas jazdy nemá dispečer možnosť podrobného sledovania v reálnom čase. Prenos dát je realizovaný vo veľkej väčšine pomocou sériového rozhrania RS 232. V prípade, že jednotka v automobile je schopná prenosu GSM, môžu sa používať SMS správy, ktorými sa prenášajú údaje o polohe a krátke informácie o stave automobilu. V dôsledku malého objemu dát možného prenášať v jednej SMS správe (160 znakov) a finančných nákladov s tým spojených je ale toto riešenie výhodné iba v neobvyklých resp. krízových situáciách.

2.1.2 Online

Táto varianta je schopná nielen ukladať získané dáta do pamäte jednotky v automobile, ale aj ich okamžite prenášať do dispečerského centra. Najpoužívanejším spôsobom prenosu je pomocou GSM a dátových služieb GPRS. Dáta získane o polohe a stavu vozidla sa po spracovaní prenášajú prostredníctvom internetu do dispečerského centra. Dispečer má potom prístup k polohe a stavu vozidla v ľubovolný časový okamžik. Tým môže efektívne meniť trasu vozidla podľa potreby resp. rýchlo reagovať na havarijne situácie. V prípade nemožnosti nadviazania GPRS prenosu resp. jeho cenovej nevýhodnosti (roamingové poplatky zo zahraničia) sa používajú iné služby GSM ako napr. SMS správy.

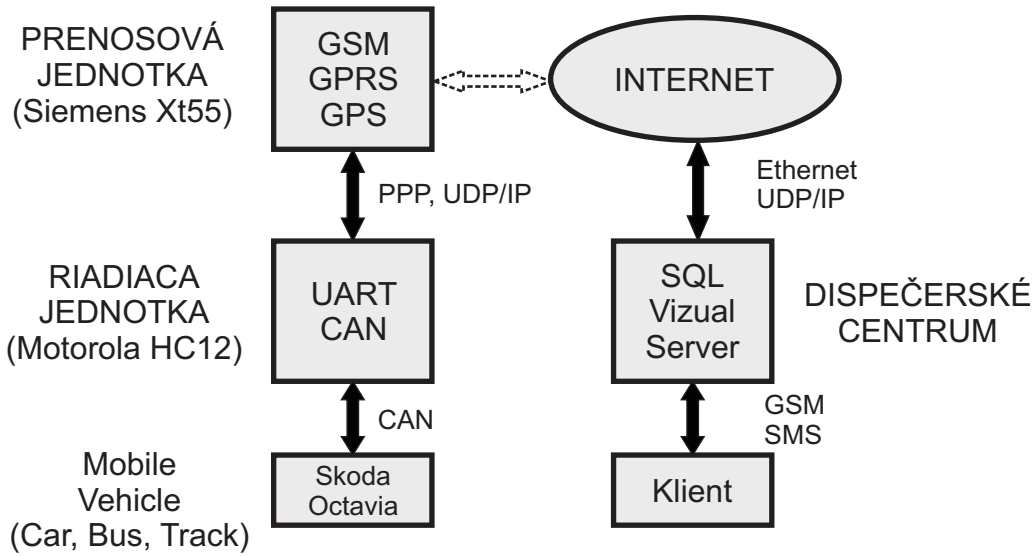
2.2 Realizácia systému

V súčasnosti existuje viacej komerčných riešení systému elektronickej knihy jázd. Každý je vytvorený na základe požiadaviek, ktoré môžu byť na cenu, použitý hardware jednotky v automobile, dát ktorý je systém schopný získať, spôsobom uskladnenia a prenosu týchto dát a realizáciou aplikácie na strane dispečera.

Náš systém elektronickej knihy jázd vychádza z týchto základných požiadaviek:

- Použiť dostupný hardware.
- Možnosť získať informácie o stave automobilu priamo s interných zberníc automobilu.
- Získanie dát o polohe automobilu.
- Realizácia systému Online
- Ukladanie dát na strane dispečera vo forme súboru alebo databáze

Na základe týchto požiadaviek sme zvolili hardwarové riešenie jednotky v automobile s dvoch častí a to riadiacej a prenosovej jednotky. Výsledná štruktúra celého systému je zobrazená na obr. 2.1. Jednotlivé bloky reprezentujú časti hardwaru z ktorého sa systém



Obr. 2.1: Blokové schéma systému.

skladá, pričom ich popis je uvedený vedľa nich. V blokoch samotných sú uvedené technológie, ktoré poskytuje daný hardware a spojnice medzi jednotlivými blokmi popisujú použité komunikačné protokoly.

- **Riadiaca jednotka** - Vývojová doska s mikrokontrolérom Motorola HC12 vytvorená ako diplomová práca pána Miroslava Musila [1]. Táto doska nám umožňuje prístup na internú zbernicu automobilu CAN s ktorej získavame informácie o jeho stave vdaka radiču msCAN12, ktorý obsahuje mikrokontrolér HC12. Taktiež obsahuje externú pamäť RAM veľkosti 32kB na ukladanie získaných dát a externú pamäť ROM typu FLASH veľkosti 32kB slúžiacu na uloženie riadiaceho programu jednotky umiestnenej v automobile (vid kapitola 6). Ďalšie použité periférie pre náš systém sú 2 asynchrónne sériové rozhrania slúžiace na obojsmernú komunikáciu s prenosovou jednotkou.

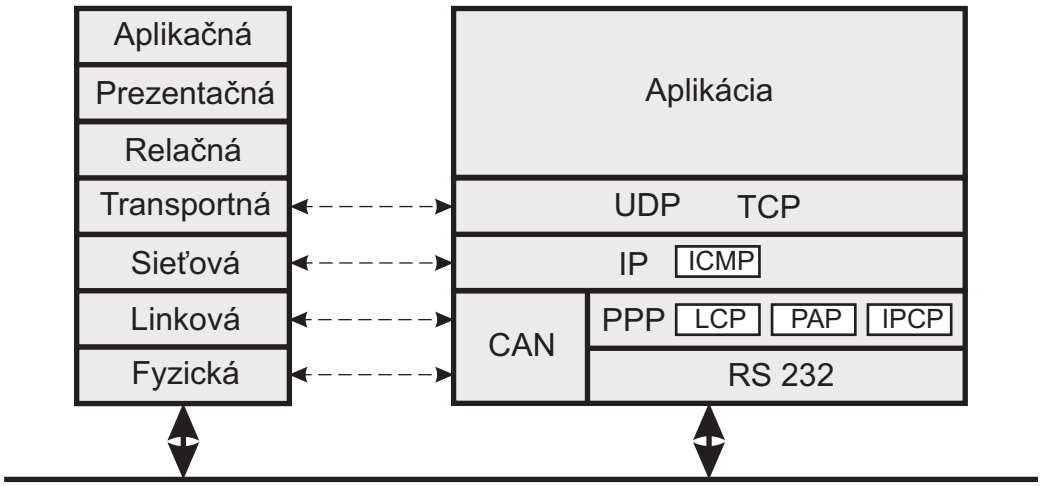
- **Prenosová jednotka** - Vývojová doska s modulom Siemens XT55 vytvorená ako diplomová práca pána Miroslava Černého [2]. Použitím tejto dosky získame možnosť prijímania dát o polohe zo systému GPS. Taktiež nám umožňuje prenos získaných dát použitím GSM/GPRS (vid kapitola 5). Použité periférie sú jedno asynchrónne sériové rozhranie GPS na prenos dát o polohe do riadiacej jednotky a jedno asynchrónne sériové rozhranie GSM na obojsmerný prenos dát a príkazov do riadiacej jednotky.

Týmto nám vznikla jednotka ktorá splňuje základné hardwarové požiadavky kladené na systém. Ďalej musíme vyriešiť komunikáciu medzi jednotlivými časťami systému.

Na strane dispečerského centra sa používajú štandartné protokoly ako je ethernet na fyzickej resp. linkovej vrstve a TCP/IP na sieťovej a transportnej vrstve. Obsluhu týchto protokolov zaisťuje Operačný systém a preto sa jej bližšie nebudem venovať. Na strane automobilu sa jedná o komunikácie:

- **automobil - riadiaca jednotka:** Interná komunikácia automobilu používa komunikačnú zbernicu CAN. Pomocou riadiacej jednotky sme schopný sa pripojiť na túto zbernicu a prijímať s nej dáta. S pohľadu modelu ISO/OSI vid obr. 2.2 je CAN definovaný na fyzickej a linkovej vrstve. Použil som už vytvorené knihovny `can.h`, `msCan.h` a `com.h`, ktoré umožnujú príjem a vysielanie CAN správ. Tieto knihovne boli vytvorené v rámci diplomovej práce pána Michaela Krakoru [3]

- **riadiaca jednotka - prenosová jednotka:** Tieto dve časti systému navzájom komunikujú prostredníctvom asynchrónnych sériových rozhraní. S pohľadu ISO/OSI



Obr. 2.2: Model komunikácie ISO/OSI.

obr. 2.2 sa jedná o fyzickú vrstvu RS 232. Na základe programového vybavenia vytvoreného pánom Ondrejom Špinkom som vytvoril knižnicu `uart.h`. Ovladače som musel doplniť o cyklické bufferi na strane príjmu z dôvodu časovej náročnosti niektorých úloh a tým spôsobenia možnej straty prijatého znaku v prípade výskytu prerušenia.

- **riadiaca jednotka - internet:** Táto komunikácia sa používa pre prenos získaných dát do dispečerského centra. Jej súčasťou je komunikácia medzi riadiacou a prenosovou jednotkou a medzi prenosovou jednotkou a prístupovým bodom operátora, ktorý nám poskytuje službu GPRS (Internet Service Provider - ISP). O druhú komunikáciu sa stará priamo prenosová jednotka. O prvú sa musíme postarať mi. Jak som už zmienil na fyzickej vrstve modelu ISO/OSI sa jedná o RS 232. S dôvodu vytvorenia trvalého spojenia je nutné vyriešiť aj komunikáciu na vyšších vrstvách ISO/OSI obr. 2.2. Vytvorenie nástrojov potrebných pre túto komunikáciu je práve stežejnou úlohou tejto diplomovej práce. Na vytvorenie spojenia s ISP sa používa protokol linkovej vrstvy Point-To-Point (PPP), ktorý umožňuje vyjednanie spojenia s požadovanými vlastnosťami. Popisu PPP sa venuje kapitola 3 a jeho implementácii kapitola 6. Na prenos dát v rámci internetu sú ďalej potrebné protokoly sieťovej a transportnej vrstvy kde sa jedná o Internet Protocol (IP) a User Datagram Protocol (UDP). Tieto protokoly sú popísane v kapitole 4 a ich implementácia v kapitole 6.

Kapitola 3

Point To Point protokol

3.1 Úvod

Mnoho aplikácii vyžaduje všestrannejšie sériové rozhrania než je veľmi jednoduchý sériový protokol SLIP. V takom prípade sa používa PPP (Point-to-Point Protocol) . Tento protokol umožňuje operácie pracujúce full-duplex za predpokladu že pakety budú doručené v správnom poradí, tzn. rovnako ako boli vyslané. Používa sa špeciálne na vytvorenie telefónneho spojenia k ISP (Internet Service Provider), pretože obsahuje systém na určenie identity uživatela a získanie klientovej IP adresy. PPP obsahuje niekoľko ďalších protokolov s ktorých každý dokáže vyjednávať nezávisle. Preto na vytvorenie PPP spojenia je potrebne prejsť určitými častami vyjednávania pri ktorých sa používajú protokoly LCP (Link Control Protocol), rodina protokolov NCP (Network Control Protocol), medzi ktoré patrí napr. IPCP (Internet Protocol Control Protocol) a často aj authentifikačnej časti napríklad PAP (Password Authentication Protocol). V prípade vytvárania spojenia neexistuje žiadna skratka v tejto postupnosti. Ak jedna z týchto troch častí neprebehne úspešne, PPP spojenie sa uzatvorí a žiadne dáta nebudú môcť byť prenášaná.

3.2 Konštrukcia PPP

PPP je zameraný na vytvorenie jednoduchého spojenia medzi dvoma stanicami. Býva používaný na RS232 linke pri vytvárani spojenia k ISP, ale býva používaný aj pri iných situáciách, kde je potrebné spojenie medzi dvoma stanicami napr. PPP v Ethernete poskytuje PPP rúru medzi dvoma špecifickými stanicami Ethernetovej siete.

PPP poskytuje:

- **Rámcovanie** (framing). Rozdelenie prúdu dát po sériovej linke do blokov kde každý obsahuje začiatočnú a koncovú značku.
- **Kontrola chýb**. Na zabezpečenie kontroly chýb slúži CRC (Cyclic Redundancy Code).
- **Escape sekvencia** (Escape sequences). Bežné kontrolné znaky, ktoré sa môžu vyskytovať vo forme dát, môžu byt eliminované tým že ich zahrnieme do predpísanej escape sekvencie.
- **Vyjednávanie**. Dve stanice môžu vyjednávaním dosiahnuť vzájomne prípustné protokoly a možnosti ktoré obom vyhovujú. Toto často zahŕňa autentifikáciu užívatela a vyjednanie adresy napríklad dynamickej IP adresy z určitého rozsahu.

3.2.1 Rámcovanie

Rámcovacia schéma pochádza z ISO štandardu komunikačného protokolu HDLC (High-level Data Link Control), ktorý bol vytvorený spoločnosťou IBM. Všeobecná štruktúra HDLC paketu a daný PPP paket vypadajú následovne tabuľka 3.1.

HDLC	**Flag**	**Adresa**	**Kontrolný znak**	**Informácia**	**FCS**	**Flag**
	1 byte	1 byte	1 byte	0-1502 bytov	2 byty	1 byte
PPP	`7Eh`	`FFh`	`03h`	`DATA`	`CRC`	`7Eh`

Tabuľka 3.1: Štruktúra HDLC/PPP paketu

HDLC *flag* začína a ukončuje každý paket. Pretože PPP môžeme zaradiť s pohľadu ISO/OSI modelu obr. 2.2 na linkovú vrstvu potom polia *adresa* a *kontrolný znak* považujeme za linkovú hlavičku (link header) a (*FCS* (Frame Check Sequence) linkový príves (link trailer). HDLC adresa vyjadruje adresu stanice ktorej je paket určený. PPP používa hodnotu FF (broadcast). Riadiaci znak slúži na rozlíšenie typov HDLC paketov podľa najnižších 2 bitov. Sú 3 možnosti: Informačné pakety (najnižší bit 0), nečíslované pakety (najnižšie 2 bity sú 11) a pakety supervizoru (najnižšie 2 bity sú 10). U PPP sú to vždy nečíslované pakety teda hodnota `03h`.

Môžeme predpokladať, že PPP paket bude vždy začínať sekvenciou `7F FF 03h`. Je to vhodné pri analyzovaní komunikácie medzi dvoma stanicami a jednoduchej detekcii začiatku paketu. Existujú ale konfiguračné možnosti, ktoré sa dajú vyjednať (vid dodatok A), pri ktorých je adresné a riadiace pole odstránené.

Pole *Informácia* obsahuje prenášané dáta. Hlavička HDLC paketu neposkytuje možnosť špecifikácie protokolu vyššej vrstvy, tj. neumožňuje miešať pakety protokolov napr. IP a IPX. Voľba protokolu sa určuje v počiatočnom inicializačnom dialógu.

Toto obmedzenie platí pre číslované pakety. U nečíslovaných (čo je prípad PPP) je možne dát na počiatok dátového pola špecifikáciu protokolu.

Pole (*FCS* slúži na kontrolu chýb.

3.2.2 Kontrola chýb

Pole *FCS* v HDLC rámci umožňuje strane príjemcu skontrolovať integritu prichádzajúceho paketu. Bežne sa používa 16 bitová CRC ale môžu byť vyjednané aj iné možnosti. Z prenášaných dát, adresného a riadiaceho pola sa vypočíta kontrolní súčet, ktorý je prenesený spolu z daným paketom, s ktorého bol vytvorený. Príjemca vypočíta kontrolní súčet s celého paketu teda aj s poslednými dvoma bytmi ktoré tvoria CRC. V prípade výsledku `F0B8h` je paket prijatý správne. Ak sa súčet nerovná `F0B8h`, paket je zahodený.

3.2.3 Escape sekvencia

Bolo uvedené, že PPP paket sa začína aj končí znakom ASCII hodnoty `7Eh`. Problém nastáva ak sa tento znak vyskytne ako normálny byte v dátach ktoré chceme prenášať. Potom by mohlo dôjsť k zámene jeho významu za ukončovací znak. Na ošetrenie tejto aj ďalších možností slúži Escape sekvencia. Skladá sa zo znaku ASCII hodnoty `7Dh` nasledovaným originálnou hodnotou na ktorej je prevedený exclusive OR s hodnotou `20h`. Tým sa napr. `7Eh` sa zmenilo na `7D 5Eh`. Taktiež ak by sa v dátach vyskytoval znak s hodnotou `7Dh`, potom sa tento nahradí sekvenciou `7D 5Dh`.

Escape sekvencia taktiež slúži ako ochrana na pomýlenie si riadiaceho znaku, ktorý má špeciálny význam v sériovej komunikácii. Špeciálne znaky XOFF a XON sa používajú v nízko úrovňových ovládačoch na pozastavenie a pokračovanie v komunikácii. Preto aj tieto hodnoty sa vysielajú v podobe escape sekvencie teda napr. XOFF hodnoty `14h` sa vyšle ako `7D 34h`. Podľa základného nastavenia bývajú všetky znaky v rozsahu hodnôt `00h` -

`1Fh` vysielane v podobe escape sekvencie. Stanice môžu vyjednať pri ďalšej komunikácii že sa nebude používať escape sekvencia nikde alebo len na špecifických znakoch ktoré definuje konfiguračna možnosť ACCM, vid dodatok A .

Dôležite je si uvedomiť že escape sekvencia sa používa nielen na dáta ale aj na kontrolné a FCS pole. Teda začiatok PPP paketu vypadá na začiatku vyjednávania väčšinou ako `7E FF 7D 03h`. V prípade FCS ktorá by napríklad vypadala následovne `01 7D` sa musí zmeniť na `7D 21 7D 5D`.

3.2.4 Vyjednávanie

Vyjednávanie je najdôležitejšia súčasť PPP. Pri vytvorení point-to-point linky sú nastavené štandartné parametre. Ako prvý krok pri vytváraní spojenia nastane vyjednávanie na nových parametroch ktoré vyhovujú jednej alebo druhej stanici ako napríklad kompresia dát, adresovanie a autentifikácia. Vyjednávanie má nasledujúce charakteristiky.

- **Symetria** Dva kompletné súbory požadovaných parametrov sú prijate z oboch strán
- **Možnosti** Veci ktoré sa vyjednávajú sa nazývajú konfiguračné možnosti a môžu byť
  1. Hodnota - napr. maximálna veľkosť paketu
  2. Logická premenná - napr. povolenie/zakázanie kompresie
  3. Protocol - napr. ktorý protokol sa bude používať pri autentifikácii
- **Požiadavka/Odpoveď** Vyjednávanie radou konfiguračných požiadaviek (configure request) a k nim príslušných odpovedí.
- **ACK/NAK/REJ** Konfiguračná možnosť môže byt prijatá (ACK) alebo príjemca ohlasí chybu (NAK/REJ) a uvedie dôvod. Viac v kapitole LCP 3.3.2.1
- **Začatie/Ukončenie** Prenos konfiguračných požiadaviek naznačuje, že vysielajúca stanica chce inicializovať PPP spojenie. Potvrdenie všetkých požadovaných konfiguračných možností pri všetkých protokoloch v oboch smeroch naznačuje, že PPP spojenie bolo vytvorené. Toto platí výlučne do tedy pokým nie je vyslaný požiadavok ukončenia spojenia (terminate-request) a prijatý kladná odpoveď (terminate-acknowledge).

- **Protokoly** PPP nie je iba jeden protokol ale obsahuje súbor protokolov ktoré slúžia na vyjednávanie základných parametrov linky.

    1. Linkový riadiaci protokol (LCP - Link Control Protocol) - definuje základne komunikačné parametre
    2. Rodina autentifikačných protokolov (PAP - Password Authentication Protocol, CHAP - Challenge Handshake Authentication Protocol) - zabezpečuje bezpečnosť
    3. Rodina sieťových protokolov (NCP - Network Control Protocol) - vyjednáva všetko potrebné pre daný typ sietového protokolu. Napríklad IP riadiaci protokol (IPCP - IP Control Protocol) vyjednáva IP, DNS adresu u IP.

- **Sieťová vrstva** - Niektoré sieťové protokoly sa ďalej delia ako napr. IPCP na riadiaci protokol slúžiaci na vyjednávanie a sieťový protokol na prenos dát.

- **Stavový automat** - Proces vyjednávania je kontrolovaný štandarizovaným stavovým automatom, ktorý zabezpečuje správnu činnosť v ľubovolnej situácii. Jeden stupeň je potrebný pre každú zložku protokolu.

- **Zbližovanie** - Vyjednávací proces by mal dospieť k zblíženiu keď dve stanice dosiahnu zhodu vo vyjednaných konfiguračných možnostiach, ktoré budú prijateľné pre obe. Ak je jedna špecifická konfiguračná možnosť podstatná pre jednu stranu ale druhá ju nemôže prijať nedôjde k zblíženiu.

- **Spoľahlivosť** - Prenos dát sa považuje za nespoľahlivý a preto ak neprijmeme odpoveď musí byt prenos zopakovaný.

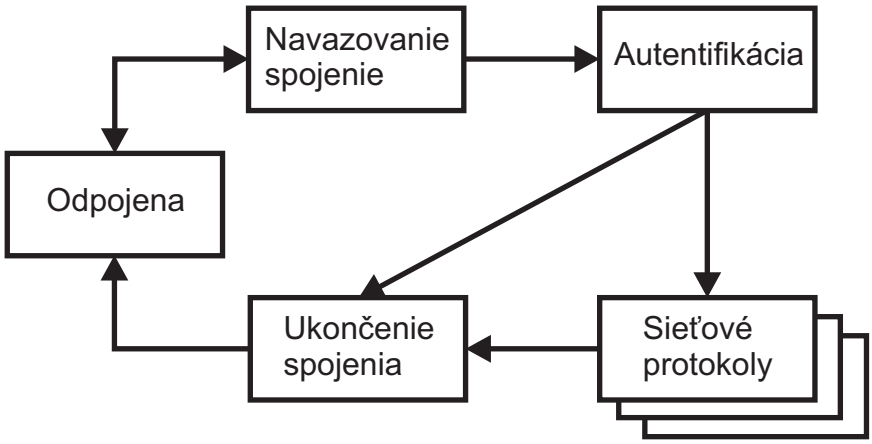
- **Spojenie** - Ak sa stanice dohodnú na spoločne použiteľnom súbore protokolov a konfiguračných možnostiach, potom PPP spojenie bude vytvorené.

- **Dáta** - Ako jedna z častí vyjednávania je dohoda o protokole na prenos dát na sieťovej vrstve (napr. IPCP umožní prenášať IP pakety po úspešnom vyjednaní spojenia). Protokoly sieťovej vrstvy zabaľujú dáta čo umožňuje PPP spojeniu prenos rôznych sieťových protokolov.

3.3 Vytvorenie PPP spojenia

3.3.1 Fáze spojenia

Spojenie PPP sa dá rozdeliť do piatich hlavných fáz. Jednotlivé fáze a ich postupnosť zobrazuje obr. 3.1.



Obr. 3.1: Jednotlivé fáze linky v protokolu PPP.

3.3.1.1 Linka odpojená (Link Dead)

Spojenie vždy začína a končí v tejto fáze. Keď dôjde k nejakej externej udalosti (napr. modemy stratia medzi sebou spojenie alebo sieťový administrátor vydá príkaz k ukončeniu spojenia) prechádza linka do tejto fáze.

3.3.1.2 Vyjednávanie spojenia (Link Establishment)

Vyjednanie spojenia sa prevádza výmenou konfiguračných paketov. Táto výmena je ukončená ak obe stanice prijali potvrdzovací konfiguračný paket. V priebehu vyjednávania spojenia sa žiadne dátové pakety neprenášajú. V prípade výskytu je takýto paket zahodený.

3.3.1.3 Autentifikacia (Authentication)

V tejto fázy klient preukazuje svoju totožnosť, ktorá musí byt nutne overená, aby mohlo dôjst k prenosu dát sieťových protokolov. Pod pojmom klient myslime tú stanicu, ktorá bola k tomu vyzvaná (štandartne autentifikácia nie je nastavená a musí byt vyjednaná v predošlej fázi). Po preukázaní totožnosti jednej stanice si môžu stanice svoju úlohu

vymeniť. V praxi sa ale väčšinou preukazuje len jedna stanica a teda v našom prípade (GSM sieť) to je jednotka umiestnená v automobile.

3.3.1.4 Protokol sieťových vrstiev (Network-Layer Protocol)

Táto fáza v sebe zahrnuje mnoho protokolov. Ak PPP spojenie úspešne ukončilo predchádzajúce fáze, musí každý sieťový protokol(IP, IPX, AppleTalk), ktorý chce prenášať dáta byť oddelene nakonfigurovaný príslušným sieťovým kontrolným protokolom. Dátové pakety sieťového protokolu, ktorý nie je nakonfigurovaný sa zahodia. Zároveň môže byť nakonfigurovaných aj viac sieťových protokolov.

3.3.1.5 Ukončenie spojenia (Link Termination)

PPP môže ukončiť spojenie v ľubovolný okamžik v dôsledku stratenia spojenia, chyby autentifikácie, zhoršenie kvality spojenia, pretečeniu kontrolného čítača alebo s administratívnych dôvodov. Behom tejto fáze sa všetky ostatné pakety, ktoré niesu protokolu LCP zahadzujú. LCP používa na ukončenie spojenia ukončovacie pakety (Terminate packets). Po výmene ukončovacích paketov nastáva príkaz fyzickej vrstve o odpojení a tá môže reagovať napr. zavesením komutovanej linky.

3.3.2 Protokoly

Jak som sa zmienil, PPP obsahuje množinu protokolov, ktoré slúžia na vyjednávanie a prenos dát. Pri zostavovaní spojenia sa jednotlivé protokoly používajú v nasledujúcom poradí.

3.3.2.1 Linkový riadiaci protokol - LCP

Protokol LCP sa používa ešte pred tým, než sa vôbec uvažuje, aký protokol sa bude používať na sieťovej vrstve. Je to spoločný protokol pre všetky sieťové protokoly, tzn. v každom prípade začína komunikácia pri PPP práve ním. LCP je určeny na naviazanie spojenia, ukončenie spojenia, výmenu autentifikačných informácii a pod. LCP protokol je zabalený v základnom PPP paketu v poli *information* teda v úseku kde sa s pohľadu PPP prenášajú dáta. Formát paketu LCP vypadá následovne.

Protokol	**Kód**	**Identifikátor**	**Dĺžka paketu**	**Dáta**
2 bytes	1 byte	1 byte	2 byty	0-1496 bytov

Tabuľka 3.2: Štruktúra LCP paketu

Identifikátor protokolu LCP je `C021h` a je umiestnený v poli *Protokol*. Pole *kód* dĺžky 1 byte špecifikuje typ príkazu resp. odpovedi protokolu LCP vid tabuľka 3.3.

Pole *Identifikátor* veľkosti 1 byte slúži na presnú identifikáciu daného požiadavku. Pri vyslaní paketu zvolí odosielateľ identifikáciu (teda vyberie číslo) a odpoveď na tento požiadavok je vyslaná späť z rovnakým identifikátorom. Pomocou tohto pola dokážeme určiť príslušnosť odpovedi k danému požiadavku.

Pole *Dĺžka paketu* veľkosti 2 byty obsahuje číslo udávajúce súčet dĺžok polí: Kód, Identifikátor, dĺžka paketu a dáta.

Pole *dáta* obsahuje požadované konfiguračné voľby (alebo prípadne odpovedí), ktoré sa líšia od implicitných parametrov linky. Toto pole sa skladá z jednej alebo viacerých konfiguračných volieb. Jednotlivé voľby sú ukladané sekvenčne za sebou jak je znázornené v tabuľke 3.4.

Typ voľby	**Dĺžka**	**Dáta**	**Typ voľby**	**Dĺžka**	**Dáta**	
1 byte	1 byte	1-n bytov	1 byte	1 byte	1-n bytov	...

Tabuľka 3.4: Štruktúra konfiguračných volieb u LCP

Typ voľby je veľkosti 1 byte a identifikuje o akú konfiguračnú metódu sa jedná. Jednotlivé voľby konfigurácie a im odpovedajúce hodnoty sú vypísane v tabuľke 3.5. Ich popis sa nachádza v dodatku. Pre podrobnejšie informácie o konfiguračných možnostiach doporučujem [4]. Dalšie špecifikacie možno nájst v [5] a [6].

Kód	Názov	Význam
1	Configure-Request	Konfiguračný paket, ktorý nesie požiadavky na zmenu implicitného parametru linky.
2	Configure-Ack	Konfiguračný paket s kladným potvrdením požiadavkou na zmenu implicitných parametrov linky. Tj. všetky požadované zmeny parametrov sú akceptované.
3	Configure-Nak	Konfiguračný paket s odpoveďou. Druhá strana neakceptuje všetky požiadavky na zmenu parametrov linky. Tie ktoré neakceptuje sú v tomto pakete špecifikované. Ostatné požiadavky sú akceptované (tj. nešpecifikované požiadavky v paketu Configure-Nak sú akceptované).
4	Configure-Reject	Konfiguračný paket odmietajúce všetky požiadavky ktoré obsahuje. To môže nastať napríklad v dôsledku chybného kódu požiadavku alebo prijatia požiadavku, ktorý nieje implementovaný.
5	Terminate-Request	Požiadavok na ukončenie spojenia
6	Terminate-Ack	Potvrdenie požiadavku na ukončenie spojenia
7	Code-Reject	Odmietnutie požiadavku z dôvodu neznámeho kódu. Môže byt spôsobené tým že druhá stanice využíva inú verziu protokolu.
8	Protokol-Reject	Druhá stanica nepodporuje uvedený protokol.
9	Echo-Request	Podpora testovania smyčky na linkovej úrovni.
10	Echo-Reply	Povinná odpoveď na Echo-Request.
11	Disckard-Request	Zahodenie paketu. Používa sa pri testovaní linky pri záťaži, tj. odosielateľ generuje pomocou týchto paketov umelú záťaž linky.

Tabuľka 3.3: Typy LCP paketov

Typ voľby	Názov
0	Rezervovaný (RESERVED)
1	Maximum prijatých zložiek (Maximum-Receive-Unit)
2	Asynchrónna kontrolná mapa (Async-Control-Character-Map)
3	Autentifikáčný protokol (Authentication-Protocol)
4	Protokol kvality (Quality-Protocol)
5	Magické číslo (Magic-Number)
6	Rezervovaný (RESERVED)
7	Kompresie dátového pola PPP paketu (Protocol-Field-Compression)
8	Kompresia adresového a kontrolného pola (Address-and-Control-Field-Compression)

Tabuľka 3.5: Typy konfiguračných volieb u LCP

Dĺžka je pole veľkosti 1 byte a určuje dĺžku konfiguračnej možnosti.

Dáta sú veľkosti 0 alebo viac bytov a ich formát je určený obsahom pola `typ voľby`.

3.3.2.2 Rodina autentifikačných protokolov

Preukazovať totožnosť je možne v protokolu PPP troma spôsobmi.

- **Terminálový dialóg** - Terminálový dialog nesúvisí s protokolom PPP ale s jeho implementáciou. Väčšinou sa totiž uživatel prihlasuje po sériovej linke k serveru. Na serveru odpočúva túto linku terminálový proces vyžadujúci meno uživatela a heslo. Až rozpozná že sa nejedná o bežného uživatela terminálu ale uživatela pre ktorého má na linke štartovať protokol PPP, potom je možno autentifikačnú fázu protokolu PPP preskočiť.
- **Password Authentication Protocol (PAP)** - Tento protokol je obdobou autentifikácie pomocou terminálového dialógu. Tj. uživatel preukazuje svoju totožnosť tiež pomocou mena uživatela a hesla. Pre výmenu autentifikačných informácii sa ale používa protokol LCP, tj. meno uživatela a heslo sa nevkladá priamo na linku ale balí sa do protokolu LCP.

- **Challange Handshake Authentication Protocol (CHAP)** - Je považovaný za dokonalejší ako predchádzajúce dva. Oba konce zdieľajú rovnaké tajomstvo (v podstate je to šifrovací kľúč symetrickej šifry). Stanica, ktorá autentifikáciu inicializuje, vygeneruje náhodný reťazec ako dotaz (challenge), ktorý odošle druhej strane. Druhá strana tento reťazec zašifruje pomocou zdieľaného tajomstva a odošle späť. Stanica, ktorá autentifikáciu inicializovala, tak obdržala zašifrovaný reťazec, ktorý dešifruje. Porovná oba reťazce. Ak sú rovnaké, potom druhej strane potvrdí úspešný výsledok autentifikácie. V opačnom prípade odpovie že autentifikácia prebehla neúspešne a môže začať opäť z navazovaním spojenia.

Výhodou protokolu CHAP je že oba konce majú zdieľané tajomstvo - v dôsledku čoho je ľahké previesť obojstrannú identifikáciu. Zdieľané tajomstvo je súčasne nevýhodou protokolu CHAP, pretože nemožno zabrániť zneužitiu tohto tajomstva druhou stranou (na rozdiel od autentifikácie heslom kde má druhá strana prístup iba k zašifrovanému heslu). Protokol CHAP bližšie špecifikuje [7].

Ďalší problém autentifikácie spočíva v tom, že klient sa bude chcieť prihlasovať nie stále na jeden prístupový server, ale na rôzne prístupové servery. Klasickým prípadom je pripojenie k poskytovateľovi internetu (ISP), ktorý má svoje prístupové body v rôznych mestách. V takomto prípade by museli byt autentifikačné údaje uložené na každom prístupovom serveru čo nie je žiaduce. Riešenie spočíva v centralizácii autentifikačných informácii. V sieti je jeden (alebo viacej záložných) serverov ktoré udržujú autentifikačné informácie o každom uživatelovi. Okrem autentifikačných informácii môžu byť uložené aj konfiguračné informácie (napr. IP adresa uživatela, prístupové filtre). Prístupový server potom voči takémuto serveru vystupuje ako klient, ktorý požaduje službu: overenie autentifikačných odpovedí alebo poskytnutie IP adresy, ktorú ma protokolom IPCP predať uživatelovy atd. Ako protokol medzi prístupovým serverom a serverom s autentifikačnými a konfiguračnými informáciami sa dnes často používa protokol RADIUS alebo protokol TACAC+.

3.3.2.3 IP Riadiaci Protokol (IPCP)

Sieťových protokolov je veľké množstvo. Pre nás je najdôležitejší internet protokol (IP). Pred tým než pomocou neho môžeme prenášať dáta ho musíme vyjednať. Na to slúži IPCP protokol ktorý sa radí medzi sieťové kontrolné protokoly (NCP).

Formát paketu je znázornený v tabuľke 3.6.

Protokol	**Kód**	**Identifikátor**	**Dĺžka paketu**	**Dáta**
2 bytes	1 Byte	1 Byte	2 bytes	0-1496 bytes

Tabuľka 3.6: Formát IPCP paketu

Pole *protokol* obsahuje identifikátor IPCP protokolu čo je číslo `8021h`. Pole *kód* veľkosti 1 byte špecifikuje typ príkazu resp. odpovedi protokolu IPCP vid tabuľka 3.7.

Kód	**Názov**	**Význam**
1	Configure-Request	Konfiguračný paket, ktorý nesie požiadavky na zmenu implicitného parametru linky.
2	Configure-Ack	Konfiguračný paket s kladným potvrdením požiadavkou na zmenu implicitných parametrov linky. Tj. všetky požadované zmeny parametrov sú akceptované.
3	Configure-Nak	Konfiguračný paket s odpoveďou. Protejšia strana neakceptuje všetky požiadavky na zmenu parametrov linky. Tie ktoré neakceptuje sú v tomto pakete špecifikované. Ostatné požiadavky sú akceptované (tj. nešpecifikované požiadavky v paketu Configure-Nak sú akceptované).
4	Configure-Reject	Konfiguračný paket odmietajúce všetky požiadavky. To sa môže napr. stať v dôsledku chybného kódu požiadavku.
5	Terminate-Request	Požiadavok na ukončenie spojenia
6	Terminate-Ack	Potvrdenie požiadavku na ukončenie spojenia
7	Code-Reject	Odmietnutie požiadavku z dôvodu neznámeho kódu. Môže byt spôsobené tým, že druhá stanice využíva inú verziu protokolu.

Tabuľka 3.7: Typy IPCP paketu

Pole *identifikátor* veľkosti 1 byte slúži na presnú identifikáciu daného požiadavku. Pri vyslaní paketu zvolí odosielateľ identifikáciu (teda vyberie číslo) a odpoveď na tento

požiadavok je vyslaná späť zo rovnakým identifikátorom. Pomocou tohto pola dokážeme určiť príslušnosť odpovedi k danému požiadavku.

Pole *Dĺžka* paketu veľkosti 2 byty obsahuje číslo udávajúce súčet veľkostí polí: Kód, Identifikátor, Dĺžka paketu a Dáta.

Pole *Dáta* obsahuje požadované konfiguračné voľby (alebo prípadne odpovedí), ktoré sa líšia od implicitných parametrov linky. Toto pole sa skladá z jednej alebo viacerých konfiguračných volieb. Jednotlivé voľby sú ukladané sekvenčne za sebou jak je znázornené.

Formát je rovnaký ako v prípade LCP tabulka 3.4

Voľba je veľkosti 1 byte a identifikuje o akú konfiguračnú metódu sa jedná. Jednotlivé voľby konfigurácie a im odpovedajúce hodnoty tabulka 3.8

Typ voľby	**Názov**
2	Protokol kompresie IP (IP-Compresion-Protocol)
3	IP adresa (IP-adress)
129	Primárna DNS adresa (Primary-DNS-adress)
131	Sekundárna DNS adresa (Secondary-DNS-Adress)

Tabuľka 3.8: Typy volieb pre IPCP

Protokol kompresie IP (IP-Compresion-Protocol) Kompresia TCP/IP záhlavia

Typ voľby	Dĺžka	Dáta

Typ voľby = 2
Dĺžka = 6
Dáta Pole je veľkosti štyri byty kde prvé dva sú hodnoty 002Dh a ďalšie dva obsahujú parametre kompresie.

IP adresa (IP-adress) Predanie IP adresy druhej strane. Takto je možné dynamicky prideľovať IP adresy. Ak chce druhá strana používať inú IP adresu potom odpovie paketom Configure-Nak, kde túto adresu špecifikuje

Typ voľby	Dĺžka	Dáta

Typ voľby = 3
Dĺžka = 6
Dáta Pole je veľkosti 4 byty a obsahuje IP adresu.

Primárna DNS adresa (Primary-DNS-adress) Špecifikuje primárny menový server (DNS) bližšie info vid [8]

Typ voľby	Dĺžka	Dáta

Typ voľby = 129
Dĺžka = 6
Dáta Pole je veľkosti 4 byty a obsahuje IP adresu DNS serveru.

Sekundárna DNS adresa (Secondary-DNS-Adress) Špecifikuje sekundárny menový server (DNS) bližšie info vid [8]

Typ (Type)	Dĺžka	Dáta

Typ voľby = 131
Dĺžka = 6
Dáta Pole je veľkosti 4 byty a obsahuje IP adresu DNS serveru.

V rámci protokolu PPP sa pre prenos dátových IP paketov verzie 4 používa identifikácia `0021h`. V tomto prípade je situácia komplikovanejšia, pretože nie všetky pakety majú komprimované záhlavie (nie všetku IP pakety nesú protokol TCP, napr. ICMP pakety sa nekomprimujú), Je teda nutné rozlišovať v prenášaných paketoch pakety komprimované TCP/IP záhlavím a pakety s nekomprimovaným záhlavím. Preto v záhlaví PPP paketu v poli protokol majú nekoprimované pakety identifikáciu 0021h a pakety s komprimovaným IP záhlavím identifikáciu `002Dh`.

3.3.3 Stavový automat

Štandartný stavový automat je používaný na smerovanie vyjednávania pri LCP a IPCP. V norme RFC [4] je vyjadrený ako dvojdimenzionálna tabuľka kde riadky reprezentujú udalosti (events) a stĺpce reprezentujú stavy (states). V skutočnej realizácii (PPP stacku) je táto tabulka prepísaná ako dvojrozmerné pole v jazyku C, ktorý umožňuje okamžité vyhľadávanie správnej akcie (action) na základe stavu, v ktorom sa nachádzame a udalosti, ktorá nastala. Stavy s ktorých sa stavový automat skladá sú uvedené v tabuľke 3.9. Udalosti, ktoré môžu nastať sú v tabuľke 3.10 a akcie v tabuľke 3.11. Samotný stavový automat je s dôvodou umiestnenia rozdelený na dve tabuľky, keď prvá 3.12 zahŕňa stavy 0-4 a druhá 3.13 stavy 5-9.

Stav (State)	**Význam**
Inicializačný (Initial)	Uzatvorený stav na počiatku
Štartovací (Starting)	Otvorený stav pred tým ako sa nižšia vrstva (myslené z pohľadu ISO/OSI) dostane do aktívneho stavu
Uzatvorený (Closed)	Stav ktorý nastane po uzatvorení linky, očakáva sa že spojenie sa ukončí
Zastavený (Stopped)	Očakáva ukončenie
Uzatváranie (Closing)	Aktívne uzatvorenie: pokus o ukončenie spojenia
Zastavenie (Stopping)	Pasívne uzatvorenie: druhá stanica začala s ukončovaním spojenia
Vyslaný požiadavok (Request-sent)	Konfiguračný požiadavok bol vyslaný
Potvrdenie prijate (ACK-received)	Požiadavok našej strany bol potvrdený ale vzdialený požiadavok (teda požiadavok vyslaný druhov stranou) ešte nebol akceptovaný
Potvrdenie vyslané (ACK-send)	Vzdialený požiadavok bol potvrdený ale požiadavok našej strany ešte nebol akceptovaný
Otvorený (Opened)	Spojenie vytvorené

Tabuľka 3.9: Stavy v stavom automate

Udalosťi (Events)	**Význam**
Up	Nižšia vrstva je pripravená
Down	Nižšia vrstva už nie je pripravená
Open	Požiadavok na vytvorenie spojenia
Close	Požiadavok na uzatvorenie spojenia
TO+, TO-	Vypršal čas reštartovacieho čítača (čítač je ¿0 alebo =0)
RCR+, RCR-	Prijatý konfiguračný požiadavok (*Configure-Request*) akceptovatelný alebo neakceptovatelný
RCA	Prijaté konfiguračné potvrdenie (*Configure-Ack*)
RCN	Prijaté konfiguračné odmietnutie (*Configure-Nak*)
RTR	Prijatý požiadavok ukončenia spojenia (*Terminate-Request*)
RTA	Prijaté potvrdenie na ukončenie spojenia (*Terminate-Ack*)
RUC	Prijatý neznámy kód
RXJ+, RXJ-	Prijaté odmietnutie kódu alebo protokolu
RXR	Prijaté echo (*Echo-Request*, *Echo-Reply* alebo *Discart-Request*)

Tabuľka 3.10: Udalosti v stavovom automе

Akcie (Actions)	**Význam**
XXX	Nelegálna udalosť, nikdy by nemala nastať
TLU	Signál vyššej vrstve že táto vrstva je pripravená (Up)
TLD	Signál vyššej vrstve že táto vrstva už nie je pripravená
TLS	Signál nižšej vrstve že táto vrstva začala činnosť
TLF	Signál nižšej vrstve že táto vrstva ukončila činnosť
IRC	Inicializuj čítač slúžiaci na reštart na najvyššiu hodnotu
ZRC	Vynuluj čítač slúžiaci na reštart
SCR	Vyšli konfiguračný požiadavok (*Configure-Request*)
SCA	Vyšli konfiguračné potvrdenie (*Configure-Ack*)
SCN	Vyšli konfiguračné odmietnutie alebo zamietnutie (*Configure-Nak* or *Configure-Reject*)
STR	Vysli ukončovací požiadavok (*Terminate-Request*)
STA	Vyšli ukončovacie potvrdenie (*Terminate-Ack*)
SCJ	Vyšli odmietnutie kódu
SER	Vyšli echo (*Echo-Reply*)

Tabuľka 3.11: Akcie v stavovom automе

Každé pole v stavovom poli má jednu alebo viacej akcii a prípadne zmenu, stavu ktorý je udávaní číselnou hodnotou. Napríklad pole v diagrame IRC+SCR+6 znamená inicializuj reštartový čítač, vyšli konfiguračný požiadavok a prejdi do stavu 6 (Request-Send). Stavový automat teda vypadá následovne.

(0)Initial	(1)Starting	(2)Closed	(3)Stopped	(4)Closing	
2	IRC + SCR + 6	XXX	XXX	XXX	Up
XXX	XXX	0	TLS + 1	0	Down
TLS + 1	XXX	IRC + SCR + 6	3	5	Open
0	TLS + 0	2	3	4	Close
XXX	XXX	XXX	XXX	STR + 4	TO+
XXX	XXX	XXX	XXX	TLF + 2	TO-
XXX	XXX	STA + 2	IRC + SCR + SCA + 8	4	RCR+
XXX	XXX	STA + 2	IRC + SCR + SCN + 6	4	RCR-
XXX	XXX	STA + 2	STA + 3	4	RCA
XXX	XXX	STA + 2	STA + 3	4	RCN
XXX	XXX	STA + 2	STA + 3	STA + 4	RTR
XXX	XXX	2	3	TLF + 2	RTA
XXX	XXX	SCJ + 2	SCJ + 3	SCJ + 4	RUC
XXX	XXX	2	3	4	RXJ+
XXX	XXX	TLF + 2	TLF + 3	TLF + 2	RXJ-

Tabuľka 3.12: Stavový automat (stavy 0 - 4)

(5)Stopping	(6)ReqSend	(7)AckRecv	(8)AckSend	(9)Opened	
XXX	XXX	XXX	XXX	XXX	Up
1	1	1	1	TLD + 1	Down
5	6	7	8	9	Open
4	IRC + STR + 4	IRC + STR + 4	IRC + STR + 4	TLD + IRC + STR + 4	Close
STR + 5	SCR + 6	SCR + 6	SCR + 8	XXX	TO+
TLF + 3	TLF + 3	TLF + 3	TLF + 3	XXX	TO-
5	SCA + 8	SCA + TLU + 9	SCA + 8	TLD + SCR + SCA + 8	RCR+
5	SCN + 6	SCN + 7	SCN + 6	TLD + SCR + SCN + 8	RCR-
5	IRC + 7	SCR + 6	IRC + TLU + 9	TLD + SCR + 6	RCA
5	IRC + SCR + 6	SCR + 6	IRC + SCR + 8	TLD + SCR + 6	RCN
STA + 5	STA + 6	STA + 6	STA + 6	TLD + ZRC + STA + 5	RTR
TLF + 3	6	6	8	TLD + SCR + 6	RTA
SCJ + 5	SCJ + 6	SCJ + 7	SCJ + 8	SCJ + 9	RUC
5	6	6	8	9	RXJ+
TLF + 3	TLF + 3	TLF + 3	TLF + 3	TLD + IRC + STR + 5	RXJ-

Tabuľka 3.13: Stavový automat (stavy 5 - 9)

3.3.4 Príklad vyjednávania PPP spojenia

Pre jednoduchosť uvediem ukažkové pakety, ktoré sa vymieňajú medzi riadiacou jednotkou (DTE - Data Terminal Equipment) a modemom (DCE - Data Communications

Equipment). Modem je v našom prípade Siemens XT55 a riadiaca jednotka je buď PC alebo vývojová doska s Motorolou HC12.

Ako prvé pred samotnou výmenou paketov musíme zadať AT-príkazy, ktoré umožnia testovanie funkčnosti modemu a inicializáciu spojenia.

Zariadenie	Príkaz	Význam
DTE 1	`AT(CR)`	Testovanie pripojenia modemu
DCE 1	`(CRLF)OK(CRLF)`	Odozva modemu
DTE 2	`ATD*99***1#(CR)`	Vytočenie GPRS spojenia
DCE 2	`(CRLF)CONNECT(CRLF)`	Potvrdenie modemu že nastáva pripojenie

3.3.4.1 LCP

Keď sa modem spojí z druhou stanicou, začne sa vyjednávanie pomocou LCP. V stavovom automate sa nachádzame v stave *Initial*. Nás zámer vytvorenia spojenia realizujeme vytvorením udalosťi *Open*. Tým sa nastane akcia TLS a dostaneme sa do stavu *Starting*. V prípade že sme obdržali odpoveď CONNECT to znamená, že modem potvrdil vytvorenie spojenia vytvoríme udalosť *Up*, teda nastanu akcie IRC a SCR a dostávame sa do stavu *ReqSend*. Náš prvý request vypadá následovne

Význam	Paket
Flag a PPP hlavička	`7E FF 7D 23`
LCP Configure-Request, ID=1, Dlžka=23 a konfig. možnosti:	`C0 21 7D 21 7D 21 7D 20 7D 34`
Asynchrónna kontrolná mapa (ACCM)=0A00	`7D 22 7D 26 7D 20 7D 2A 7D 20 7D 20`
Magické číslo=31 2A 82 5A	`7D 25 7D 26 31 2A 82 5A`
Kompresie dátového pola (PFC)	`7D 27 7D 22`
CRC a Flag	`AD 50 7E`

Treba si uvedomiť že `7Dh` je escape sekvencia a znak nasledujúci po ňom je vytvorený pomocou XOR s `20h`. CRC v tomto prípade nesúhlasí pretože je to len príklad. Tento *Configure-Request*, ktorý vyšle naša strana teda riadiaca jednotka požaduje:

- Escape sekvencia sa bude používať iba na kontrolné znaky XON/XOFF
- Použi magické číslo ako kontrolu proti uzatvorenej smyčke v spojení
- Použi kompresiu na dátové pole PPP

Po vyslaní tohto paketu neprijmeme ihneď odpoveď na náš požiadavok, ale ISP vyšle svoj vlastný *Configure-Request*, ktorý môže vypadať nasledujúco.

Význam	**Paket**
Flag a PPP hlavička	`7E FF 7D 23`
LCP Configure-Request, ID=3, Dlžka=23 a konfig. možnosti:	`C0 21 7D 21 7D 23 7D 20 7D 39`
Asynchrónna kontrolná mapa (ACCM)=0A00	`7D 22 7D 26 7D 20 7D 2A 7D 20 7D 20`
Kompresie dátového pola (PFC)	`7D 27 7D 22`
Kompresia adresového a kontrolného pola (ACFC)	`7D 28 7D 22`
Magické číslo=86 61 CC 92	`7D 25 7D 26 86 61 CC 92`
Autentifikačný protokol (AP)=CHAP	`7D 23 7D 25 C2 23 7D 25`
CRC a Flag	`4D 50 7E`

ISP teda požaduje:

- Escape sekvencia sa bude používať iba na kontrolné znaky XON/XOFF
- Použi magické číslo ako kontrolu proti uzatvorenej smyčke v spojení
- Použi kompresiu na dátové pole PPP
- Použi kompresiu adresového a kontrolného pola (v mojom prípade je táto možnosť zakázaná mojou implementáciou kvôli schopnosti rozpoznávania začiatku paketov, ale o tom vid 6)
- Ako autentifikačný protokol sa bude používať CHAP

Tým, že sme prijali konf. request, ktorý vyžaduje kompresiu adresového a kontrolného pola, ktorú naša strana nepodporuje, sa nám vytvorila udalosť RCR-. Stojí za to si povšimnúť, že kompresia adresového a kontrolného pola môže platiť pre jednu stranu, v tomto prípade nás a pre druhú stranu môže byt zakázaná ISP. Teda keď sme v stave šesť a nastala udalosť RCR- tomu zodpovedá akcia SCN a zostávame v tom istom stave.

Po prijatí požiadavku zo strany ISP prijmeme okamžite odpoveď na náš požiadavok. V tomto prípade vyhovujú ISP všetky požiadavky a teda vyslal konfiguračne potvrdenie v ktorom schválil všetko čo sme žiadali:

Význam	**Paket**
Flag a PPP hlavička	`7E FF 7D 23`
LCP Configure-Ack, ID=1, Dĺžka=23 a konfig. možnosti:	`C0 21 7D 22 7D 21 7D 20 7D 34`
Asynchrónna kontrolná mapa (ACCM)=0A00	`7D 22 7D 26 7D 20 7D 2A 7D 20 7D 20`
Magické číslo=31 2A 82 5A	`7D 25 7D 26 31 2A 82 5A`
Kompresie dátového pola (PFC)	`7D 27 7D 22`
CRC a Flag	`AD 50 7E`

Prijem tohto paketu vytvorí udalosť RCA a tomu odpovedá akcia IRC a presun do stavu AckRecv. Tento paket spracujeme skôr ako sa začne vytvárať paket na základe akcie SCN, ktorá vznikla po prijme *Configure-Requestu* zo strany ISP vid vyššie. Preto už platia možnosti pre našu stranu ktoré boli vyjednané a to sa aj prejaví v ďalších nami vyslaných paketoch. Vyslané zamietnutie (*Configure-Reject*), ktoré povie ISP, že túto možnosť nepodporuje naša strana teda vypadá následovne:

Význam	**Paket**
Flag a PPP hlavička	`7E FF 03`
LCP Configure-Reject, ID=3, Dĺžka=6 a konfig. možnosti:	`C0 21 04 03 00 06`
Kompresia adresového a kontrolného pola (ACFC)	`08 02`
CRC a Flag	`E7 F1 7E`

Vidieť že vyslaný paket ma komprimované telo, teda nepoužíva sa escape sekvencia na všetky znaky medzi `0-1Fh`, ale iba na kontrolne znaky XON/XOFF. V stavovom automate zostávame v tom istom stave.

ISP prijme našu odpoveď a po spracovaní vyšle Konfiguračný požiadavok, ktorý je skoro úplne totožný s predošlým, len ma iné ID a dĺžku a neobsahuje nami zamietnutú možnosť kompresie adresového a kontrolného pola:

Význam	**Paket**
Flag a PPP hlavička	`7E FF 7D 23`
LCP Configure-Request, ID=5, Dlžka=21 a konfig. možnosti:	`C0 21 7D 21 7D 25 7D 20 7D 37`
Asynchrónna kontrolná mapa (ACCM)=0A00	`7D 22 7D 26 7D 20 7D 2A 7D 20 7D 20`
Kompresie dátového pola (PFC)	`7D 27 7D 22`
Magické číslo=86 61 CC 92	`7D 25 7D 26 86 61 CC 92`
Autentifikačný protokol (AP)=CHAP	`7D 23 7D 25 C2 23 7D 25`
CRC a Flag	`4D 50 7E`

Tento Konf požiadavok obsahuje autentifikačnú metódu protokol CHAP. Tato možnosť nám nevyhovuje, pretože máme implementovaný protokol PAP. Tým pádom vyšleme Configure-Nak, pričom navrhneme v dátovej oblasti (oblasti voľby) konfiguračnej možnosti autentifikácie, aby sa používal PAP a stav sa nemení. Toto sme nemohli spraviť v predošlej odpovedi Configure-Reject, pretože tým by sme zamietli možnosť ako celok, ale nám nevyhovuje len spôsob autentifikácie, nie autentifikácia celá (PAP je plnohodnotná náhrada za CHAP). Tým pádom naša odpoveď bude:

Význam	**Paket**
Flag a PPP hlavička	`7E FF 03`
LCP Configure-Nak, ID=5, Dlžka=23 a konfig. možnosti:	`C0 21 03 05 00 08`
Autentifikačný protokol (AP)=PAP	`03 04 C0 23`
CRC a Flag	`E7 F1 7E`

ISP, ktorý nevyžaduje iba CHAP, tento náš požiadavok prijme a vyšle ďalší Konfiguračný požiadavok, v ktorom okrem ostatných možností požaduje/suhlasí s autentifikáciou pomocou PAP. Tu je patrný rozdiel medzi Configure-Reject a configure-Nak. Pri prvej možnosti sa vylučuje možnosť natrvalo, pričom pri druhej sa navrhuje riešenie, ktoré by vyhovovalo strane príjemca.

Význam	**Paket**
Flag a PPP hlavička	`7E FF 7D 23`
LCP Configure-Request, ID=7, Dlžka=20 a konfig. možnosti:	`C0 21 7D 21 7D 27 7D 20 7D 37`
Asynchrónna kontrolná mapa (ACCM)=0A00	`7D 22 7D 26 7D 20 7D 2A 7D 20 7D 20`
Kompresie dátového pola (PFC)	`7D 27 7D 22`
Magické číslo=31 2A 82 5A	`7D 25 7D 26 86 61 CC 92`
Autentifikačný protokol (AP)=PAP	`7D 23 7D 24 C0 23`
CRC a Flag	`4D 50 7E`

Tieto požiadavky nám vyhovujú a teda vznikla udalosť RCR+ a nej zodpovedajú akcie vyslania SCA, TLU a dostaneme sa do stavu Opened, čo znamená že časť vyjednávania LCP je skončená a prechádzame na autentifikáciu (ak je vyjednaná). Nami vyslaný Configure-Ack bude vypadať následovne.

Význam	**Paket**
Flag a PPP hlavička	`7E FF 03`
LCP Configure-Ack, ID=7, Dlžka=20 a konfig. možnosti:	`C0 21 02 07 00 17`
Asynchrónna kontrolná mapa (ACCM)=0A00	`02 06 00 0A 00 00`
Kompresie dátového pola (PFC)	`07 02`
Magické číslo=31 2A 82 5A	`05 06 86 61 CC 92`
Autentifikačný protokol (AP)=PAP	`03 04 C0 23`
CRC a Flag	`4D 50 7E`

V prípade potreby ukončenia spojenia sa používa tiež protokol LCP. Strana, ktorá požaduje s určitého dôvodu ukončenie spojenie vyšle Terminate-Request ktorý obsahuje iba základné informácie a žiadne dáta resp. konfiguračné voľby.

Význam	**Paket**
Flag a PPP hlavička	`7E FF 03`
LCP Terminate-Request, ID=7, Dlžka=4 a konfig. možnosti:	`C0 21 7D 25 7D 27 7D 20 7D 24`
CRC a Flag	`E4 7D 31 7E`

Tým vznikne udalosť RTR na ktorú je pre každý stav definovaná akcia STA, teda vyslanie Terminate-Ack. Ten by v tomto prípade vypadal.

Význam	**Paket**
Flag a PPP hlavička	`7E FF 03`
LCP Terminate-Ack, ID=7, Dlžka=4 a konfig. možnosti:	`C0 21 7D 26 7D 27 7D 20 7D 24`
CRC a Flag	`29 34 7E`

3.3.4.2 PAP

V dôsledku, že ISP vyžaduje autentifikáciu, čo sme sa dozvedeli vo fáze LCP, vyšleme naše identifikačné údaje pomocou protokolu PAP. Náš požiadavok na autentifikáciu bude vypadať následovne.

Význam	**Paket**
Flag a PPP hlavička	`7E FF 03`
PAP Authentification-Request, ID=1, Dlžka=xx xx:	`C0 23 01 01 xx xx`
Uživatelske meno	`xx xx xx xx xx ... xx xx`
Dlžka hesla	`xx xx`
Heslo	`xx xx ... xx`
CRC a Flag	`xx xx 7E`

Užívatelské meno a heslo je rôzne u jednotlivých operátoroch a jednotlivých paušalov, takže uvádzanie niečoho presného nemá zmysel. Ak nami zaslané údaje sú správne potom nám ISP zašle autentifikačné potvrdenie:

Význam	**Paket**
Flag a PPP hlavička	`7E FF 7D 23`
PAP Authentification-Ack, ID=1, Dlžka=5:	`C0 23 7D 22 7D 21 7D 20 7D 25 7D 20`
CRC a Flag	`8B 3B 7E`

Tým sme dokončili časť autentifikácie a následuje časť sieťových kontrolných protokolov (NCP), v našom prípade IPCP.

3.3.4.3 IPCP

Začiatok vyjednávania spočíva ako u LCP v tom, že každá strana skoro naraz vyšle svoj konfiguračný požiadavok druhej strane. Používa sa pri tom rovnaký stavový automat, len s troska odlišným počiatočným stavom a priebehom. Nami vyslaný požiadavok bude mat tvar:

Význam	**Paket**
Flag a PPP hlavička	`7E FF 03`
IPCP Configure-Request, ID=2, Dlžka=16:	`80 21 01 02 00 0A`
IP adresa=0.0.0.0	`03 06 00 00 00 00`
(Primárny DNS=0.0.0.0	`81 06 00 00 00 00`)
CRC a Flag	`25 56 7E`

V dôsledku, že máme paušál s dynamickou IP adresou, teda očakávame že nám nejakú pridelí ISP je v IP tvare 0.0.0.0. Primárny DNS, NBNS resp. sekundárny DNS, NBNS uvádzam iba ako príklad pre našu aplikáciu, to nie je nutné a preto predpokladajte, že tato možnosť vo vyslanom pakete nebola.

Prijatý požiadavok ISP:

Význam	**Paket**
Flag a PPP hlavička	`7E FF 7D 23`
IPCP Configure-Request, ID=1, Dlžka=10:	`80 21 7D 21 7D 21 7D 20 7D 2A`
IP adresa=192.168.254.254	`7D 23 7D 26 C0 A8 FE FE`
CRC a Flag	`36 22 7E`

Tým sme dostali informáciu akú IP adresu má prístupový bod nášho ISP. V tomto prípade nemôžeme vyjednávať (a ani nie je treba) a preto ju schválime:

Význam	**Paket**
Flag a PPP hlavička	`7E FF 03`
IPCP Configure-Ack, ID=1, Dlžka=10:	`80 21 02 01 00 0A`
IP adresa=192.168.254.254	`03 06 C0 A8 FE FE`
CRC a Flag	`5F 56 7E`

Zároveň nám ISP zaslal konfiguračné odmietnutie, ktoré obsahuje nám pridelenú IP adresu.

Význam	**Paket**
Flag a PPP hlavicka	`7E FF 7D 23`
IPCP Configure-Nak, ID=2, Dlžka=10:	`80 21 7D 23 7D 22 7D 20 7D 2A`
IP adresa=172.24.136.235	`7D 23 7D 26 AC 7D 38 88 EB`
CRC a Flag	`D5 CB 7E`

Tým sme získali IP adresu. To musíme ešte dať najavo tým, že vyšleme konfiguračný požiadavok, ktorý túto IP adresu bude obsahovať:

Význam	**Paket**
Flag a PPP hlavička	`7E FF 03`
IPCP Configure-Request, ID=3, Dlžka=10:	`80 21 01 03 00 0A`
IP adresa=192.168.254.254	`03 06 AC 18 88 EB`
CRC a Flag	`5F 56 7E`

a samozrejme prijmeme odpoveď konfiguračné potvrdenie:

Význam	**Paket**
Flag a PPP hlavička	`7E FF 7D 23`
IPCP Configure-Ack, ID=3, Dlžka=10:	`80 21 7D 22 7D 23 7D 20 7D 2A`
IP adresa=172.24.136.235	`7D 23 7D 26 AC 7D 38 88 EB`
CRC a Flag	`D5 CB 7E`

A tým sme ukoncili fázu NCP pre IP protokol a môže nasledovat prenos datových paketov tohto protokolu.

Kapitola 4

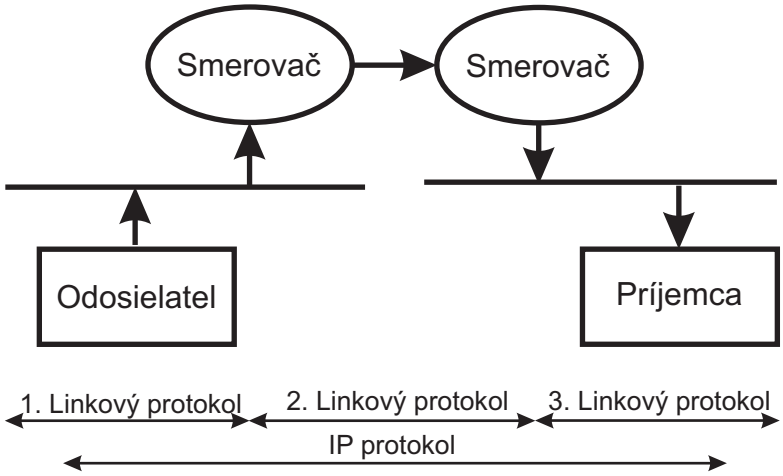
IP a UDP protokol

4.1 IP protokol (Internet protokol)

V predchádzajúcej kapitole 3 som popísal linkový protokol PPP a jeho konfiguračné možnosti. Po vytvorení spojenia, ktoré nám on umožní, chceme prenášať užitočné dáta a na to nám slúži ďalšia rodina protokolov. Najpoužívanejším z nich je IP protokol. Z pohľadu sieťového modelu ISO/OSI je IP protokol na sieťovej vrstve ako ukazuje obr. 2.2. Každý sieťový protokol (NCP), medzi ktoré patrí aj IP, musí byť najprv vyjednaný ak prenos na linkovej úrovni zabezpečuje PPP. Pre IP protokol slúži IP kontrolný protokol (ICMP), ktorý som popísal tiež v predchádzajúcej kapitole 3.3.2.3.

Linkové protokoly (PPP, SLIP, Ethernet) slúžia na dopravu dát medzi dvoma stanicami, prípadne v rámci lokálnej siete. IP protokol na miesto toho dopravuje dáta medzi dvoma ľubovolnými počítačmi v Internetu, teda cez viaceré siete LAN. Dáta sú od odosielatela k príjemcovy dopravovaná cez smerovače (routery). Na ceste sa môže vyskytnúť viac smerovačov. Každý rieši samostatne smerovanie k nasledujúcemu smerovaču, resp. stanici. Linkový protokol sa použije k doprave dát k smerovaču. Ten z linkového paketu vybalí dáta, tzn. časť paketu predstavujúca dáta si ponechá a zbytok linkového paketu zahodí. Následne tieto dáta zabalí do iného linkového paketu rovnakého alebo rozdielneho linkového protokolu. Tento princíp zobrazuje obr. 4.1

Tieto dáta predstavujú práve zabalený sieťový IP protokol do linkového protokolu. Obsah IP paketu nesmie byt smerovačom zmenení. Výnimkou je iba položka TTL z hlavičky IP paketu, ktorý je každý smerovač povinný zmenšiť aspoň o jedna a v prípade zmenšenia na nulu sa IP paket zahadzuje. Týmto spôsobom sa internet bráni nekonečnému prenosu toho istého paketu. Existujú výnimky ako napríklad fragmentácia paketou.



Obr. 4.1: Linkové protokoly a IP protokol.

4.1.1 Štruktúra IP paketu

IP paket sa skladá z hlavičky a prenášaných dát. Hlavička má väčšinou 20 bytov, ale v niektorých prípadoch môže obsahovať aj voliteľné položky. V tom prípade je hlavička o niečo väčšia. Štruktúru IP paketu zobrazuje tabulka 4.1

Jednotlivé položky hlavičky IP paketu majú nasledujúci význam.

Verzia IP(Version) je prvá položka IP hlavičky. Je dĺžky 4 bity a obsahuje a obsahuje verziu IP protokolu. IP protokol sa delí na verziu 4 (IPv4), ktorá ma IP adresu dĺžky 4 byty a verziu 6 (IPv6), ktorá ma IP adresu dĺžky 16 bytov. Vo väčšine prípadov sa používa verzia 4 a je to aj náš prípad. Teda pri verzii 4 sa táto položka rovná hodnote 4.

Dĺžka hlavičky(Header length) obsahuje dĺžku hlavičky udávanú v bytoch, a to spôsobom že hodnota v tomto poli vynásobená 4 (tzv. štyrbytom), dá správny udaj. Teda ak má hlavička 20 bytov (čo vo väčšine prípadov tak skutočne je), je hodnota tejto položky rovná 5. Preto dĺžka hlavičky musí byt v prípade použitia voliteľných položiek násobkom 4. V prípade, že by záhlavie nevyšlo na násobok 4, potom sa musí doplniť bezvýznamnou výplňou aby túto podmienku splňoval. S toho plynie aj obmedzenie na maximálnu dĺžku hlavičky, ktorá môže byt teda 15*4=60 bytov.

Typ služby (type of service - TOS) je položka, ktorá v praxi nenašla praktické použitie a bola pôvodne myslená ako priorita.

Celková dĺžka (total length) IP paketu obsahuje celkovú dĺžku IP paketu v bytoch. Pretože je táto hodnota dvojbytová, maximálna dĺžka IP paketu je 65535 bytov.

Identifikácia IP paketu (identification) obsahuje identifikáciu IP paketu, ktorú vkladá operačný systém odosielateľa. Táto položka spolu s položkami príznaky (flags) a posunu-

<table>
<tr><td colspan="4">0 16</td></tr>
<tr><td>Verzia IP
4 bity</td><td>Dĺžka
hlavičky</td><td>Typ služby
1 byte</td><td>Celková dĺžka IP paketu
2 byty</td></tr>
<tr><td colspan="3">Identifikácia IP paketu
2 byty</td><td>Príznaky
(flags)</td><td>Posunutie fragmentu od počiatku
(fragment offset) - 13 bitov</td></tr>
<tr><td>Doba života paketu
(TTL) - 1 byte</td><td colspan="2">Protokol vyššej vrstvy
(protocol) - 1 byte</td><td colspan="2">Kontrolný súčet s IP hlavičky
2 byty</td></tr>
<tr><td colspan="5">IP adresa odosielateľa (source IP adress)
4 byty</td></tr>
<tr><td colspan="5">IP adresa príjemca (destination IP adress)
4 byty</td></tr>
<tr><td colspan="5">Voliteľné položky hlavičky</td></tr>
<tr><td colspan="5">Prenášané dáta
(nepovinné)</td></tr>
</table>

Tabuľka 4.1: Štruktúra IP paketu

tie fragmentu (fragment offset) využíva mechanizmom fragmentácie.

Doba života paketu (time to live - TTL) slúži k zamedzeniu nekonečného cyklenia IP paketu internetom. Každý smerovač znižuje hodnotu o jedna. Ak sa zníži úroveň na nulu, paket sa zahadzuje v danom smerovači a odosielateľovi je táto informácia signalizovaná protokolom ICMP.

Protokol vyššej vrstvy (protocol) obsahuje číselnú identifikáciu protokolu vyššej vrstvy, ktorá používa IP protokol k svojmu prenosu. V praxi sa nekomunikuje priamo IP protokolom, ale používajú sa protokoly vyššej vrstvy ako napríklad TCP a UDP alebo služobné protokoly ICMP a IGMP. Tieto protokoly sú síce súčasťou protokolu IP ale chovajú sa ako protokoly vyššej vrstvy teda napríklad po IP hlavičke nasleduje ICMP hlavička a potom jej dáta, ktoré prenáša. Číslo protokolu vyššej vrstvy priraďuje tvorcom protokolou organizácia IANA. Priradená čísla je možno zistiť na http://www.iana.org. Najdôležitejšie čísla protokolou sú uvedené v tabulka 4.2.

Protokol	**Číslo protokolu vyššej vrstvy**
ICMP	1
TCP	6
UDP	17

Tabuľka 4.2: Identifikátory vyšších protokolov v IP paketu

Kontrolní súčet s IP hlavičky (header checksum) obsahuje kontrolní súčet, ale iba s hlavičky IP paketu a nie s paketu celého. Problém s kontrolním súčtom spočíva v tom, že keď smerovač zmení nejakú položku v IP hlavičke (napr. TTL čo musí), potom musí aj vypočítať opäť kontrolní súčet, čo vyžaduje určitý výpočetný výkon.
IP adresa odosielateľa a príjemcu (source and destination adress) obsahuje 4 bytovú IP adresu odosielatela alebo príjemcu, pomocou ktorej je každý uživateľ presne identifikovaný.
Voliteľné položky sú využívane ojedinele a spravidla smerovače bývajú nakonfigurované tak, aby IP pakety s použitými voliteľnými položkami boli zahodené.

4.1.2 Protokol ICMP

Protokol ICMP je služobný protokol, ktorý je súčasťou IP protokolu. Slúži k signalizácii mimoriadnych udalostí v sieťach postavených na IP protokolu. ICMP paket býva zabalený do protokolu IP, teda do jeho datovej oblasti. IP paket, ktorý obsahuje ICMP dátový paket bude vypadať vid tabuľka 4.3

IP protokol		
	ICMP protokol	
IP hlavička	ICMP hlavička	ICMP dáta

Tabuľka 4.3: Celková štruktúra ICMP paketu

Protokolom ICMP je možné signalizovať najrôznejšie situácie, ale v praxi mnohé bývajú s bezpečnostných dôvodov zahadzované. Hlavička ICMP paketu je vždy dlhá 8 bytov viz tabuľka 4.4.

Typ	Kód	Kontrolný súcet	Premenná cast záhlavia
1 byte	1 byte	2 byty	4 byty

Tabuľka 4.4: Štruktúra ICMP hlavičky

Typ je parameter na hrubé delenie ICMP paketov
Kód signalizuje o aký presný problém sa jedná teda je to jemné delenie odvodené od *Typu*
Kontrolní súčet vypočítaný s celého ICMP paketu

4.1.2.1 Echo

Existuje mnoho udalostí, ktoré sa dajú signalizovať protokolom ICMP. Najväčšie využitie má ale Echo. Je to jednoduchý nástroj protokolu ICMP, ktorým môžeme testovať dosažitelnosť jednotlivých uzlov v internetu. Žiadateľ vyšle ICMP paket žiadosť o Echo (Echo request) a cieľový uzol je povinný odpovedať paketom Echo. Na tomto princípe pracuje aj všeobecne známy program ping ktorým uživateľ odosiela práve žiadosť o Echo a zobrazuje prípadnú odpoveď Echo. V prípade Echo paketu sú volitelné položky hlavičky ICMP paketu následovne tabuľka 4.5

Typ	Kód	Kontrolný súčet	Premenná časť záhlavia	
1 byte	1 byte	2 byty	4 byty	
0 - odpoveď	0	Kontrolný súčet	Identifikátor	Poradové číslo
8 - požiadavok	0	2 byty	2 byty	2 byty

Tabuľka 4.5: Štruktúra ICMP paketu typu Echo

Identifikátor slúži na identifikáciu ktorá odpoveď patrí ku ktorej žiadosti.
Poradové číslo slúži na presnú identifikáciu paketu.

4.2 UDP protokol

Vo väčšine prípadov sa používa spojenie protokolov TCP/IP, čo je veľmi známa skratka internetovej komunikácie. Pre náš prípad ale stačí protokol UDP, ktorý je z pohľadu ISO-OSI obr. 2.2 tiež na transportnej vrstve ale je o dosť jednoduchší.

Protokol UDP je oproti protokolu IP protokolom vyššej vrstvy. IP protokol slúži na prenos dát medzi ľubovolnými počítačmi v Internete a UDP dopravuje dáta medzi dvoma konkrétnymi aplikáciami bežiacimi na týchto počítačoch. IP adresou ktorú používa IP protokol sa adresuje iba sieťové rozhranie počítača. Portom ktorým sa adresuje v UDP protokole sa adresuje daná aplikácia.

Protokol UDP je nespojová služba (na rozdiel od TCP ktorý je spojový), teda nenavezuje spojenie. Odosielateľ odošle UDP paket príjemcovy a už sa nestará o to, či ho príjemca prijal alebo či sa náhodou paket nestratil (o to sa musí postarať aplikačný protokol). Zapúzdrenie UDP paketu v IP paketu nám zobrazuje nasledujúca tabuľka 4.6

IP protokol		
	UDP protokol	
IP hlavicka 20 bytov	UDP hlavicka 8 bytov	UDP dáta volitelná

Tabuľka 4.6: Celková štruktúra UDP paketu

UDP hlavička dĺžky 8 bytov vypadá následovne tabuľka 4.7

0	16
Zdrojový port (source port) 2 byty	Cieľový port (destination port) 2 byty
Dĺžka dat (UDP length) 2 byty	Kontrolný súčet (UDP checksum) 2 byty

Tabuľka 4.7: UDP hlavička

Vidieť že hlavička protokolu UDP je jednoduchá. Jednotlivé položky a ich význam *Zdrojový a cieľový port* obsahuje číslo portu aplikácie, ktorá UDP paket vyslala a má prijať. Tu je treba upozorniť, že aj pri TCP protokole sa používajú porty, ale ich čísla

nijak nesúvisia navzájom. Teda ak príde UDP paket na port 1234 a TCP paket na port 1234 nemusí sa jednať o rovnakú aplikáciu. UDP protokol má svoju vlastnú sadu čísiel portov.

Dĺžka dat obsahuje dĺžku UDP paketu (dĺžka hlavičky + dáta). Minimálna dĺžka je 8 bytov tj. v prípade že UDP paket neobsahuje dáta.

Kontrolný súčet slúži podobne ako u PPP a IP protokoloch na bezpečnostné účely. U UDP protokolu sa dá ale vypnúť, teda sa nemusí vypĺňať v prípade že toto pole vyplníme nulou. Tým dávame najavo že sa nepoužíva. V prípade použitia sa vypočítava z pseudohlavičky, ktorá sa skladá z časti IP a UDP hlavičky. Vypadá následovne tabuľka 4.8

<table>
<tr><td colspan="3">IP adresa odosielateľa (source adress)
4 byty</td></tr>
<tr><td colspan="3">IP adresa príjemca (destination adress)
4 byty</td></tr>
<tr><td>Binárne nuly</td><td>Protokol vyššej vrstvy
(protocol) - 1 byte</td><td>Celková dĺžka IP paketu
2 byty</td></tr>
<tr><td colspan="2">Zdrojový port (source port)
2 byty</td><td>Cieľový port (destination port)
2 byty</td></tr>
<tr><td colspan="2">Dĺžka dát (UDP length)
2 byty</td><td>Kontrolný súčet (UDP checksum)
2 byty</td></tr>
<tr><td colspan="3">Dáta</td></tr>
<tr><td colspan="3">Prípadná výplň na párny počet bytov</td></tr>
</table>

Tabuľka 4.8: Štruktúra pseudohlavičky

UDP protokol umožňuje fragmentáciu podobne ako IP protokol. Avšak snaha u UDP protokolu smeruje k tomu, aby sa nepoužívala. Ďalšou odlišnosťou UDP protokolu je možnosť posielania obežníkov. To znamená že adresátom UDP paketu nemusí byt iba jedna adresa, tj. jedno konkrétne sieťové rozhranie počítača. Môže ním byt skupina staníc. Takto adresované pakety sa nazývajú obežníky a sú typu broadcast, keď pakety sú vysielané všetkým alebo multicast keď sú adresované určitej skupine a tým dochádza k úspore prenosovej kapacity ciest.

Kapitola 5

Prenosová jednotka a GPRS

5.1 Úvod

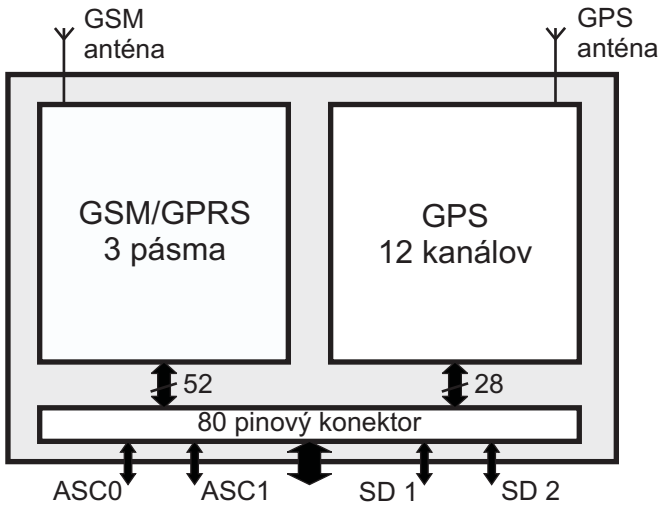
Siemens XT55, je prvý modul GSM spoločnosti Siemens, ktorý obsahuje integrovaný modul GPS pre satelitnú navigáciu. Podporuje tri-band GSM/GPRS na frekvenciách GSM 900MHz, GSM 1800MHz a GSM 1900MHz. Umožňuje kódovacie schémata CS1, CS2, CS3 a CS4 pre GPRS dátový prenos. Kombinácia týchto dvoch technológii, umožňuje relatívne lacné realizovanie systémov sledovania vozidiel, tovaru, ľudí.

5.2 Hardware XT55

O Module XT55 môžeme uvažovať, ako o dvoch nezávislých častiach - GSM a GPS. Každá s týchto častí má svoje sériové rozhrania obr. 5.1.

5.2.1 GSM časť

GSM časť modulu, má dve asynchrónne sériové rozhrania ASC0 a ASC1. Dátový rámec pre obe je konfigurovaný na 8 dátových bitov, bez parity, 1 stop bit. Komunikácia je možná rýchlostiach 1200, 2400, 4800, 9600, 38400, 115200, 230400 bit/s. Obe rozhrania umožňujú HW i SW (XON/XOFF) kontrolu toku dát. GSM časť sa chová ako DCE. ASC0 má nasledujúce vlastnosti.



Obr. 5.1: Architektúra XT55.

- 8 signálov
- obsahuje 2 dátové signáli GSM_TXD0, GSM_RXD0, stavové signáli GSM RTS0, GSM CTS0 a riadiace signáli modemu GSM_DTR0, GSM_DSR0, GSM_DCD0 a GSM_RING0.
- je určený predovšetkým na hlasové volania, CSD, fax, GPRS a ovládanie GSM časti pomocou AT príkazov
- umožňuje pracovať v multiplexnom režime pri ktorom je rozhranie rozdelené na 3 virtuálne kanáli.
- dotazovanie na signál GSM_DTR sa deje jeden krát za sekundu
- podporuje automatickú detekciu komunikačnej rýchlosti (autobauding)
- autobanding nemožno používať pri práci v multiplexnom režime
- používa sa pri upgrade firmwaru XT55

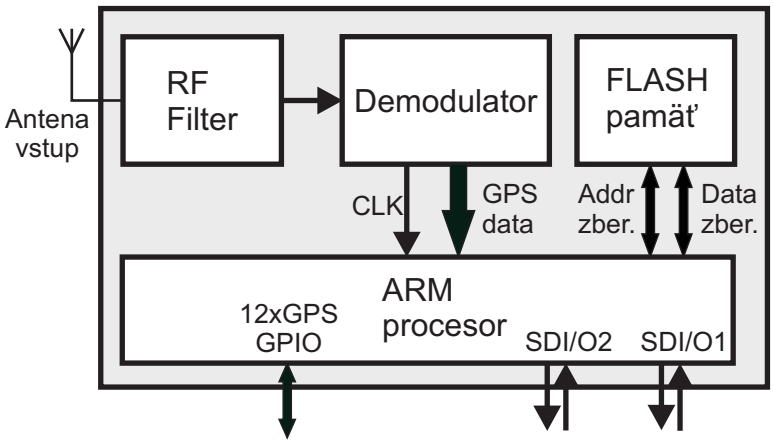
Druhé seriové rozhranie ASC1, na rozdiel od ASC0 neobsahuje riadiace signáli modemu a jeho funkcie sú v porovnaní s ASC0 obmedzené.

- 4 signáli
- obsahuje 2 dátové signáli GSM_TXD1, GSM_RXD1, stavové signáli GSM_RTS1, GSM_CTS1

- je určený predovšetkým na hlasové volania, GPRS a ovládanie GSM časti pomocou AT príkazov. Nie je možné použiť pre CSD, fax a prácu v multiplexnom režime.
- ak pracuje ASC0 v multiplexnom režime nemožno použiť ASC1

5.2.2 GPS časť

Prijímacia časť GPS modulu sa skladá zo štyroch hlavných častí: RF dolní frekvenční konventor, digitálna demodulácia, ARM mikroprocesor a interná 1 MB FLASH pamäť obr. 5.2.



Obr. 5.2: Architektúra GPS prijímača.

Funkčnosť jednotlivých častí je následovná

- V RF časti je GPS signál zachytený anténou zosilnení, vyfiltrovaný a premenený na stredné frekvencie(IF). Následne je v A/D konvertoru analógový signál premenený na digitálny.
- Digitálny signál sa následne demoduluje (možnosť až 12 kanálov) na signáli z jednotlivých satelitov
- Mikroprocesor spočítava na základe prijatých dát zo satelitov a algoritmu, ktorý je uložený v internej FLASH pamäti polohu, rýchlosť a čas.
- Následne tieto dáta vyšle sériovým rozhraním v podobe NMEA vety

GSP časť modulu je vybavená 2 asynchrónnymi sériovými rozhraniami, ktoré sú až na napäťové úrovne kompatibilné s RS 232. Napäťové úrovne sú CMOS 3,3V. Formát

dátového rámca, je pre oba nastavený na 8 dátových bitov, bez parity, 1 stop bit. Prvé rozhranie SD1 má nasledujúce parametre

- 2 signáli
- obsahuje dátové signáli SDI(vysielanie) a SDO(príjem)
- podporovaná rýchlosť 4800 bps zo štandartným firmwarom, 9600 bps s TCP/IP firmwarom

Druhé rozhranie SD2 má nasledujúce parametre

- 2 signáli
- obsahuje dátové signáli SDI(vysielanie) a SDO(príjem). Je určený na komunikáciu s GSM/GPRS častou modulu XT55
- podporovaná rýchlosť 9600 bps

Pre podrobnejšie informácie o hardwaru modulu Siemens XT55 vid [9].

5.2.3 Firmware

Firmware XT55 je uložený v FLASH pamäti časti GPS.Pre informácie o spôsobe nahrávania firmwaru vid [12]. Štandartne obsahuje už zmienený algoritmus vypočítavania dát o polohe zo surových dát prijatých zo satelitov. Existujú okrem toho, ešte dva nadštandartné softwarové balíky a to AVL (*Automatic Vehicle Location*) software a TCP/IP software.

5.2.3.1 AVL

umožňuje rozšírenie možností práce s GPS. Jedno s jeho hlavných použití, je samostatná činnosť XT55 bez riadiacej jednotky. Ovládanie v tomto prípade sa realizuje CSD spojenia. Jeho parametre sú následovné:

- NMEA dáta môžu byt čítané na porte SD1
- Možnosť nakonfigurovania 10 telefónnych čísiel na vzdialený prístup (vzdialený požiadavok o pozícii, real-time sledovanie pomocou GSM)

- Všetky nastavenia a funkcie prístupné cez CSD spojenie a sériový port
- Možnosť konfigurácie funkcie History, ktorá ukladá NMEA dáta do pamäti FLASH (až 10 000 NMEA viet uchovaných v FLASH pamäti, filter na ukladanie správ na základe rýchlosti a času)
- Možnosť cyklického posielania SMS správ obsahujúcich dáta pozície
- 2 tlačítka alarmov
- Možnosť zasielania AT príkazov cez SD1

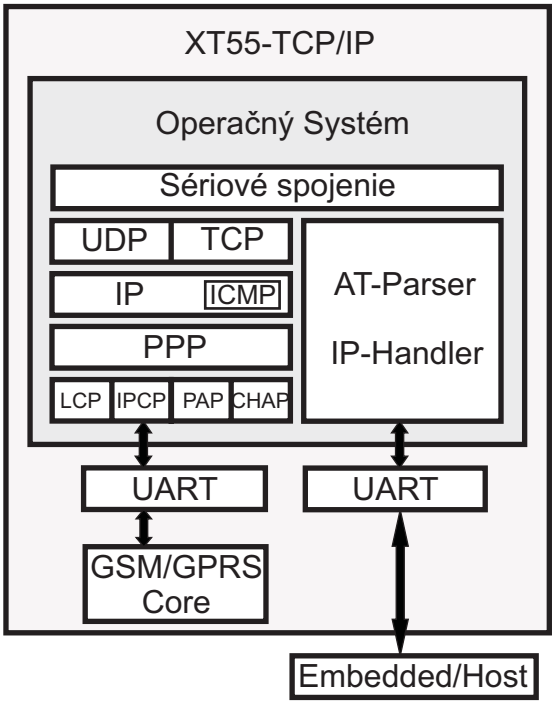
Pre bližšie informácie vid [14].

5.2.3.2 TCP/IP

umožňuje nadviazanie TCP/IP spojenia zo zvolenou stanicou v sieti danou IP adresou pomocou IP príkazov. V prípade jeho použitia, odpadá nutnosť realizácie TCP/IP stacku v externej riadiacej jednotke. V konfiguračnej časti sa nastaví požadované parametre operátora ako PDP kontext, uživateľské meno a heslo, IP adresa cieľovej stanice a vytočí sa spojenie. O tvorbu TCP paketov sa stará modul XT55 a externá jednotka vysiela/prijíma iba čisté dáta. Zobrazenie TCP/IP stacku v XT55 vid obr. 5.3

Jeho možnosti sa dajú zhrnúť:

- TCP/IP stack bežiaci na ARM mikroprocesore časti GPS
- IP príkazy pre možnosť konfigurácie GPRS a TCP/IP
- IP príkazy umožňujúce pripojenie XT55 TCP/IP na server
- Sériové spojenie cez TCP/IP. Všetky dáta ktoré prichádzajú sériovým portom SD1 sú následne zabalené do TCP paketov a prenesené na vzdialený server. Prichádzajúce TCP pakety sú preložené a prenesené na sériový port SD1.
- Podpora vyššých protokolov (napr. HTTP, HTTPS, SMTP)
- Nastavenie sa ukladá do FLASH pamäti
- Automatické spustenie a pripojovacia funkcia. Po uložení nastavenia sa pripojenie na server prevádza automaticky.

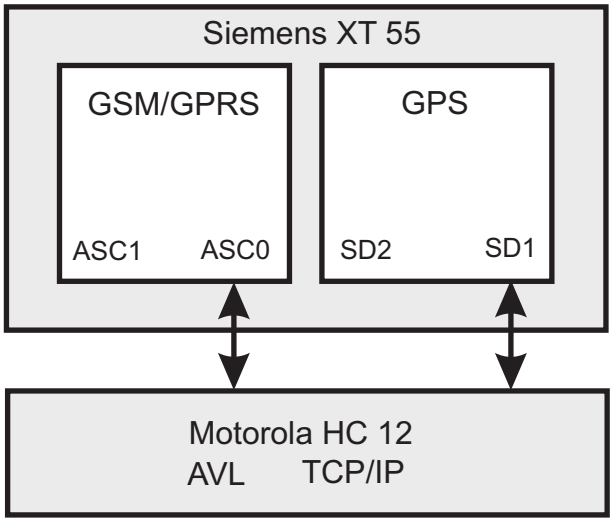


Obr. 5.3: TCP/IP stack v XT55.

Tu dochádza ale k jednému zásadnému problému. V dôsledku, že firmware je nahrávaný do FLASH pamäte časti GPS, dôjde k prepísaniu jej pôvodného obsahu. S toho dôsledku prídeme o možnosť prijímania dát GPS, pretože na ich interpretáciu zo surových dát získaných zo satelitov je nutný algoritmus výpočtu pôvodne umiestnený v pamäti FLASH. Tým modul XT55 prichádza o svoju GPS časť a stáva sa z neho obyčajný GSM/GPRS modul. Pre bližšie informácie vid [13]. To ale pre našu aplikáciu je nevyhovujúce a s toho dôsledku nám možnosť nahrania tohto firmwaru odpadá. TCP/IP stack musíme realizovať na externej riadiacej jednotke.

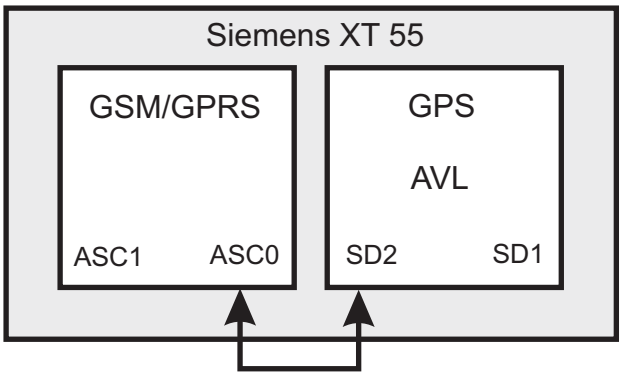
AVL firmware nespôsobuje stratu možnosti prijímania dát GPS. Teda bez možnosti TCP/IP stacku realizovaného priamo v XT55 nám zostávajú nasledujúce možné hardwarové konfigurácie.

XT55 a externá riadiaca jednotka s AVL a TCP/IP: V tomto zapojení, sú sériové kanáli ASC0 a SD1 spojené s externou riadiacou jednotkou, v ktorej je implementovaný software AVL i TCP/IP. Nemáme možnosť ukladať do FLASH pamäte NMEA správy. Taktiež, musíme realizovať PPP a TCP/IP stack pre vzájomnú komunikáciu medzi XT55 a riadiacou jednotkou. Teda celý vyjednávací algoritmus na nadviazanie GPRS spojenia a spracovanie paketov vyšších sieťových protokolov (IP,TCP,UDP), musíme realizovať v riadiacej jednotke obr. 5.4.



Obr. 5.4: XT55 a externá riadiaca jednotka.

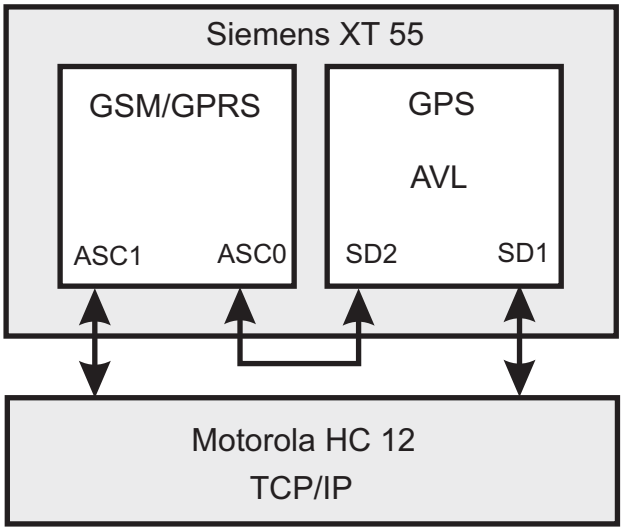
XT55 s AVL: V tomto prípade, máme nahratý AVL firmware. Sériové kanáli ASC0 a SD2 sú externe prepojené a externá riadiaca jednotka nie je prítomna. Môžeme využívať všetky už zmienené možnosti AVL ale prichádzame o možnosť GPRS obr. 5.5.



Obr. 5.5: XT55 s AVL.

XT55 s AVL a externá riadiaca jednotka s TCP/IP: V XT55 je firmware AVL, sériové kanálu ASC0 a SD2 sú externé prepojené a kanáli ASC1 a SD1, sú pripojené k externej riadiacej jednotke na ktorej je implementovaný TCP/IP stack. Môžeme využívať všetky už zmienené možnosti AVL a taktiež GPRS obr. 5.6.

V dôsledku, hardwarového riešenia vývojovej dosky z modulom Siemens XT55 a zabezpečenia požadovanej funkčnosti nášeho systému, som zvolil riešenie XT55 a externá riadiaca jednotka s AVL a TCP/IP. AVL na riadiacej jednotke nebude obsahovať všetky možnosti, ktoré by sme mali v prípade originálneho firmwaru. V tejto etape vývoja sys-



Obr. 5.6: XT55 s AVL a externá riadiaca jednotka s TCP/IP.

tému, požadujeme hlavne ukladanie prichádzajúcich NMEA viet do pamäte riadiacej jednotky. HÍbka ich spracovania nie je až tak moc podstatná. Naopak, podstatné pre náš systém je realizácia GPRS spojenia na prenos dát. Tu musíme vytvoriť priamo na riadiacej jednotke PPP a TCP/IP (resp. UDP/IP) stack, ktorý bude dáta určené k vyslaniu baliť do paketov a naopak pakety získané rozbaľovať.

5.3 Inicializácia GPRS

5.3.1 Prihlásenie sa do siete GSM

Pred tým než sa pokúsime nadviazať GPRS spojenie, musíme byt pripojený do GSM siete. To znamená, hlavne poskytnúť operátorovi našu identifikáciu. Táto úloha pozostáva z viacerých kontrolných mechanizmov, popísaných napríklad v diplomovej práci pána Černého [2]. S uživateľského hľadiska to znamená hlavne nutnosť pripojenej SIM karty a zadanie správneho PIN kódu.

V prípade, že nutnosť zadania PIN kódu po každej aktivácii je priamo vypnutá v SIM karte XT55 sa pripojí automaticky. Inak musíme zadať PIN kód pomocou AT príkazu
Príkaz: `AT+CPIN=1234(CR)` Odpoveď: `OK`

Ďalším krokom je definovanie PDP kontextu (packet data protocol context). Jeho

súčasťou je adresa GGSN uzlu, ktorý slúži ako brána do vonkajšej siete, špecifikácia dohodnutej kvality služieb (QoS) a vlastná adresa v rámci siete. PDP kontext sa získava počas operácie GPRS attach a následne musí byt aktivovaný, aby bol uzol viditelný v sieti. Jeho definovanie môže nastať nasledujúcimi spôsobmi.

- Ak nie je požadovaný žiadne extra nastavenie modemu, na vytvorenie PDP sa použije vytočenie čísla `*99#`. V tomto prípade sa použije prednastavené hodnoty, ktoré pochádzajú s posledne vytvoreného spojenia
- Ak boli vyplnené extra nastavenia, potom na miesto vytočenia *99(krizik) sa vytáča `*99***(CID)#` kde CID je definovaný PDP kontext.
- PDP kontext môže byt definovaný aj manuálne aplikáciou uživateľa.

Na definovanie PDP kontextu sa používa AT príkaz
`AT+CGDCON=1,IP,internet.t-mobile.cz`
Bližšie informácie vid [11].

Po dokončení definície PDP kontextu, môžu byť nastavený QoS profil, ktorý definuje požadovanú a minimálnu kvalitu služieb. Slúžia na to AT príkazy
`AT+CGQREQ=x.x.x.x.x.x(CR)`
`AT+CGQMIN=x.x.x.x.x.x(CR)`

5.3.2 Aktivácia PDP kontextu

5.3.2.1 Diskrétna aktivácia PDP kontextu

nastáva po nasledujúcom slede krokov

- Zadanie čísla PIN
  Príklad:

```
AT+CPIN=1234(CR)
OK
```

- Definovanie PDP kontextu
  Príklad:

```
AT+CGDCON=1,IP,internet.t-mobile.cz(CR)
```

- Definovanie QoS
  Príklad:

```
AT+CGQREQ=1,2,4,3,0,0(CR)
AT+CGQMIN=1,0,0,0,0,0(CR)
```

- Pripojenie GPRS
  Príklad:

```
AT+CGATT=1(CR)
OK
```

- Aktivácia PDP kontextu
  Príklad:

```
AT+CGACT=1,(CID)(CR)
OK
```

  Tu môže byt definovaných aj viac PDP kontextou.

- Vstup do PPP online módu tzn. GPRS dáta módu a aktivácii PPP stacku
  Príklad:

```
AT+CGDATA=PPP,(CID)(CR)
CONNECT
(PPP DATA)
```

  Ak dostaneme odpoveď `CONNECT`, dostali sme sa do dátového módu a nemôžme už používať AT príkazy. Iba v prípade zaslania príkazu `+++` ktorý nás prepne s dátového módu do príkazového. Naspäť prepnutie do dátového módu sa robí pomocou príkazu ATO.

5.3.2.2 Aktivácia PDP kontextu kompatibilná s modemom

- Nedefinovaný PDP kontext použije sa vytočenie už zmieneného čísla `*99#`.
  Príklad:

```
ATD*99#
CONNECT
(PPP DATA)
```

- Definovaný PDP kontext nastavením parametrov a kvality služieb
  Príklad:

```
AT+CGDCON=1,IP,internet.t-mobile.cz
OK
AT+CGQREQ=1,2,4,3,0,0
OK
ATD*99***1#
CONNECT
(PPP DATA)
```

Pre celý zoznam AT príkazov ktoré podporuje XT55 vid [10]. Nastavenie XT55 ako modemu pripojeného k PC a jeho nakonfigurovania vid [11].

Kapitola 6

Implementácia riadiacej jednotky

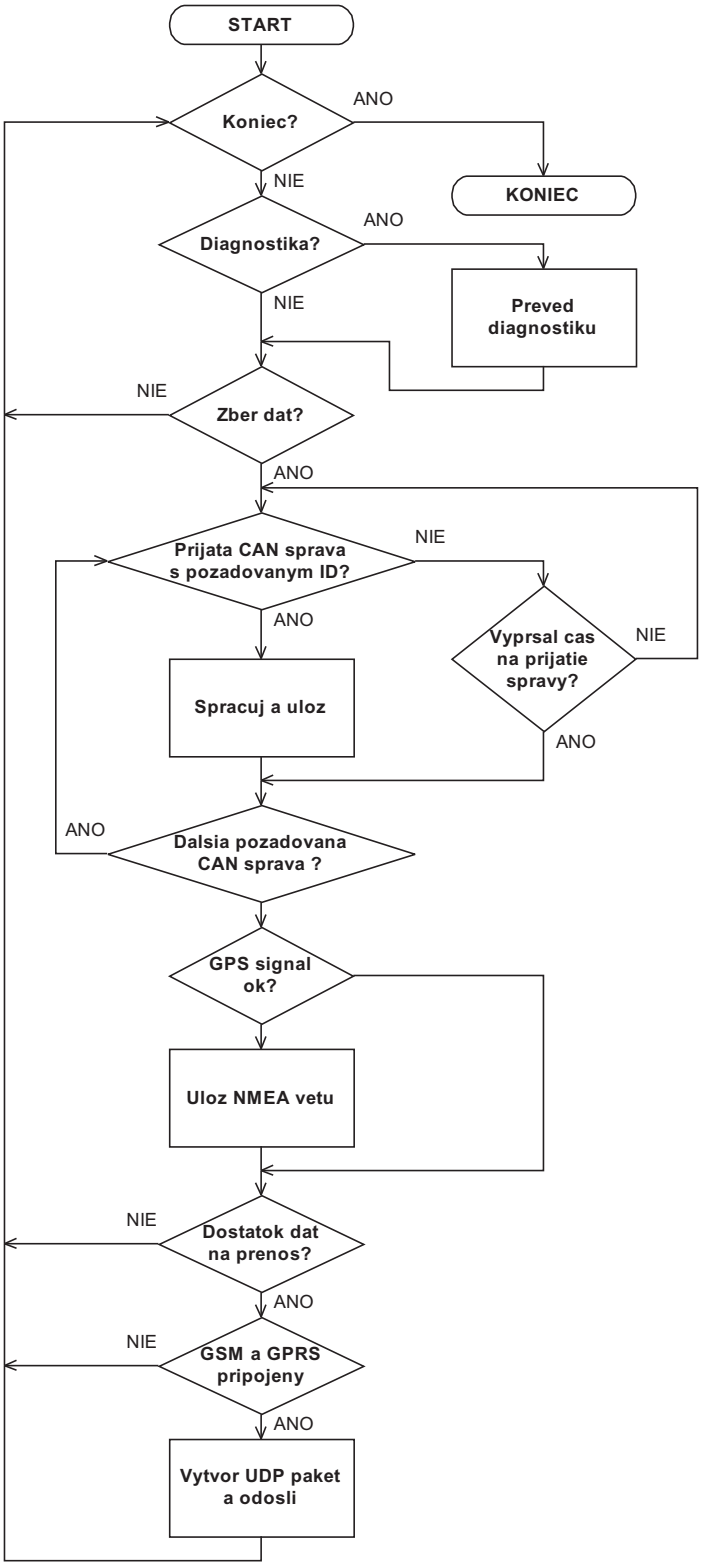
6.1 Hlavný algoritmus

Funkčnosť riadiacej jednotky, zabezpečuje na najvyššej úrovni hlavny resp. riadiaci algoritmus. Musí spĺnať základné požiadavky:

- Zabezpečenie prijmu nami požadovaných dát - Musíme byť schopný jednoducho nadefinovať presne, ktoré CAN správy a formát NMEA viet chceme prijímať.
- Definovanie podmienok na prenos dát - Možnosť nastavenia maximálnej veľkosti paketov a prípady ich vyslania
- Možnosť diagnostiky - Schopnosť detekcie pripojenia do GSM a GPRS siete.
- Ošetrenie chybových stavou - Vypadnutie GSM/GPRS by nemalo mať vliv na ukladanie dát. Neprijatie požadovanej CAN správy by nemalo viesť k nekonečnému čakaniu na ňu. Taktiež ošetrenie pri výpadku GPS signálu.

Na základe týchto požiadavkov som vytvoril algoritmus vid obr. 6.1. Skladá sa z nekonečného cyklu, ktorý je možné opustiť v prípade požiadavku na ukončenie činnosti systému. Následuje možnosť diagnostiky, ktorá sa vykonáva vždy pri spustení systému a potom cyklicky s nadefinovanou časovou periodou. Tým sa získavajú informácie, potrebné na riadenie algoritmu ako je napríklad dostupnosť GSM resp. GPRS, testovanie pripojenia CAN zbernice a prijímania GPS signálu.

Nasleduje časť ukladania príchodzích dát ak to požadujeme. Ako prvé sa ukladajú požadované CAN správy. Nastavý sa maska prijmu na požadovaný identifikátor a spustí sa čítač čakania na správu. V prípade, že správa nepríde do doby určenej čítačom, prejde



Obr. 6.1: Vývojový diagram hlavného algoritmu.

sa na ďalšiu definovanú spravu alebo ak to bola posledná sprava v zozname prejde sa na príjem GPS dát. V prípade jej prijatia, sa spracuje a uloží. V prípade že prijímame signal GPS spracuje sa a uloží príchodzia NMEA veta. Ak neprijímame, potom sa preskočí ukladanie a pokračuje následujúci krok v algoritmu.

Po zberu dát s jedného cyklu nastane časť na ich prípadné vyslanie. V prípade nazbierania požadovaného množstva dát a pripojenia do GSM/GPRS siete vyšleme UDP paket a nastáva návrat na začiatok cyklu.

Náš riadiaci systém nie je Real-Time. V prípade prijmu CAN správy alebo MNEA vety počas vysielania UDP paketu sú tieto data zahodené. Táto realizácie je možná vďaka tomu, že nepracujeme s časovo kritickými udalosťmi, tzn. ich stav sa nemení podstatne počas jedného cyklu riadiaceho algoritmu. V prípade že by sme chceli obsluhovať časovo kritické udalosti museli by sme implementovať v riadiacej jednotke operačný systém realneho času ako napr. OSEK

V nadchádzajúcich kapitolách sa budem venovať priamo implementácii jednotlivých protokolov.

6.2 PPP stack

Implementácia PPP stacku na mikrokontroléri musí splňovať štandarty, ktoré sú definovane v normách RFC. Je to v dôsledku zamedzenia chýb komunikácie, ktoré by mohli byť spôsobené neúplnosťou alebo chybou interpretácie protokolu. Teoreticky PPP paket môže byt dlhý 1500 bytov takže nároky na pamäť RAM musia byt prispôsobené vysielacím a prijímacím bufferom. Taktiež, je nutné aby v ten istý moment bežali štyri stavové automaty(modem, LCP, PAP a IPCP). Medzi LCP a IPCP existuje spoločne využiteľné prvky automatu, ale rozdiely sú na toľko veľké že zneefektívňujú spoločné použitie kódu. Základné úlohy ktoré musí PPP spĺnať sú:

- Príjem a vysielanie
- Dekódovanie prijatých paketov
- Stavový automat
- Obsluha udalostí

6.2.1 Príjem a vysielanie

Na príjem a vysielanie sa používajú receive a transmit buffer. Ich dĺžka je maximálni počet bytov, ktorý môže obsahovať PPP paket. Uloženie paketu v bufferi je navrhnuté tak, že prvý prvok buffera odpovedá poli paketu *Adresa* (`FFh`). Ďalej potom nasleduje paket v podobe uvedenej v kapitole PPP. Teda začiatočný *flag* `7Eh` nie je uložený v bufferi ani pri prijímaní ani pri vysielaní. Taktiež koncový *flag* `7Eh` nie je uložený. Tieto značky slúžia na detekciu začiatku resp. konca paketu. V prípade detekcie 7Eh nasledovaným bytom FFh, dostávame informáciu o začiatku paketu. Po detekcii začiatku paketu, si zapamätáme že sme uprostred prijímania paketu, teda v prípade prijatia 7E vieme že sa jedná o ukončovací znak. Znaky ktoré odpovedajú CRC, pri prijme uložíme do buffera, ale ich informáciu potrebujeme iba k tomu aby sme vypočítali správnosť prenosu. Teda v prípade správneho prenosu tieto byty vymažem s buffera spôsobom, že vrátim dĺžku paketu o 2 byty menšiu teda CRC, ktorá predstavuje posledné dva prijate byty sa ďalej v programe nepoužije.

Vďaka tomuto spôsobu spracovania, môžeme PPP paket prerobiť na architektúru hlavička - dáta, ktorý je bežný pri protokoloch vyššých vrstiev ako napr. IP, UDP, TCP. Hlavičku PPP paketu, ktorá vychádza s popisu protokolu máme definovanú pomocou položiek štruktúry, ktoré nám umožňujú jednoduchý a rýchly prístup k jednotlivým polom paketu.

```
typedef struct
{
        BYTE addr; //adresa
        BYTE ctrl; //kontrolný znak
        WORD pcol; //protokol zapúzdreny v PPP paketu
} PPPHDR;
```

Celý PPP paket, tzn. hlavička i dáta sú definované tiež ako štruktúra, ktorá obsahuje ako svoje položky štruktúru PPP hlavičky a dáta sú reprezentované statickým polom znakov.

```
typedef struct
{
        PPPHDR p; //hlavička paketu
        BYTE buff[PPP_MRU];  //dáta tzn. protokol zapúzdrený v PPP
} PPPKT;
```

Práca s paketom, potom pozostáva z vytvorenia pointera na štruktúru `PPPKT` a odovzdaním mu adresy začiatku prijímacieho resp. vysielacieho buffera. Tým, že štruktúra neobsahuje voľné pamäťové miesta vďaka zarovnaniu typov na párne adresy prekladačom, sa nám prekryje štruktúra PPPKT s paketom a jednotlivé polia paketu budú prístupné priamo pomocou položiek tejto štruktúry.

Funkcie slúžiace na príjem a vysielanie pracujú zo sériovou linkou a úložnými bufferami. V prípade prijatia znaku po sériovej linke, musí nastať jeho spracovanie a prípadné uloženie do prijímacieho buffera (receive buffer). Ak sa jedná o znak v rámci paketu, tak sa musí vypočítať CRC, ktoré po konci prijímania paketu určí jeho správnosť. Na to slúži funkcia

```
void poll_net(void)
```

Použité premenné:

Názov	**Typ**	**Význam**
`b`	`BYTE`	Obsahuje prečítaní znak zo sériovej linky pomocou funkcie `CheckIncomingUart2Data`
`saver`	`BYTE`	Určuje či sa má daný znak uložiť do receive buffera (`rxbuff[]`)
`lastb`	`BYTE`	Posledne prijatý znak
`crc`	`BYTE`	Postupne vypočítavaná hodnota CRC

Algoritmus funkcie vid obr. 6.2. Táto funkcia prijme prípadný PPP paket, skontroluje jeho správnosť a uloží ho do receive buffera. Pokazené pakety zahadzuje.

Pri vysielaní PPP paketu predpokladáme, že tento paket máme vytvorený v transmit bufferi. Začiatočný a koncový znak `7E` a `crc` sa vysielajú priamo teda nenachádzajú sa v transmit bufferi. Vysielanie obstarávajú funkcie

```
void transmit_ppp(WORD dlen, BYTE mode)
void send_ppp_byte(BYTE b)
```

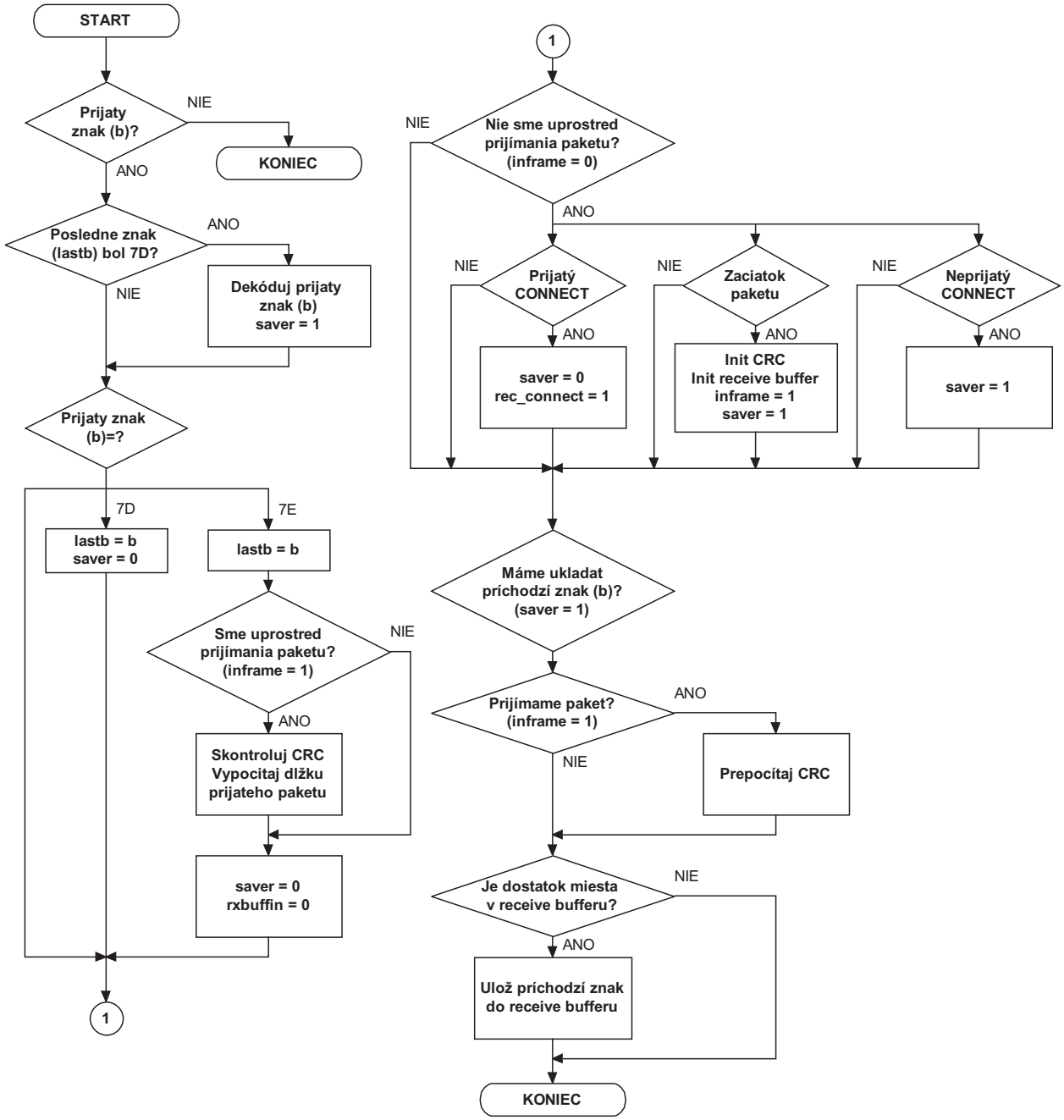
Algoritmi týchto funkcii sú znázornené na obr. 6.3 a obr. 6.4

Použité premenné v `transmit_ppp`:

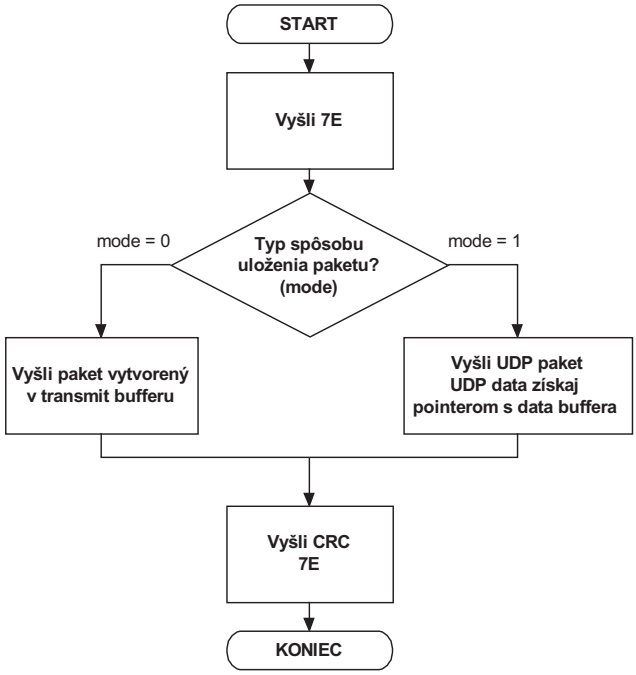
Názov	**Typ**	**Význam**
`mode`	`BYTE`	Definuje akým spôsobom sa získajú dáta k prenosu. Definované kvôli UDP paketom
`dlen`	`WORD`	Dĺžka paketu pripraveného na vysielanie v transmit bufferi (`txbuff[]`)

Použité premenné v `send_ppp_byte`:

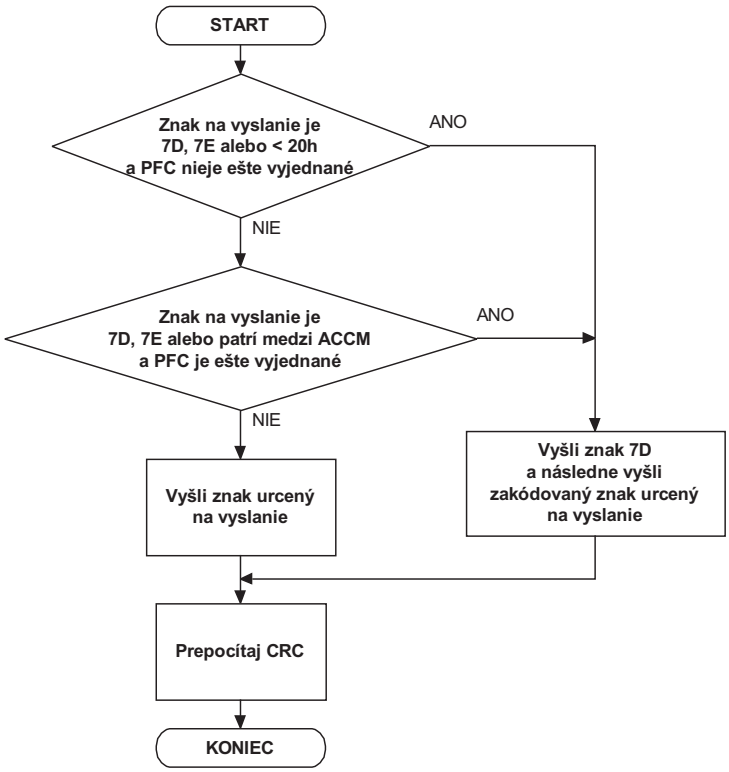
Názov	Typ	Význam
b	BYTE	Byte ktorý chceme vyslať



Obr. 6.2: Vývojový diagram funkcie poll_net.



Obr. 6.3: Vývojový diagram funkcie transmit_PPP.



Obr. 6.4: Vývojový diagram funkcie send_ppp_byte.

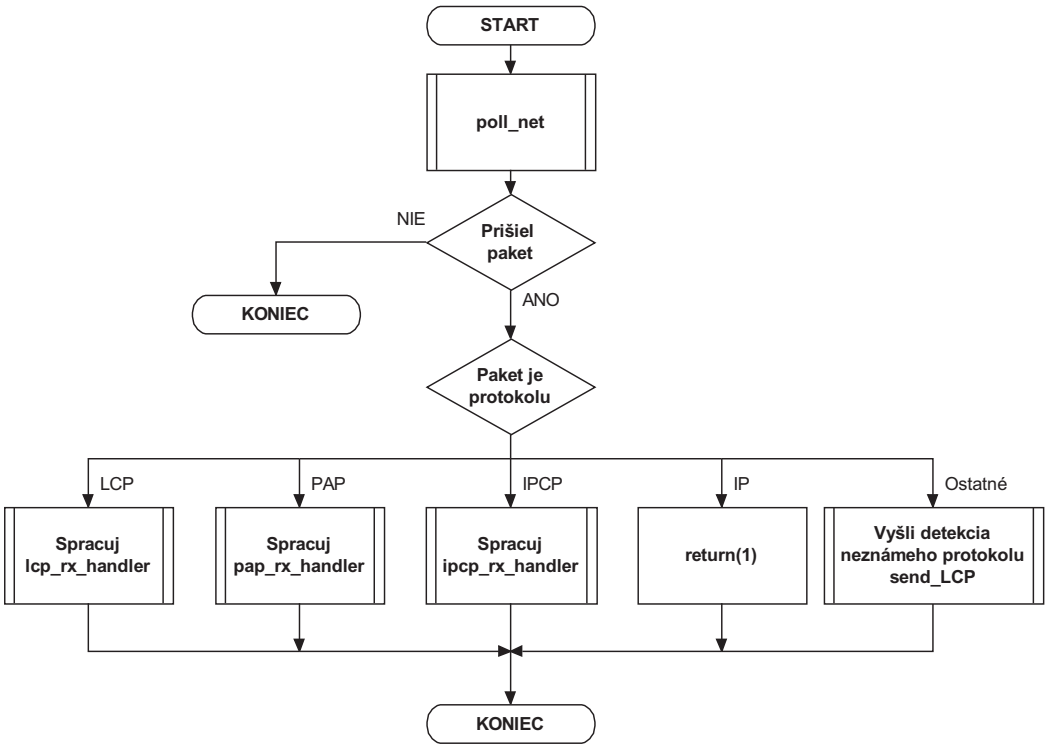
6.2.2 Dekódovanie

Po úspešnom prijatí paketu, ho musíme dekódovať aby sme zistili informáciu ktorú nesie. To znamená, zistenie o aký typ protokolu sa jedná a zavolanie príslušného handlera, ktorý bude paket ďalej spracovávať. V prípade prijatia paketu s neznámym protokolom nastane jeho odmietnutie.

```
void get_net(void)
```

Algoritmus funkcie vid obr. 6.5. Použité premenné v `void get_net`:

Názov	Typ	Význam
`net_rxin`	`WORD`	V prípade, že sme prijali celý paket udáva jeho dĺžku. Inak je nulový
`ppp_col`	`WORD`	Obsahuje protokol, ktorý je zapúzdrený v PPP pakete. Možnosti, ktoré podporujeme sú LCP, PAP, IPCP a IP



Obr. 6.5: Vývojový diagram funkcie get_net.

Pre každý protokol existuje handler, ktorý ho obsluhuje. Pre názornosť uvediem handler, ktorý obsluhuje LCP.

```
void lcp_rx_handler(void)
```

Algoritmus funkcie vid obr. 6.6. Použité premenné v `lcp_rx_handler`:

Názov	**Typ**	**Význam**
`ppp_code`	`WORD`	Obsahuje číselnú identifikáciu významu prijatého paketu LCP. Teda či ide o *Configure-Request*, *Configure-Ack/Nak* alebo iné
`opt`	`BYTE`	Obsahuje číselnú identifikáciu konfiguračnej možnosti
`optlen`	`BYTE`	Obsahuje dĺžku číselnej identifikácie konfiguračnej možnosti
`rejects`	`BYTE`	V prípade že konfiguračný požiadavok obsahuje možnosť, ktorú nemôžeme spracovať sa inkrementuje
`non_ack`	`BYTE`	V prípade že konfiguračný požiadavok obsahuje možnosť, ktorú môžeme spracovať ale nevyhovuje nám jej hodnota sa inkrementuje
`xxx_rec`	`xxx`	Premenné slúžiace na detekovanie možností, ktoré sú vyjednávané

6.2.3 Stavový automat a obsluha udalostí

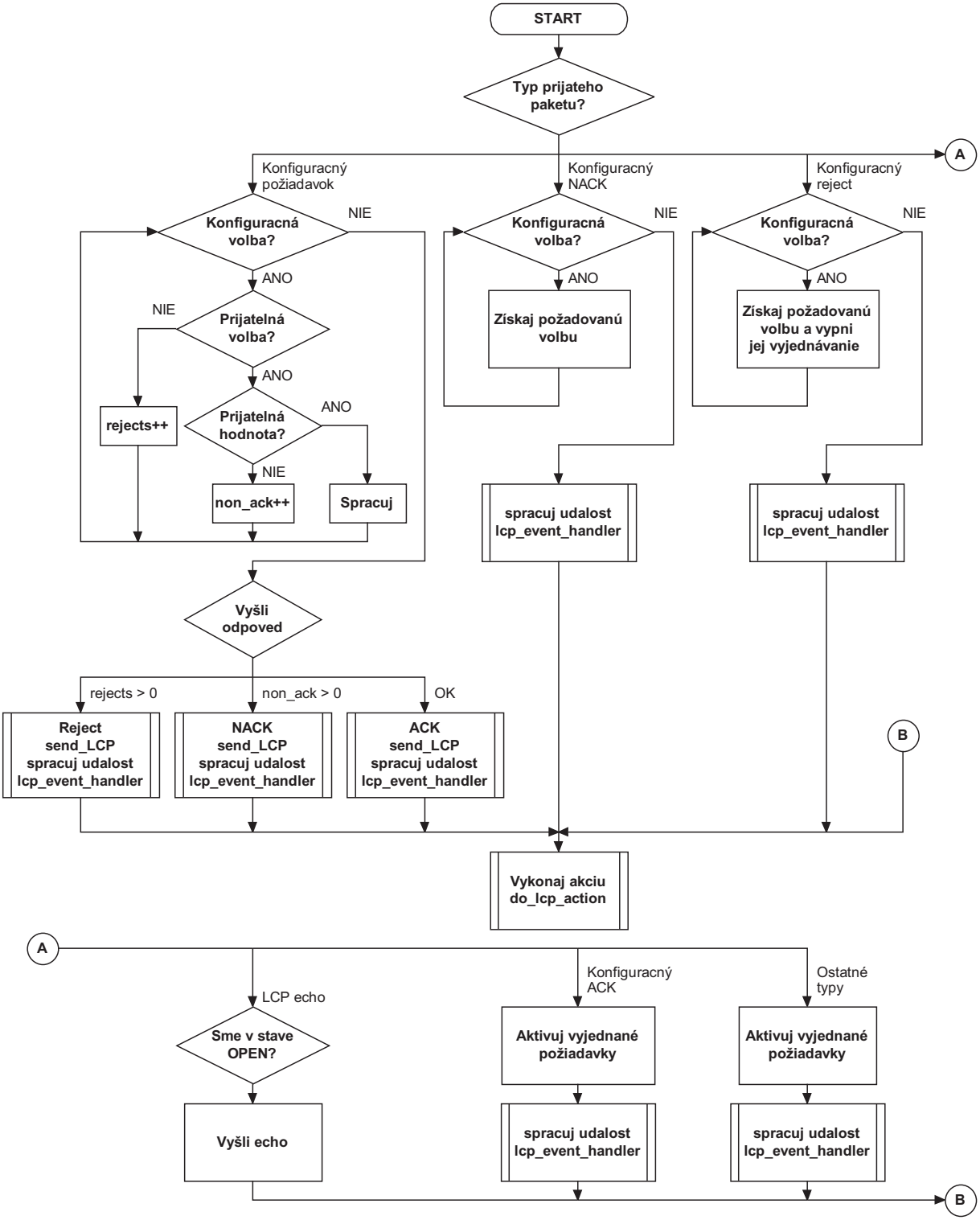
Stavový automat funguje spôsobom popísaným v kapitole PPP. Po spracovaní prichádzajúcich paketov a na základe posunu v stavovom automate, vzniká udalosť (event). V prípade vzniku udalosti, sa volá funkcia obsluhy udalostí pre daný typ protokolu, ktorého paket udalosť spôsobil. Napríklad pre LCP protokol je to

```
void lcp_event_handler(BYTE event)
```

Použité premenné v `lcp_event_handler`:

Názov	**Typ**	**Význam**
`event`	`BYTE`	Udalosť ktorá nastala
`lcp_state`	`BYTE`	Pôvodný stav
`lcp_action`	`BYTE`	Akcia ktorá sa má vykonať

`new_state`
 	
`state`
	`BYTE`	Pomocné premenné, ktoré slúžia na uloženie starého a získanie nového stavu



Obr. 6.6: Vývojový diagram funkcie lcp_rx_handler.

Funkcie obsluhujúce udalosti nám vrátia nový stav do ktorého stavový automat prešiel a príslušné akcie, ktoré musia byt vykonané. Podobne ako pre obsluhu udalosti, existujú aj funkcie na obsluhu akcii. Opäť pre každý typ protokolu existuje jedna funkcia obsluhy akcii. Napríklad pre LCP protokol je to

```
void do_lcp_actions(void)
```

Použité premenné v `do_lcp_actions`:

Názov	**Typ**	**Význam**
`ppp_pcol`	`WORD`	Definovanie použitého protokolu. V tomto prípade je nastavený na LCP pretože táto funkcia sa volá jedine vtedy ak nastala udalosť v LCP stavovom automate
`lcp_action`	`BYTE`	Akcia ktorá sa má vykonať
`req_conf`	`BYTE`	Obsahuje nenulovú hodnotu ak chceme vyjednať s našej strany ľubovolné konfiguračné možnosti
`accm, magic, pfc ap, mru, qsp`	`BYTE`	Premenné určujúce či danú možnosť chceme vyjednávať

6.3 IP stack

IP paket je v našom prípade zapúzdrený jedine v PPP pakete, čím nám odpadá nutnosť detekcie protokolu linkovej vrstvy. Z tohto dôvodu, som ale nezačlenil PPP priamo do štruktúr IP, pretože by to v ďalšom použití IP stacku stažovalo prechod na iné linkové protokoli. Teda IP paket nemá s pohľadu štruktúr jeho popisujúcich nadradený protokol, podobne ako PPP(u ICMP resp. UDP tomu už tak nie je). Jeho hlavička definovaná v kapitole TCP/IP je realizovaná štruktúrou, ktorej položky odpovedajú jednotlivým polom jak veľkosťou aj umiestnením

```
typedef struct
{
    BYTE  vhl,              //Verzia a dĺžka hlavicky
          service;          //Kvalita služieb IP
    WORD  len,              //Celková dĺžka IP paketu
          ident,            //Identifikátor
          frags;            //Flagy & fragmentacia
```

```
    BYTE  ttl,              //Čas žitia paketu
          pcol;             //Použitý protokol vyššej vrstvy
    WORD  check;            //Kontrolní súčet
    LWORD sip,              //IP adresa odosieľatela
          dip;              //IP adresa príjemcu
#if BIGHEAD
          LWORD option;     //Dodatočné možnosti
#endif
} IPHDR;
```

Celý IP paket tzn. hlavička i dáta sú definované tiež ako štruktúra, ktorá obsahuje ako svoje položky štruktúru IP hlavičky a dáta sú reprezentované statickým polom znakov.

```
typedef struct
{
    IPHDR  i;               //IP hlavička
    BYTE   data[MAXIP];     //Dáta
} IPKT;
```

Práca s paketom potom pozostáva z vytvorenia pointera na štruktúru `IPKT` a odovzdaním mu adresy prvku buffera, ktorý odpovedá jej fyzickému umiestneniu v PPP paketu. Tým že štruktúra neobsahuje voľné pamäťové miesta vďaka zarovnaniu typov na párne adresy prekladačom, sa nám prekryje štruktúra IPKT s paketom a jednotlivé polia paketu budú prístupné priamo pomocou položiek tejto štruktúry.

Jednotlivé operácie, ktoré musí IP stack vykonávať sú

- Vytvorenie IP paketu
- Kontrola, či prijatý paket je protokolu IP
- Fragmentácia
- Obsluha ICMP protokolu

6.3.1 Vytvorenie IP paketu

Na vytvorenie IP paketu slúži funkcia:

```
WORD make_ip(PPPKT *ppp, NODE *srcep, NODE *destp, BYTE pcol,
WORD dlen)
```

Táto funkcia sa volá pri konštrukcii IP paketu, teda jej volanie následuje vytvoreniu protokolu vyššej vrstvy (napr. UDP). Po vytvorení IP paketu, teda potom nasleduje vytvorenie paketu PPP a potom jeho fyzické vyslanie po sériovej linke. Treba si uvedomiť, že pri vytváraní paketov postupujeme v modelu ISO/OSI obr. 2.2 zhora nadol.

Vstupné parametre a použité premenné v `make_ip`:

Názov	**Typ**	**Význam**
`ppp`	`PPPKT *`	Určuje umiestnenie PPP paketu, ktorý nesie tento konkretny IP paket. Používa sa na získanie pointera na IPKT, pretože začiatok pola dáta s PPPKT odpovedá začiatku uloženia IP paketu
`srcep`	`NODE *`	Informácie o odosieľatelovy
`destp`	`NODE *`	Informácie o príjemcovy
`pcol`	`BYTE`	Odpovedá identifikátoru IP protokolu používaného v nižšom protokolu teda v našom prípade PPP
`dlen`	`WORD`	Veľkosť dát, ktoré IP protokol má prenášať (tzn. v podstate dlžku vyššieho protokolu napr. ICMP, UDP)
return	`WORD`	Návratová hodnota udáva dĺžku IP paketu

Štruktúra NODE udáva všetky informácie, ktoré sú potrebné na komunikáciu v prostredníctvom TCP/IP. Vypadá následovne:

```
typedef struct
{
    WORD  dtype;              //Typ drivera linkovej vrstvy (PPP)
    LWORD ip;                 //IP adresa
    LWORD mask;               //Maska
    LWORD gate;               //Adresa brány
    WORD  port;               //Port
} NODE;
```

6.3.2 Kontrola prijatého paketu

Na kontrolu prijatého paketu či je protokolu IP slúži funkcia:

```
WORD is_ip(PPPKT *ppp, int len)
```

Táto funkcia sa volá po prijme PPP paketu. Či sa jedná o IP protokol to zistíme už v funkcii `get_net` ale táto funkcia nám dá podrobné informácie o jednotlivých parametroch a skontroluje správnosť prijatého paketu. Na základe výsledku tejto funkcie, potom môžeme ďalej postupovať modelom ISO/OSI obr. 2.2 v tomto prípade zdola nahor.

Vstupné parametre a použité premenné v `is_ip`:

Názov	**Typ**	**Význam**
`ppp`	`PPPKT *`	Určuje umiestnenie PPP paketu, ktorý nesie tento konkrétny IP paket. Používa sa na získanie pointera na IPKT, pretože začiatok pola dáta s PPPKT odpovedá začiatku uloženia IP paketu
`len`	`WORD`	Dĺžka paketu
return	`WORD`	Návratová hodnota udáva dĺžku dát IP paketu

6.3.3 Fragmentácia

Fragmentácia slúži v prípade vyslania resp. prijatia IP paketov dlhších ako je protokol linkovej vrstvy schopný prenášať. V tomto prípade, je daný IP paket rozdelení do viacerých paketov, ktoré sú už schopné prenosu linkovým protokolom. V našom stacku nie je táto možnosť implementovaná v dôsledku úplnej kontroly dĺžky paketu na oboch stranách. Teda dĺžku paketu pri vysielaní si kontrolujeme programovo, jak v riadiacej jednotke tak i dispečerskom centre. V prípade požiadavku prístupu na server resp. aplikáciu, ktorá nie je plne ovládaná nami by sa táto funkcia musela implementovať.

6.3.4 ICMP protokol

ICMP protokol je doplnok IP protokolu a dáva jednotlivým uzlom siete diagnostické možnosti. Najpoužívanejšou služba je PING ktorá kontroluje dostupnosť stanice na sieti. ICMP je podrobne popísaný v kapitole TCP/IP. Spôsob jeho implementácie je zhodný s IP protokolom len s odpovedajúcimi vlastnými štruktúrami, ktoré ho popisujú. Jeho hlavička a celý paket sú realizované následovne

```
typedef struct
{
```

```
    BYTE  type,             //Typ správy
          code;             //Kód správy
    WORD  check,            //Kontrolní súčet
          ident,            //Identifikátor
          seq;              //Sekvenčné číslo
} ICMPHDR;

typedef struct
{
    IPHDR    i;             //IP hlavička
    ICMPHDR  c;             //ICMP hlavička
    BYTE     data[MAXICMP]; //Data
} ICMPKT;
```

Protokolol ICMP je braný ako protokol rovnakej vrstvy ako IP, ale v prípade prenosu musí byť zapúzdrený v IP protokolu. Preto s pohľadu IP je to protokol vyššej vrstvy. Tým obsahuje jeho štruktúra realizácie paketu aj IP hlavičku. To nám umožňuje rýchly prístup priamo k ICMP paketu, pretože zostáva rovnaký ako pre IP. Funkcie na prácu s ICMP sú:

```
void make_icmp(PPPKT *ppp, NODE *srcep, NODE *destp, BYTE type,
BYTE code, WORD dlen)
void is_icmp(IPKT *ip, int len)
void swap_icmp(ICMPKT *icmp)
```

Vstupné parametre a použité premenné v `make_icmp`:

Názov	Typ	Význam
ppp	PPPKT *	Určuje umiestnenie PPP paketu, ktorý nesie tento konkrétny ICMP paket. Používa sa na získanie pointera na ICMPKT, pretože začiatok pola data s PPPKT odpovedá začiatku uloženia ICMP paketu
srcep	NODE *	Informácie o odosieľatelovy
destp	NODE *	Informácie o príjemcovy
type	BYTE	Odpovedá typu ICMP paketu, teda určuje typ služby
code	BYTE	Odpovedá kódu ICMP paketu, teda bližšie špecifikuje typ služby
dlen	WORD	Veľkosť dát ktoré ICMP protokol má prenášať (tzn. v podstate dĺžku vyššieho protokolu napr. ICMP, UDP)
return	WORD	Návratová hodnota udáva dĺžku ICMP paketu

Vstupné parametre a použité premenné v `is_icmp`:

Názov	Typ	Význam
ip	IPKT *	Určuje umiestnenie IP paketu, ktorý nesie tento konkrétny ICMP paket. Používa sa na získanie pointera na ICMPKT, pretože začiatok pola data s IPKT odpovedá začiatku uloženia ICMP paketu
return	WORD	Návratová hodnota udáva dĺžku dát ICMP paketu

Vstupné parametre a použité premenné v `swap_icmp`:

Názov	Typ	Význam
icmp	ICMPKT *	Určuje umiestnenie ICMP paketu

6.4 UDP stack

UDP sa radí medzi protokoly transportných služieb. Jeho úloha je prenos uživateľských dát. Býva zapúzdrený v protokolu IP a preto obsahuje jeho štruktúra realizácie paketu aj IP hlavičku. Práca s UDP je podobná ako s IP a ICMP, teda jedná sa o rovnaké funkcie prispôsobené UDP.

V dôsledku ušetrenia nárokov na procesor, som realizoval dve možnosti UDP paketu. Navzájom sa líšia získavaním dát. Paket typu PPPKT má dáta realizované podobne ako

PPP, IP, ICMP a to pomocou statického pola. Tým pádom, takýto paket sa musí počas vysielania celý nachádzať v transmit bufferu. Tento spôsob by pri prenose väčšieho poctu dát znamenal značné zaťaženie procesora a preto paket UDPKT2 ma dáta realizované ako pointer na znak. V tomto prípade, sa nachádza v transmit bufferi hlavičky použitých protokolov a UDP dáta sa získavajú priamo z ľubovolného miesta pamäti. Tým pri odosielaní nedochádza ku kopírovaniu do transmit buffera ale priamo sa vysielajú dáta z pamäte kde sa nachádzajú.

```
typedef struct
{
    WORD  sport,            //Zdrojový port
          dport,            //Cieľový  port
          len,              //DÍžka dát a hlavičky
          check;            //Kontrolný súčet
} UDPHDR;

typedef struct
{
    IPHDR   i;              //IP hlavička
    UDPHDR  u;              //UDP hlavička
    BYTE    data[MAXUDP];   //Dáta
} UDPKT;

typedef struct
{
    IPHDR   i;              //IP hlavička
    UDPHDR  u;              //UDP hlavička
    BYTE    *data; //Dáta
} UDPKT2;
```

Funkcie na prácu s UDP sú:

```
void make_udp(PPPKT *ppp, NODE *srcep, NODE *destp, WORD dlen)
void is_udp(IPKT *ip, int len)
void swap_udp(UDPKT *udp)
```

Vstupné parametre a použité premenné v `make_udp`:

Názov	Typ	Význam
ppp	PPPKT *	Určuje umiestnenie PPP paketu, ktorý nesie tento konkrétny ICMP paket. Používa sa na získanie pointera na ICMPKT, pretože začiatok pola data s PPPKT odpovedá začiatku uloženia ICMP paketu
srcep	NODE *	Informácie o odosielatelovy
destp	NODE *	Informácie o príjemcovy
dlen	WORD	Veľkosť dát, ktoré UDP protokol má prenášať
return	WORD	Návratová hodnota udáva dĺžku ICMP paketu

Vstupné parametre a použité premenné v `is_udp`:

Názov	Typ	Význam
ip	IPKT *	Určuje umiestnenie IP paketu ktorý nesie tento konkrétny UDP paket. Používa sa na získanie pointera na UDPKT/UDPKT2, pretože začiatok pola data s IPKT odpovedá začiatku uloženia UDP paketu
return	WORD	Návratová hodnota udáva dĺžku dát UDP paketu

Vstupné parametre a použité premenné v `swap_udp`:

Názov	Typ	Význam
udp	UDPKT *	Určuje umiestnenie UDP paketu

Všetky uvedené funkcie vo forme zdrojových kódov sa nachádzajú na priloženom CD. Pri spôsobu implementácii PPP a UDP/IP stacku som využil metódy uvedené v [15]. Pre bližšie pochoponie implementácie stackov v embedded systémoch je táto knižka veľmi užitočná.

Kapitola 7

Záver

Cieľom diplomovej práce bolo vytvorenie systému na diaľkové sledovanie vozu. Základné požiadavky na náš systém boli zber dat z internej zbernice CAN automobilu škoda Octavia a data o polohe zo systemu GPS. Ďalej sme museli použiť hardware realizovaný na katedre riadenia. To viedlo k použitiu vývojovej dosky s mikrokontrolerom Motorola HC12 ako riadiacej jednotky s možnostou pripojenia sa na CAN zbernicu a vývojovej dosky z modulom Siemens XT55 ako prenosovej jednotky umožnujúcej príjem signálu GPS a prenos dat pomocou GSM/GPRS.

Mojou úlohou bolo vytvorenie nástrojov potrebných na komunikáciu, teda prenos dát o polohe získaných prostredníctvom GPS a dát o stave automobilu z internej zbernice CAN, na internet a vytvorenie riadiaceho algoritmu systému.

Na získanie dát zo zbernice CAN sme použili už existujúce knižnice `can.h`, `msCan.h` a `com.h`. Mojou úlohou bolo vytvorenie Point-to-Point stacku na riadiacej jednotke. Pri jeho vytváraní som kládol dôraz na jeho univerzálnosť a možnosť jednoduchej zmeny vyjednávaných parametrov pre možnosť jeho použitia vo výuke. Ďalej som sa snažil jeho plnu implementáciu podľa RFC noriem. Pre prenos dát som realizoval UDP/IP stack taktiež na riadiacej jednotke. Jeho návrh spočíva v úplnej nezávislosti na PPP stacku s dôvodu možného ďalšieho použitia.

Pre spracovanie a ukladanie dát v riadiacej jednotke som vytvoril riadiaci algoritmus. Jeho úlohou je zabezpečenie prijmu dát, ich efektívne spracovanie s pohladu veľkosti, diagnostika a ošetrenie chybových stavov. Ďalej vysielanie dat v podobe UDP paketou musí byť realizované s pohladu čo najmänšieho zaťaženia pri ich kopírovaní v pamäti. To sme vyriešili dynamickým smerovaním s vysielacieho buffera do datového buffera. Tým nedochádza ku kopírovaniu dát do vysielacieho buffera pred samotným vyslaním, ale berú sa priamo s datového buffera.

Rozšírenie a zdokonalenie systému dialkového sledovania vozu GSM technologiami by v budúcnosti mohlo viesť na ďalšie práce na katedre riadiace techniky. Možným zdokonalením by mohlo byť:

- Implementácia operačného systému realneho času OSEK v riadiacej jednotke
- Implementácia TCP protokolu a vyšších protokolov HTML, FTP
- Realizácia dispečerského centra zo zobrazením polohy a ukladaním dát do databaze.

Literatúra

[1] Musil, M. *Operační systém reálného času pro automobily :* diplomová práce. Praha : ČVUT-České vysoké učení technické v Praze, Fakulta elektrotechnická, 2003.

[2] Černý, M. *Komunikace CAN v automobilu :* diplomová práce. Praha : ČVUT-České vysoké učení technické v Praze, Fakulta elektrotechnická, 2005.

[3] Krakora, M. *Operačný systém OSEK a zbernica CAN :* diplomová práce. Praha : ČVUT-České vysoké učení technické v Praze, Fakulta elektrotechnická, 2003.

[4] Simpson, W. *The Point-to-Point Protocol.* RFC 1331, 1992.

[5] Perkins, D., Hobby, R. *Point-to-Point Protocol (PPP)initial configuration options.* RFC 1172, 1992.

[6] Simpson, W. *The Point-to-Point Protocol (PPP).* RFC 1548, 1993.

[7] Lloyd, B., Simpson, W. *PPP Authentication Protocols.* RFC 1334, 1992.

[8] McGregor, G. *The PPP Internet Protocol Control Protocol.* RFC 1332, 1992.

[9] Siemens, *XT55 Hardware Interface Desciption, Version 02.*

[10] Siemens, *XT55 AT Command Set, Version 1.00.*

[11] Siemens, *XT55 GPRS Startup User's Guide.*

[12] Siemens, *Upgrading XT55 firmware.*

[13] Siemens, *XT55 TCP/IP Software User's Guide.*

[14] Siemens, *XT55 AVL Software Instruction User's Guide.*

[15] Bentham, J. *TCP/IP Lean - Web servers for embedded systems (Second Edition).* Lawrence, Kansas 66046 : CMP Books 2002.

[16] Dostálek, L., Kabelová, A. *Velký pruvodce protokoly TCP/IP a systémem DNS.* Praha : Computer press 2000.

Dodatok A

Konfiguračné možnosti LCP

Maximum prijatých zložiek (MRU) Táto konfiguračná voľba sa používa v prípade ak chceme informovať protiľahlú stanicu že môžeme prijímať väčšie pakety alebo naopak aby posielala menšie pakety ako je prednastavená hodnota. Táto hodnota (MTU) je 1500 bytov. Ak sú požadované menšie pakety naša realizácia prijímania musí byt schopná prijať aj pakety pôvodnej dĺžky v prípade straty synchronizácie spojenia. Pole pre túto možnosť vypadá nasledovné

Typ voľby	Dĺžka	Maximum prijatých zložiek

Typ voľby = 1
Dĺžka = 4
Maximum prijatých zložiek Pole je veľkosti 2 byty a obsahuje číslo, ktoré určuje maximálny počet bytov.

Asynchrónna kontrolná mapa (ACCM) Konfiguračná možnosť umožňuje vyjednať aké mapovanie kontrolných znakov sa bude používať na asynchrónnych linkách. Pôvodne sú nastavené PPP mapy všetkých kontrolných znakov na dvojznakovú dlžku použitím escape sekvencie. PPP realizácia môže použiť túto konfiguračnú možnosť pre informovanie vzdialenej stanice, ktorý kontrolný znak musí zostať mapovaný a ktorý nepotrebuje byť mapovaný, keď ho vzdialená stanica vyšle. Vzdialená stanica ale môže naďalej vysielať tieto kontrolné znaky v mapovanom formáte ak je v tomto smeru obmedzená.

Typ voľby	Dĺžka	Asynchrónna kontrolná mapa

Typ voľby = 2
Dĺžka = 6

Asynchrónna kontrolná mapa Pole je veľkosti 4 byty definuje nové mapovanie kontrolných znakov.

Authentifikačný protokol (AP) Konfigurácia obsahuje metódu na dorozumenie sa ktorý autentifikačný protokol sa bude používať napr. PAP, CHAP. Ako prvý sa snažíme vyjednať protokol, ktorý nám najviac vyhovuje a v prípade prijatia paketu *Configure-Nak* potom vyjednávame ďalší protokol. Tá stanica ktorá vyšle *Configure-Request* vyžaduje od opačnej stanice autentifikáciu. Tá pošle odpoveď podľa toho či jej protokol vyhovuje - *Configure-Ack* alebo ak nevyhovuje *Configure-Nak*.

Typ voľby	Dĺžka	Autentifikačný protokol	Dáta

Typ voľby $= 3$
Dĺžka ≥ 4
Maximum prijatých zložiek Pole je veľkosti 2 byty a obsahuje kód autentifikačného protokolu

- C0 23 - Password Authentication Protocol (PAP)
- C2 23 - Challenge Handshake Authentication Protocol (CHAP)

Dáta obsahujú dodatočné informácie k danému protokolu

Protokol kvality (QP) Pri niektorých spojeniach môže byt žiaduce presné určenie kedy a ako často ma linka prestať prenášať dáta. Tento proces sa nazýva monitorovanie kvality. Konfigurácia obsahuje metódu na vyjednanie presne, ktorý protokol pre kontrolu kvality sa bude používať. V pôvodnom nastavení je monitorovanie kvality vypnuté. Vyslanie *Configure-Request* stanicou znamená že očakáva informáciu monitorovania kvality od druhej stanici.

Typ voľby	Dĺžka	Protokol kvality	Dáta

Typ voľby $= 4$
Dĺžka ≥ 4
Maximum prijatých zložiek Pole je veľké 2 byty a obsahuje požadovaný protokol monitorujúci kvalitu spojenia.

- C0 25 - Link Quality Report

Dáta obsahujú dodatočné informácie k danému protokolu

Magické číslo (Magic-Number) Táto konfiguračná možnosť poskytuje metódy na detekciu uzatvorených smyčiek a iných anomálií na dátovej linke. Býva vyžadovaná niektorými inými konfiguračnými možnosťami ako je napríklad protokol kvality. V pôvodnom nastavení nie je magické číslo požadované. Pred tým než táto konfiguračná možnosť je požadovaná musí naša aplikácia určiť magické číslo. Je doporučené aby bolo vybrané čo možno najväčším náhodným spôsobom aby bola garantovaná veľká pravdepodobnosť že číslo bude jedinečné. Dobrou cestou k získaniu jedinečných čísel je vybrať si jedinečný zdroj. Doporučené zdroje sú napríklad sériové číslo, iné sieťové hardwarové adresy, čas a deň a podobne. Dobrou náhodnou postupnosťou je meranie počtu stlačených kláves alebo príchodzích paketov z inej siete za určitý čas alebo odozva serveru. Keď je *Configure-Request* prijatý s magickým číslom tak je toto číslo porovnané s predchádzajúcim magickým číslom prijatého *Configure-Requestu*. Ak sú tieto dve čísla rôzne potom sa v spojení nevyskytuje uzatvorená smýčka a magické číslo má byt potvrdené. Ak sú rovnaké potom je tu možnosť ale nie istota že v spojení je uzatvorená smyčka. Kvôli rozhodnutiu o tom či existuje uzatvorená smyčka musí byt vyslané *Configure-Nak* s iným magickým číslom. Nový *Configure-Request* by nemal byt vyslaný dotedy pokým nebude obnovený štandartný prevoz. Príjem *Configure-Nak* s rozdielnym magickým číslom ako bolo posledne vyslané dokazuje že v spojení sa neobjavuje uzatvorená smyčka a pri zhode sa pravdepodobnosť smyčky zvyšuje a ďalšie rozdielne magické číslo musí byt vybrané. Ak je spojenie v uzatvorenej smyčke bude sa toto vysielanie *Configure-Requestu* a *Configure-Nak* opakovať dookola.

Typ voľby	Dĺžka	Magické číslo	Cont

Typ voľby $= 5$
Dĺžka ≥ 6
Magické číslo Pole je veľkosti 4 byty a obsahuje jedinečné číslo pre daný koniec spojenia.

Kompresie dátového pola (PFC) Konfiguračná možnosť, ktorá poskytuje metódu na vyjednanie kompresie PPP protokolu. Pôvodne musí každý paket obsahovať polia veľkosti 2 byty napríklad číslo `1 = 7D 21h`, číslo `2 = 7D 22` a podobne. Niektoré hodnoty môžu byt skomprimované do 1 bytu ktorý je ľahko rozpoznatelný od komprimácie 2

bytov (číslo `1 = 1`). Paket nie je nikdy komprimovaný ak sa jedná o LCP. To zaručuje jednoznačne rozlíšenie LCP paketou.

Typ voľby	Dĺžka

Typ voľby = 7
Dĺžka = 2

Kompresia adries a kontrolných polí (ACFC) Konfiguračná možnosť, ktorá poskytuje metódu na vyjednanie kompresie adries a kontrolných polí. Pôvodne nastavená je kompresia odpovedajúca kompresii protokolu. Pretože väčšinou tieto polia majú konštantný obsah sú ľahko skomprimovatelné. Túto konfiguračnú možnosť stanica posiela protiľahlej stanici ako informáciu o tom že je schopná prijímať skomprimované adresy a kontrolné polia.

Typ voľby	Dĺžka

Typ voľby = 8
Dĺžka = 2

Dodatok B

Obsah CD

Priložené CD má nasledujúcu adresárovú štruktúru:

- **DP Latex** - Zdrojový kód diplomovej práce v publikačnom systéme LaTeX
- **PPP** - Obsahuje knižnice potrebné na Point To Point stack tzn. ppp.h, p_drv.h a netutil2.h. Ďalej obsahuje literatúru stahujúcu sa k PPP.
- **UDP/IP** - Obsahuje knižnice potrebné na UDP/IP stack tzn. ip2.h a udp.h. Ďalej obsahuje literatúru stahujúcu sa k UDP/IP protokolom.
- **Control** - Obsahuje riadiaci algoritmus a všetky súbory nutné k činnosti riadiacej jednotky a získavania dat zo zbernice CAN a GPS (aj tie ktoré boli vytvorené v rámci iných diplomových prác).
- **Siemens XT55** - Obsahuje datasheety k modulu prenosovej jednotky.
- **Motorola HC12** - Obsahuje datasheety k modulu riadiacej jednotky.