VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SIMULÁTOR PRO FINANČNÍ PROTOKOLY

FINANCIAL PROTOCOL SIMULATOR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MARTIN VYMLÁTIL**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR VESELÝ, Ph.D.**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Vymlátil Martin, Bc.**
Obor: Počítačové sítě a komunikace
Téma: **Simulátor pro finanční protokoly**
Financial Protocol Simulator
Kategorie: Počítačové sítě

Pokyny:

1. Prostudujte problematiku používaných komunikačních protokolů používaných ve finanční sféře pro autorizaci finančních transakcí provedených platební kartou.
2. Analyzujte strukturu a principy vytváření zpráv základních autorizačních protokolů ACI SPDH a ISO 8583.
3. Navrhněte serverovou aplikaci, která bude sloužit jako simulátor zpracovávající autorizační požadavky a bude vytvářet relevantní odpovědi, u kterých bude možné nastavit parametry.
4. Podle doporučení vedoucího implementujte serverovou aplikaci autorizující požadavky protokolů ACI SPDH a ISO 8583.
5. Proveďte ověření funkčnosti vytvořené aplikace vůči reálnému platebnímu terminálu. Analyzujte výsledky a diskutujte další možné rozšíření aplikace.

Literatura:

- ISO8583-1:2003, ISO/TC 68/SC 7, ICS 35.240.15, https://www.iso.org/obp/ui/#iso:std:iso:8583:-1:ed-1:v1:en

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Veselý Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Konzultant: Bělín Jan, Ing., FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce se zabývá vytvořením simulátoru pro finanční protokoly ISO8583 a SPDH. Na základě analýzy zmíněných protokolů je navržen a implementován simulátor v jazyce C++. Simulátor je testován na předem připravených zprávách, ale také vůči skutečnému platebnímu terminálu. Na závěr této práce jsou diskutována možná rozšíření.

Abstract

This paper deals with the creation of a simulator for financial protocols ISO8583 and SPDH. The simulator was designed and implemented in C++ language based on the analysis of the financial protocols. The simulator was tested on pre-prepared authorization messages and was also tested against the real POS terminal. Possible extensions for the simulator were discussed at the end of this thesis.

Klíčová slova

Finanční protokol, simulátor, ISO8583, SPDH.

Keywords

Financial Protocol, Simulator, ISO8583, SPDH.

Citace

VYMLÁTIL, Martin. *Simulátor pro finanční protokoly*. Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Veselý, Ph.D.

Simulátor pro finanční protokoly

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Veselého, PhD. Další informace mi poskytl konzultant pan Ing. Jan Bělín. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Vymlátil
22. května 2017

Poděkování

Chci tímto poděkovat vedoucímu práce Ing.Vladimíru Veselému, PhD. za dohled a věcné připomínky ke struktuře a obsahu práce. Dále chci poděkovat Ing. Janu Bělínovi za odborný dohled a cenné rady, bez nichž by tato práce vznikala velice těžce. V neposlední řadě patří mé poděkování rodině a přítelkyni Markétě za neustálou podporu.

Jak jsem slíbil, tak v rámci poděkování dále uvádím oblíbený recept, vhodný při posezení s přáteli u dobrého piva. Jedná se o plněné zapečené klobásy. Na 4 porce budeme potřebovat 250 g krájených žampionů, 1 větší cibuli, 4 rajčata, 1 větší papriku, 1 vejce, 1/8 l šlehačky, 150 g strouhaného ementálu, 400 g klobás (ideálně ostravských klobás), trochu másla, lžičku kari, sůl a pepř. Na jemno nakrájenou cibuli a papriku smícháme s vejcem, smetanou a polovinou sýra. Přidáme kari, osolíme, opepříme a vmícháme nakrájené žampiony. Vymažeme pekáč máslem, klobásy rozřízneme po délce a naplníme. Obložíme plátky rajčat a posypeme zbytkem sýra. Pečeme v troubě přibližně 20 minut na 200 °C. Podáváme s čerstvým chlebem.

Obsah

Kapitola 1

Úvod

Bezhotovostní platební styk za pomoci platební karty a platebního terminálu je v poslední době velice oblíbený. Pro vydavatele aplikací pro platební terminály je velmi dobré mít k dispozici nástroj, který bude schopen alespoň částečně simulovat chování autorizačního centra, jenž komunikuje s terminály pomocí finančních protokolů. Ty slouží pro přenos dat potřebných k uskutečnění transakce. V současné době patří mezi velmi rozšířené protokoly ISO8583[8] a SPDH[1]. Dosud existující nástroje pro simulaci chování finančních protokolů poskytují pouze malou škálu funkčností použitelných pro testování a často nesplňují všechna kritéria vyžadovaná během testování. Tyto nástroje obsahují implementace základních protokolů nebo jen jejich částí, neumí například reagovat na nevalidní zprávu a jsou velmi těžce rozšiřitelné. Výhodou vlastního simulátoru může tedy být:

- modularita, doplnění dalších specifických protokolů bez nutnosti komunikovat přímo na samotnou autorizaci a bez nutnosti vytváření další nové aplikace,
- vytvoření nejrůznější scénářů zahrnujících situace, kdy je potřeba například mít informace o samotném účtu držitele karty nebo o historii plateb, které již proběhly,
- snadnější nastavování různých návratových, a to především chybových, kódu při testování.

Cílem této diplomové práce je seznámit se s protokoly ISO8583 a SPDH, na základě získaných znalostí pak vytvořit simulátor, který bude obsahovat implementaci těchto základních finančních protokolů, a který bude do budoucna rozšiřitelný o další funkčnosti a protokoly podle požadavků společnosti Sonet, s.r.o.[1], pod jejíž záštitou tato práce vzniká. Firma Sonet, s.r.o., je tuzemská společnost zabývající se poskytováním komplexních služeb v oblasti platebních technologií. Mezi její hlavní činnosti pak patří především vývoj softwaru do platebních terminálů, dále vývoj informačního systému pro acquiring, manažerského systému a další.

V rámci této práce si nejdříve nadefinujeme některé základní pojmy z oblasti financí a bankovnictví. Nejdůležitější kapitolou 2 celé práce pak bude analýza a bližší představení finančních protokolů ISO8583 a SPDH. V následující kapitole 3 se budeme zabývat návrhem simulátoru a popisem všech potřebných součástí. Samotná implementace bude předmětem kapitoly 4, mimo jiné zde budou uvedena i možná rozšíření a další pokračování. Po popisu implementace bude následovat kapitola 5 zabývající se testováním a testovací metodikou. Na závěr budou shrnuty zásadní body práce a její přínos.

[1] Sonet, s.r.o. Dostupné z: http://www.sonet.cz/

Základní pojmy - slovník pojmů

V dalších kapitolách této práce se může vyskytovat několik pojmů z oblasti financí a bankovnictví. Některé z nich mohou být poměrně běžné, jiné však již mohou být méně známé a pro další pochopení práce je nutné tyto pojmy uvést a definovat. To bude předmětem této sekce, přičemž byly použity následující zdroje [11, 21, 10, 20, 12, 3].

- **Obecné pojmy:**
  - **Acquirer** - představuje zúčtovací banku, která uzavírá smluvní vztahy s obchodními společnostmi, zpracovává transakce provedené platebními kartami, a to buď přímo nebo prostřednictvím třetí strany, dále zajišťuje clearing a zúčtování.
  - **Autorizace (Authorization)** - je proces, při kterém je vyžádán souhlas vydavatele karty s platbou nebo výplatou hotovosti prostřednictvím platební karty. Souhlas je vyjádřen skrze poskytnutý autorizační kód. V rámci autorizace je také kontrolována platnost platební karty nebo skutečnost, zda není karta blokována. U transakcí provedených čipovou kartou může schválení/zamítnutí provést čip v rámci limitů nastavených vydavatelem karty.
  - **Autorizační kód (Response code)** - představuje alfanumerický kód vyjadřující výsledek transakce. V případě kladného výsledku vyjadřuje souhlas vydavatele karty s provedením transakce. Dále slouží také jako důkaz o poskytnutí souhlasu vydavatele s uskutečněním transakce.
  - **Bezkontaktní transakce (Contactless)** - platba provedená za pomoci integrovaného bezkontaktního čipu na kartě nebo na jiném zařízení. Takový způsob platby je zpravidla rychlejší, jelikož standardně v České republice při částce pod 500 Kč není třeba zadávat PIN a není vyžadováno ověření podpisem.
  - **Clearing** - je proces výměny údajů o finanční transakci mezi acquirerem a vydavatelem karty pro účely zaúčtování transakce na účet držitele karty a rekonciliace pozice člena asociace k vypořádání.
  - **Číslo karty (PAN - Primary Account Number)** - číslo identifikující vydavatele karty a konkrétní účet držitele karty. Skládá se z identifikátoru odvětví, vydavatele, konkrétního účtu a kontrolní číslice.
  - **EMV/ICC (Integrated Circuit Card)** - označení pro platební kartu vybavenou čipem s programovatelným mikroprocesorem a pamětí.
  - **Floor limit** - je částka dohodnutá mezi obchodníkem a zúčtovací bankou, omezující výši jedné transakce, pro kterou není nezbytné provádět online autorizaci. Pro vyšší částku již musí obchodník autorizaci provést.
  - **MAC (Message Authentication Code)** - jedná se o vypočtený podpis dat (obdobu hashe), který slouží jako ochrana finanční transakce proti náhodné nebo úmyslné změně dat (úprava dat, přidání dat nebo další zprávy).
  - **Offline autorizace** - lokální autorizace na platebním zařízení.
  - **Online autorizace** - autorizace provedená na autorizačním centru.
  - **PIN (Personal Identification Number)** - alfanumerický kód sloužící k autentizaci držitele karty.
  - **Platnost karty (Expiration Date)** - údaj vymezující období, kdy lze kartu používat. Je uveden jednak na přední straně karty (ve formátu MM/RR) a také je uložen na čipu karty nebo magnetickém proužku.

- **Response text** - doplňující text k autorizačnímu kódu (response kódu), který může obsahovat bližší popis výsledku transakce a může být například vytištěn na účtenku nebo zobrazen na displej terminálu.
- **Spropitné (Tip)** - je drobná částka přidaná k ceně za určitou službu, která představuje vyjádření spokojenosti zákazníka. Běžně se spropitné dává v restauracích, v kadeřnictví nebo například v případě využití taxi služby.
- **Transakce (Transaction)** - bezhotovostní platba za pomoci platební karty za zboží či službu anebo za výběr hotovosti kartou. Probíhá v několika fázích a to je zahájení a ověření držitele, schválení neboli autorizace platby, vyhotovení dokladu a zúčtování.
- **Vydavatel karty (Issuer)** - banka nebo finanční instituce, která vydává platební karty a má uzavřen smluvní vztah s držitelem karty.
- **Obchodník (Merchant)** - subjekt přijímající bezhotovostní platby za zboží nebo služby prostřednictvím platebních karet, který uzavírá smlouvu na přijímaní takovýchto karet s acquirerem.

- **Zprávy/transakce:**
  - **Cash back (Výběr hotovosti)** - představuje transakci, při níž je celková částka vyšší než skutečná cena zboží nebo služby. To umožňuje obchodníkovi vyplatit přebytečnou sumu zákazníkovi, který tak získá hotovost. Jedná se o výběr hotovosti u obchodníka při koupi zboží či služby.
  - **Návrat (Refund)** - připsání prostředků na účet držitele karty v případě vrácení zboží či refundaci služby.
  - **Prodej (Sale)** - nákup zboží nebo služby.
  - **Předautorizace (Pre-authorisation)** - žádost o autorizaci odhadnuté částky používaná acquirerem k zajištění finančních prostředků pro transakci, která má být dokončena později. Zahrnuje také kontrolu platnosti platební karty. Tuto transakci obvykle používají hotely, půjčovny automobilů nebo samoobslužné čerpací stanice pro rezervaci finančních prostředků.
  - **Předautorizace-dokončení (Pre-authorisation completion)** - jedná se o dokončení dřívější předautorizace, potvrzení finální částky, přičemž stržená částka by neměla být větší jako částka při předautorizaci.
  - **Storno (Void)** - představuje zrušení původně zaúčtované transakce a je provedeno manuálně.
  - **Reversal** - je druh storna, které je provedeno automaticky. Je to zpráva, která je vygenerována v případě, kdy terminál odeslal žádost o transakci do sítě, nicméně neobdržel platnou odpověď před vypršením časové limitu. Reversal je odesílán opakovaně, dokud není přijata platná odpověď. Odesílání probíhá pouze v případě online finančních transakcí.
  - **Uzávěrka (Settlement, Batch closure)** - představuje potvrzení stavu autorizovaných transakcí v daném období (dávce). Pro autorizační centrum pak znamená možnost zahájit clearing/zaúčtování.

Platební karty a terminály

Platební karta je zpravidla plastová karta obsahující informace spojující držitele karty s jeho účtem a další data potřebná k provedení platební transakce. Taková karta pak umožňuje výběr platební hotovosti nebo nákup zboží a služeb nejenom v tuzemsku, ale i v zahraničí[19]. Platební karty lze rozdělit podle několika hledisek, například podle způsobu vyúčtování na kreditní, debetní, nákupní a úvěrové, předplacené a charge karty, nebo podle způsobu provedení na embosované a elektronické. Nicméně z technického hlediska je nejzajímavější rozdělení podle použité technologie záznamu[15, 4, 5]:

- Karty s magnetickým proužkem - magnetický proužek je umístěn na zadní straně karty a obsahuje informace o kartě a jejím držiteli. Podoba a rozdělení dat na magnetickém proužku je dána normou ISO/IEC 7813 a je následující[7, 10]:
  - Track1: první stopa obsahuje 79 znaků, je určena pouze pro čtení a jsou na ní číslo účtu klienta, jeho jméno a číslo karty.
  - Track2: druhá stopa má 40 znaků, je určena také pouze pro čtení, obsahuje číslo karty a v bankovnictví je nejpoužívanější.
  - Track3: stopu 3 lze použít i k záznamu dat, nicméně není moc často používána a nemusí tedy být ani součástí samotné karty.
- Čipové karty - tyto karty jsou vybaveny čipem s programovatelným mikroprocesorem a vlastní pamětí. Čipová technologie napomáhá ke zvýšení bezpečnosti a rozšíření funkcí karty, například o různé věrnostní programy nebo v podobě elektronické peněženky [21]. Čipové karty si můžeme dále rozdělit podle toho, jakým způsobem dochází ke komunikaci mezi čipem a terminálem na [16, 5]:
  - Kontaktní - u tohoto typu čipové karty dochází k přímému kontaktu čipu s platebním terminálem. Díky tomu je zaručena vyšší bezpečnost a umožňuje nabídnout širší škálu služeb a možností.
  - Bezkontaktní - zde je čip schovaný pod laminací plastové karty a komunikace probíhá pomocí speciálního snímače.
- Hybridní karty - hybridní karta je kombinací výše uvedených a obsahuje tedy jak magnetický proužek, tak i samotný čip.

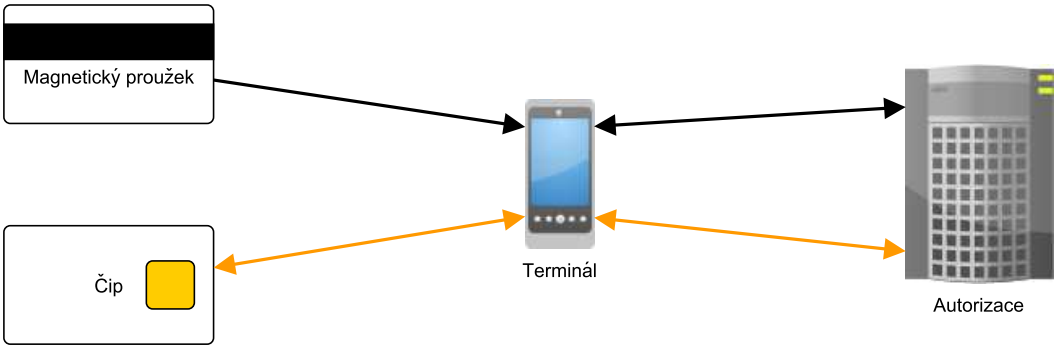
Způsob záznamu dat na platební kartě, pak určuje také způsob komunikace (nebo můžeme říct načtení informací z karty) mezi platebním terminálem a kartou. Mezi největší a nejznámější vydavatele platebních karet, kteří působí v ČR, pak patří společnosti Visa[2] a MasterCard[3] [18, 17].

Platební terminál, tak jak ho dnes známe, je elektronické zařízení umožňující provedení bezhotovostní platební transakce za pomoci platební karty nebo údajů z platební karty[10]. Z pohledu, zda aktuální transakce vyžaduje odeslání na autorizaci a schválení, rozlišujeme dva druhy transakcí [21]:

- Offline transakce - transakce nevyžaduje okamžité schválení od autorizačního centra a umožňuje lokálně zpracovat a schválit čipem zahájenou autorizační žádost pod Floor limitem. Následně pak dochází k odeslání transakce na autorizační centrum.

[2]Visa - Visa International Service Association. Dostupné z: https://https://www.visa.cz/
[3]Mastercard. Dostupné z: https://www.mastercard.cz/cs-cz.html

- Online transakce - vyžaduje vždy autorizaci od vydavatele prostřednictvím datové komunikační sítě.



Obrázek 1.1: Komunikace mezi kartou a terminálem na základě použitého vstupu.

Komunikace mezi terminálem a platebním kartou se liší podle toho, jaký vstup se použije[20, 3]:

- Magnetický proužek - v případě použití magnetického proužku dochází po protažení karty k přečtení potřebných dat pro provedení transakce. Jakmile terminál získá potřebné informace, rozhodne se, zda transakci schválí, odmítne nebo zašle k ověření na autorizaci (obrázek 1.1). V rámci transakce za pomoci magnetického proužku dochází mezi terminálem a kartou pouze k jednosměrné komunikaci, kdy terminálem čte data z karty. Toho lze zneužít, zapsaná data na proužku lze přepsat, kartu lze snadno okopírovat a karta nemá možnost zjistit, zda se jedná o skutečnou autorizaci či nikoliv.

- Čip - při použití čipové platební karty dochází již mezi terminálem a kartou ke komunikaci. Karta obsahuje mikroprocesor s pamětí, který sám provádí vlastní zpracování a zabezpečení transakce, přičemž v paměti se nachází klíče a certifikáty, které jsou přístupné pouze samotnému procesoru. Takovou kartu tedy nelze snadno okopírovat a zaručuje vyšší bezpečnost. Po získání potřebných informací terminál společně s kartou rozhodnou, zda je nutné transakci autorizovat nebo zda mohou samostatně stanovit její výsledek (obrázek 1.1).

  Průběh čipové transakce pak probíhá podle těchto bodů:

  1. Zahájení transakce - zadání částky, případně dalších údajů (spropitné, cashback, atd.),
  2. vložení karty do terminálu,
  3. výběr aplikace[4] dostupné na kartě, která je podporovaná terminálem,
  4. ověření terminálu vůči kartě,
  5. terminál poskytne kartě data transakce,

[4]Karta může obsahovat několik různých aplikací, přičemž aplikace zde představuje určitou službu, funkci, kterou karta poskytuje. Například zda je karta debetní či kreditní, zda se jedná o kartu věrnostní nebo stravenkovou, to určuje právě zvolená aplikace.

6. ověření držitele karty na základě PINu, podpisu nebo jiné metody,
7. přenos výsledku zpracování karty do terminálu a rozhodnutí o výsledku zpracování transakce, přičemž může dojít k zamítnutí transakce, k offline schválení, případně k online autorizaci.

Z hlediska komunikace mezi platebním terminál a autorizačním centrem je v praxi zvykem, že autorizace používá pouze jeden protokol, přičemž ale podporuje více karet. Použití platební karty je tedy teoreticky nezávislé na konkrétním finančním protokolu mezi terminálem a autorizací. Nicméně mohou se objevit i karty, které jsou na použitém protokolu závislé, většinou se jedné o různé věrnostní, neplatební nebo jiné karty.

Kapitola 2

Analýza finančních protokolů

V prostředí finančního světa je velké množství protokolů, sloužících k uskutečnění platební transakce, nicméně mezi základní a zároveň nejrozšířenější finanční protokoly patří právě protokol ISO8583 a protokol SPDH. Tyto protokoly si můžeme představit jako nějakou obecnou specifikaci, šablonu, která slouží pro návrh a implementaci dalších protokolů pro konkrétní platební činnost nebo služby. Téměř každé autorizační centrum pak používá vlastní modifikovanou specifikaci založenou na jednom z výše zmíněných protokolů. Oba protokoly pak pracují na principu požadavek-odpověď, tedy většinou z terminálu přichází požadavek na autorizační centrum. Takový požadavek má pro každou transakci přesně určené povinné položky, které se musí v tomto dotazu objevit. Mohou zde také být položky doporučené, nepovinné, které rozšiřují danou transakci o další informace. Například pro transakci prodej může být takový rozšiřující údaj spropitné nebo cashback. Po zpracování požadavku na autorizaci se zpět následně vrací odpověď, jejíž nejdůležitější částí je právě Response kód a Response text, které nesou data o stavu, jak dopadla daná transakce.

Při analýze samotných protokolů pro potřeby simulátoru je kladen největší důraz na formát jednotlivých zpráv a na správné určení kódu odpovědi. Kód odpovědi je v reálných podmínkách určen autorizačním centrem, které má k dispozici veškeré potřebné údaje jako jsou například stav a typ účtu držitele karty, počet, druh a výše transakcí z daného terminálu a další důležitá data. Tyto informace však simulátor k dispozici nemá, proto je nutné pro jeho základní implementaci zvolit pouze ty Response kódy, o kterých může rozhodnout jen na základě příchozího požadavku bez dalších dodatečných znalostí. Vzhledem k rozsáhlosti analyzovaných protokolů a faktu, že popisují velmi širokou škálu možných situací a transakcí, mohou být některé běžně nevyužívané položky, transakce, případně části vynechány.

2.1 Analýza protokolu ISO8583

Protokol ISO8583[8] je spravován společností International Organization for Standardization[1] a je jedním z velmi rozsáhle rodiny standardů. Hlavním cílem tohoto protokolu je definovat mezinárodně uznávaný standard pro přenos finančních zpráv. Takový jednotný standard pak umožňuje vyšší spolehlivost, flexibilitu, snížení časové odezvy, ale také nižší náklady a jeho další zdokonalování. Standard ISO8583 tedy popisuje formát zprávy a komunikaci mezi jednotlivými prvky. V současné době má tři verze ISO8583:1987, ISO8583:1993

[1] ISO. Dostupné z: http://www.iso.org/iso/home.html

a ISO8583:2003[8], které se mohou v některých částech vzájemně lišit. Nicméně stále velmi rozšířenou verzí je první z roku 1987.

Dále je popisována upravená verze protokolu ISO8583 definovaná společnosti Hypercom[2] [6].

Struktura zprávy

Zpráva se skládá ze tří základních částí (tabulka 2.1), a to je hlavička, aplikační data a CRC, přičemž hlavička společně s kontrolním součtem tvoří obálku pro aplikační data. Ta obsahují nejpodstatnější informace o prováděné transakci nebo operaci. Celá zpráva je přenášena v binární podobě[6].

Header	**Application Data**	**CRC**
7 B	var	2 B

Tabulka 2.1: Struktura protokolu ISO8583

Hlavička zprávy

Samotná hlavička (tabulka 2.2) je povinnou položkou zprávy a obsahuje informace o přenášené zprávě, ale také routovací informace. Skládá se z následujících položek[6]:

Address	**Control Byte**	**TPDU**		
		Id	Destination Address	Originator Address
		1 B	2 B	2 B
1 B	1 B	5 B		

Tabulka 2.2: Hlavička zprávy protokolu ISO8583

- Address - HDLC[3] adresa, normálně 0x30.
- Control Byte - HDLC kontrolní byte.
- TPDU(Transport Protocol Data Unit) - pět bytů, které předchází aplikačním datům. Obsahují informace o typu TPDU, kde 0x60 představuje transakci a 0x68 administrativní data, dále obsahují adresu serveru (Destination Address, Network International Identifier) a adresu terminálu (Originator Address).

Tělo zprávy

Nejdůležitější data jsou součástí těla zprávy, vyskytují se zde totiž veškeré informace o typu transakce a hodnoty jednotlivých položek. Obecně se tělo zprávy skládá ze tří částí (tabulka 2.3)[6]:

[2]Hypercom. Dostupné z: http://hypercom.org/

[3]High-Level Data Link Control. Dostpuné z: https://en.wikipedia.org/wiki/High-Level_Data_Link_Control

Message Type	Bit Map	Application Data
2 B	8 B	var

Tabulka 2.3: Tělo zprávy protokolu ISO8583

- Message Type Identifier (Idenfikátor typu zprávy) - slouží pro určení typu zprávy transakce. Skládá se ze čtyř číslic, z nichž první dvě identifikují třídu zprávy (Message Class) a následující dvě pak identifikují režim přenosu (Transmission Mode). V rámci této práce bude nejdůležitější implementace chování pro třídu zprávy 0x02 (finanční transakce) a 0x04 (reversal) a pro režim přenosu 0x00 (interaktivní požadavek) a 0x10 (interaktivní odpověď).
- Bit Map (Bitová mapa) - protokol ISO8583 [8] používá bitovou mapu pro určení přítomnosti datových polí v samotných datech. Každá položka má přiřazenou svoji unikátní pozici v rámci bitové mapy a pokud se tato položka nachází v datech, je tento bit nastaven na hodnotu *jedna*, pokud se položka ve zprávě nenachází, pak má tento bit hodnotu *nula*. Velikost jedné bitové mapy je osm bytů, můžeme tedy přiřadit až 63 položek. Pokud potřebujeme více položek, je nutné přidat další bitmapu. K identifikaci, zda aplikační data obsahují další bitmapu, slouží první bit bitové mapy, který, pokud je nastaven na hodnotu *jedna*, značí přítomnost další bitmapy. V rámci popisované specifikace [6] bude první bit vždy nastaven na hodnotu *nula*, jelikož je zde použita pouze jedna bitová mapa.
- Data Elements (Položky zprávy, data) - obsahují konkrétní položky zprávy a jejich hodnoty, přičemž datové elementy uvnitř zprávy musí odpovídat následujícím pravidlům:
  - Všechny položky jsou zarovnány podle bytů.
  - Datové elementy, které mají pevnou délku v nibblech[4] a jejichž délka je lichá, jsou zarovnány doprava a zleva doplněny nulami. Například pole s formátem "n5" (pole o délce pět nibblů) bude mít velikost tři byty a bude začínat nulou.
  - Položky s proměnnou délkou jsou reprezentovány pomocí BCD[5], zarovnány doprava a zleva doplněny nulami. Například pro pole s formátem"LLVAR" a velikostí patnáct bude mít indikátor délky hodnotu 0x15 a velikost jeden byte. Pro pole s formátem "LLLVAR" se stejnou velikostí bude mít indikátor délky hodnotu 0x0015 a velikost dva byty.
  - Indikátor délky pro elementy s proměnnou délkou (VAR) představuje počet položek, které budou následovat, nikoli velikost.

Povinnými částmi těla zprávy jsou potom identifikátor typu zprávy, bitová mapa a některé vybrané datové elementy. To, zda jsou položky povinné (případně podmíněné) nebo ne dále určuje druh transakce.

[4] Nibble = je jednotka používáná v informatice, která má velikost čtyři bity. Dva nibbly tedy tvoří jeden byte.

[5] Binary-coded Decimal. Dostupné z https://en.wikipedia.org/wiki/Binary-coded_decimal

Popis jednotlivých/vybraných položek

Vzhledem k počtu všech možných položek jsou uvedeny pouze ty, které jsou nejčastější nebo povinné pro většinu transakcí [6].

- PAN (Primary Account Number) - obsahuje číslo karty (o maximální velikosti devatenáct numerických znaků) a používá se v případě, kdy není k dispozici informace o Tracku 1 nebo Tracku 2 z platební karty a zasílá se také pro všechny offline transakce.
- Processing Code - procesní kód se používá společně s identifikátorem zprávy pro určení správného typu transakce a je zasílán z terminálu na server. Obsahuje informaci o vybraném účtu a také informaci o řízení toku. Má fixní velikost šest nibblů. Informace o zvoleném účtu reprezentují třetí a čtvrtá číslice procesního kódu, nejčastější je pak hodnota "00" představující defaultní účet. Informace o řízení toku slouží pro případ, kdy chce server předat terminálu nějaké dodatečné informace v odpovědi, většinou je však tato hodnota stejná jako v požadavku. Například kombinace identifikátoru zprávy "0200" a procesního kódu "000000" představuje transakci prodej na defaultní účet.
- STAN (Systems Trace Audit Number) - hodnota, která je automaticky generována terminálem a je inkrementována pro každou transakci. STAN používá terminál pro validaci odpovědi a mělo by se využívat pouze pro kontrolu a identifikaci transakce, nikoliv pro zjištění, zda nebyla obdržena zpráva. STAN má fixní velikost šest nibblů a nikdy nesmí mít hodnotu "000000".
- RRN (Retrieval Refference Number) - hodnota určená serverem, kterou terminál používá pro offline transakce. Pokud terminál obdrží novou hodnotu RRN v odpovědi na offline transakci, uloží si ji namísto stávající hodnoty RRN. Má velikost dvanáct alfanumerických znaků, což odpovídá dvanácti bytům.
- Response Code - status transakce, který je obsažen v odpovědi od autorizačního serveru. Má velikost dva alfanumerické znaky. Hodnota "00" představuje schválený status, ostatní hodnoty pak neschválený nebo chybový status.
- Terminal ID - unikátní identifikátor terminálu o velikost osm alfanumerických znacích, včetně speciálních znaků.
- PIN Data - pole o velikosti osm bytů, ve kterém se nachází tzv. PIN block, což je šifra vygenerovaná terminálem pomocí interních klíčů obsahující PIN.
- ICC System Related Data - obsahuje informace potřebné acquirerem pro dokončení EMV transakce. Data jsou získána terminálem z čipové karty. Tato položka je dále rozdělena na další datové elementy, které jsou popsané pomocí formátu PDS (Private Data Sub-element), jehož podobu znázorňuje tabulka 2.4.

PDS Tag	Length	PDS Data
1 - 2 B	1 B	1 - 127 B

Tabulka 2.4: Struktura formátu PDS.

Jeden PDS element obsahuje následující položky:

- PDS Tag - identifikátor datové elementu přenášeného v PDS datech. Má velikost jeden nebo dva byty, podle specifikace.
- Length - velikost položky PDS Data, vyjádřený jako binární číslo v rozsahu 1 až 127.
- PDS Data - obsahuje konkrétní data odpovídající PDS Tagu.

Pro úplnost si uvedeme příklad transakce zaslané prostřednictvím protokolu ISO8583 a rozeberme si jej. Zaslaná transakce:

```
"00ED60008400000200303C058020C0000000000000000000012345000024090241110615120051008400344761739001010010D1512221114380448931323334353637385 34E5454455354393837363534333 2"
```

Hlavička: "00ED60008400000"

- Address = 00,
- Control Byte = ED,
- ID = 60,
- Destination Address = 0084,
- Originator Address = 0000.

Tělo zprávy: "0200303C058020C000000000000000000012345000024090241110615120051008400344761739001010010D1512221114380448931323334353637385 34E5454455354393837363534333 2"

- Message Type = 0200 (finanční transakce, požadavek),
- Bit Map = 303C058020C00000 (položky 3, 4, 11, 12, 13, 14, 22, 24, 25, 35, 41 a 42),
- Processing Code (položka 3) = 000000 (Msg Type 0200 => Prodej),
- Amount (položka 4) = 000000012345 (123.45),
- STAN (položka 11) = 000024,
- Time, Local Tran. (položka 12) = 090241,
- Date, Local Tran. (položka 13) = 1106,
- Date, Expiration (položka 14) = 1512,
- POS Entry Mode (položka 22) = 051,
- NII (položka 24) = 084,
- POS Condition Code (položka 25) = 00,
- Track 2 Data (položka 35) = 4761739001010010D15122211143804489,
- Terminal ID (položka 41) = 12345678,
- Card Acq ID (položka 42) = SNTTEST98765432.

2.2 Anylýza protokolu SPDH

Protokol SPDH[1] je vydáván a spravován společností ACI Worldwide[6], která se zabývá zprostředkováním finančních transakcí a služeb, a nabízí širokou škálu softwaru pro tuto činnost. Hlavním cílem protokolu SPDH je pak popsat zprávy vyměňované mezi hostitelem a terminálem (případně jiným zařízením), definovat veškeré přípustné transakce včetně všech datových elementů, které se mohou objevit ve dvojici zpráv požadavek a odpověď, a také popsat tok zpráv mezi hostitelským serverem a platebním zařízením[1].

Struktura zprávy

Zpráva protokolu SPDH musí obsahovat pouze data v čitelné podobě (ASCII zobrazitelné znaky), má jasně danou strukturu a formát a skládá se z částí uvedených v tabulce 2.5. Některé položky zprávy, především u hlavičky, mohou být prázdné, v takovém případě musí být takové pole vyplněno mezerami (0x20).

<table>
<tr><td rowspan="2">Header</td><td>FID 1</td><td>FID x</td><td rowspan="2">ETX</td></tr>
<tr><td colspan="2">Data</td></tr>
</table>

Tabulka 2.5: Struktura zprávy SPDH.

- Header - hlavička zprávy je povinná, má pevně definovanou strukturu, její délka je 48 bytů a obsahuje třináct položek.
- Data - představují tělo zprávy, dělí se na jednotlivé položky, kterým se v rámci protokolu SPDH říká FIDy[7]. Ty obsahují data jednotlivých položek, jsou nepovinné a nezáleží na jejich pořadí. Před každým FIDem se nachází tzv. field separator "." (0x1C). FID může být dále rozčleněn na SFIDy[8], každý SFID pak předchází tzv. record separator "!" (0x1E). Na příkladu zprávy v tabulce 2.5 máme dva FIDy, jeden 1 a druhý x.
- ETX (End of text) - znak (0x03), který značí konec zprávy.

Hlavička zprávy

Hlavička zprávy je povinnou složkou každého požadavku zaslaného mezi terminálem a serverem a také každé odpovědi ze serveru na terminál. Obsahuje třináct povinných položek o celkové velikosti 48 bytů, které se musí objevit v předem stanoveném pořadí a nikdy ne jinak. V tabulce 2.6 můžeme vidět, jaké informace hlavička obsahuje. Jsou to údaje o typu transakce, zaměstnanci, který transakci provedl, identifikátor terminálu, kód transakce a další[1].

Nyní si uvedeme význam jednotlivých položek hlavičky[1]:

- Device Type (typ zařízení) - kód, který se používá pro identifikaci jednotlivých zařízení a může být použit pro přesměrování zprávy z dial-up terminálu na příslušný

[6] ACI Worldwide. Dostupné z: https://www.aciworldwide.com/
[7] FID = Field Identifier
[8] SFID = Subfield Identifier

Pozice	Velikost	Název pole
1-2	2 an	Device Type
3-4	2 n	Transmission Number
5-20	16 an	Terminal ID
21-26	6 an	Employee ID
27-32	6 n	Current Date
33-38	6 n	Current Time
39	1 an	Message Type
40	1 an	Message SubType
41-42	2 n	Transaction Code
43	1 n	Processing Flag 1
44	1 n	Processing Flag 2
45	1 n	Processing Flag 3
46-48	3 n	Response Code

Tabulka 2.6: Hlavička protokolu SPDH.

proces. Má velikost dva alfanumerické znaky a hodnota typu zařízení "9." značí dial-up zařízení.

- Transmission Number (číslo přenosu) - dvouciferné číslo, které se používá pro detekci duplicitních požadavků zarovnané vpravo, zleva doplněno nulami. Pokud toto pole obsahuje nenulovou hodnotu, která je shodná s hodnotou předchozího požadavku k němuž byla zaslána platná odpověď, tak server takovýto požadavek odmítne a pošle příslušnou odpověď. Nulová hodnota tohoto pole značí, že se nemá kontrolovat s číslem předchozího požadavku.

- Terminal ID (id terminálu) - unikátní identifikátor terminálu, který se používá pro správné zařazení záznamu o transakci do databáze hostitele. Toto pole může mít až šestnáct alfanumerických znaků zarovnaných vlevo. Pokud je id terminálu menší než šestnáct znaků, je zbylé místo ponecháno prázdné, tak aby celá hlavička měla velikost 48 bytů.

- Employee ID (číslo zaměstnance) - unikátní identifikátor zaměstnance, který provedl transakci, o maximální délce šest alfanumerických znaků zarovnaných vlevo. Využívá se pro výpočet součtů daného zaměstnance.

- Current Date (aktuální datum) - datum transakce ve formátu YYMMDD. Není povinné, pokud je však součástí požadavku, tak obsahuje lokální čas terminálu a na straně serveru probíhá jeho kontrola. Pokud je ve špatném formátu nebo chybné, server posílá správné datum. Pokud je toto pole vyplněná, musí být vyplněn i aktuální čas (Current Time).

- Current Time (akutální čas) - čas transakce ve formátu hhmmss, kde "000000" představuje půlnoc. Platí zde stejná pravidla jako v případě data, pokud je ve špatném formátu nebo chybné (například kvůli změně z letního na zimní), tak server v odpovědi zasílá správnou hodnotu. Pokud je nastaven čas, musí být nastaveno i datum.

- Message Type (typ zprávy) - typ zprávy, který značí, zda jde o transakci finanční nebo administrativní. Společně s kódem transakce (Transaction Code) identifikuje typ transakce na straně serveru. Možné hodnoty jsou:
  - A = administrativní transakce,
  - F = finanční transakce,
  - a dále hodnoy L a M, která jsou méně používané.
- Message SubType (podtyp zprávy) - podtyp zprávy, který značí, zda se má zpráva zpracovat online, zda jde o reversal nebo zda má být transakce vynucená (force-post). Přípustné hodnoty podtypu zprávy jsou:
  - O = Online,
  - F = Force-post,
  - S = Store and forward,
  - C, U, A, T, R = hodnoty používané pro různý druhy reversal transakcí.
- Transaction Code (kód transakce) - kód, který představuje typ přenášené transakce. Společně s hodnotou typu zprávy identifikuje typ transakce na straně serveru. Má podobu dvouciferného čísla.
- Processing Flags (1, 2, 3) - příznaky sloužící k upřesnění zpracování transakce a případné specifikaci další operace. Například Processing Flag 2 v odpovědi značí, že terminál má provést inicializaci.
- Response Code (kód odpovědi)- hodnota, která představuje výsledek aktuálně zpracované transakce. V rámci požadavku se tato položka nevyužívá. Smysl má až v případě odpovědi. Návratových kódů je opravdu velké množství, ty nejdůležitější jsou však "000", který představuje schválenou transakci a hodnota "050" pak zamítnutou.

Tělo zprávy

Tělo zprávy obsahuje další informace o právě prováděné transakci jako je například částka, id zákazníka, Track 2 nebo zpráva, která se má zobrazit na obrazovce platebního zařízení. Většina položek je zde nepovinných, nicméně záleží na typu transakce. Položka v rámci protokolu SPDH se nazývá FID, pro každý FID pak platí:

- Každý FID má unikátní identifikátor o velikost jeden byte,
- vždy mu předchází tzv. field separator "." (0x1C),
- nezáleží na pořadí jednotlivých FIDů,
- FID může být dále rozdělen na SFIDy:
  - Každý SFID má svůj unikátní jeden byte identifikátor,
  - vždy mu předchází tzv. record separator "!" (0x1E),
  - nezáleží na pořadí jednotlivých SFIDů.
- pouze FIDy 6, 7, 8 a 9 obsahují SFIDy.

Popis jednotlivých/vybraných položek

Vzhledem k počtu možných položek jsou uvedeny pouze důležité nebo nejčastěji se vyskytující položky[1].

- Amount 1 (FID B) - primární pole pro přenos částky transakce. V případě, že transakce obsahuje pouze jednu částku, je uvedena v tomto poli, v případě více částek je zde originální částka. Lze zadat maximálně osmnáctimístnou částku.
- Amount 2 (FID C) - sekundární pole pro přenos částky. Používá se v případě, kdy jsou vyžadovány dvě částky, například pro transakci s cash backem je originální částka v poli FID B (Amount 1) a částky cash backu v položce FID C (Amount 2). Stejně jako originální částka může mít maximálně osmnáct číslic.
- Application Account Type (FID D) - tato položka má velikost jeden byte a představuje typ účtu pro danou transakci. Pokud není součástí požadavku, je vybrán defaultní typ účtu.
- Approval Code (FID F) - obsahuje unikátní číslo vygenerované autorizací pro aktuální transakci.
- Authentication Code (FID G) - obsahuje osmi bytový hexadecimální MAC, který je použit k verifikaci zprávy.
- Track 2/Customer (FID q) - položka obsahuje přečtený Track 2 (případně jeho ekvivalent) z karty zákazníka. Pro většinu finančních transakcí je povinný buďto Track 2 (FID q) nebo Track 1 (FID 2). Pole může obsahovat maximálně čtyři alfanumerické znaky a má specifický formát.
- Track 1/Customer (FID 2) - tato položka má svůj specifický formát, může obsahovat maximálně 82 alfanumerických znaků a slouží pro přenos dat přečtených z Tracku 1 zákazníkovi karty.
- Optional Data Subfields (FID 6) - obsahuje SFIDy s dalšími údaji, jako je například POS Entry Mode, STAN nebo EMV request data. Například položka POS Entry Mode (SFID E) je třímístný kód, který obsahuje informace o tom, jakým způsobem byla transakce zadána a potvrzena, zda manuálně, magnetickým proužkem nebo třeba pomocí čipové karty.

Pro úplnost si dále uvedeme příklad transakce zaslané prostřednictvím protokolu SPDH a její rozbor. Zaslaná transakce:

```
"9.23POS-TERMINAL-056987654161231235959F000000000.qM1234567890123456789=0108
0?.B12345"
```

Hlavička: `"9.23POS-TERMINAL-056987654161231235959F000000000"`

- Device Type = 9. (dial-up zařízení),
- Transmission Number = 23 (číslo bude ověřeno vůči předchozí transakci),
- Terminal ID = POS-TERMINAL-056,

- Employee ID = 987654,
- Current Date = 161231,
- Current Time = 235959,
- Message Type = F (Finanční transakce),
- Message SubType = O (zpracování online),
- Transaction Code = 00 (prodej),
- Processing Flags = 0, 0, 0,
- Response Code = 000.

Tělo zprávy: `".qM1234567890123456789=01080?.B12345"`

- FID q (Track 2) = M1234567890123456789=01080?
- FID B (Amount 1) = 123.45

Protokoly ISO8583 a SPDH jsou vzájemně velice odlišné. Můžeme říci, že společnými prvky těchto protokolů je to, že jsou využívané pro přenos finančních/bankovních dat a také jejich obsah je v případě mnoha finančních transakcí velmi podobný, ne-li stejný. Nicméně v dalších ohledech už jsou protokoly odlišné. Ať už je to podoba přenášených dat, jejich formát, nebo podoba samotné zprávy a jejich položek. Krátké srovnání obou protokolů je uvedeno v tabulce 2.7.

	ISO8583	**SPDH**
Formát dat	binární	ASCII
Hlavička	směrovací informace	transakční data
Položky zprávy	pevné pořadí	na pořadí nezáleží
Identifikace přenášených položek	bitová mapa a offset	field/record separator

Tabulka 2.7: Porovnání protokolů ISO8583 a SPDH.

I přesto, že jsou oba zmíněné protokoly odlišné, mohou být zpracovány na jednom a tom samém platebním terminálu. To, zda terminál umí komunikovat pomocí určitého protokolu, záleží pouze na aplikaci, která je v něm nahraná. Současně tedy může být jeden terminál schopen zpracovat několik různých finančních protokolů.

Kapitola 3

Návrh simulátoru

Pro návrh simulátoru je nutné znát základní pojmy z oblasti bankovnictví, požadované autorizační protokoly, dále principy práce a postupy samotné autorizace, ale také mít pokud možno přesný popis požadavků a specifikaci výsledného programu. Po konzultacích ve společnosti Sonet, s.r.o., o podobě simulátoru pro finanční protokoly, vzešly tyto minimální požadavky:

- Správné parsování protokolů, neboli určení hlavičky a jejich částí, získání těla zprávy a jednotlivých přenášených položek.
- Validace protokolů:
  - kontrola správného formátu zprávy včetně jejich jednotlivých částí,
  - kontrola zda daná zpráva obsahuje všechny povinné položky.
- Vytvoření relevantní odpovědi na přijatou zprávu v minimálním rozsahu, čímž se rozumí zpracování základních response kódů jako je například "Approved" (Schváleno), "Declined" (Zamítnuto), "Wrong format" (Nevalidní formát), atd.
- Možnost upravit pro odchozí odpověď položky Response code a Response text.
- Zobrazovat zpracované zprávy:
  - neformátové, tedy tak, jak byly přijaty,
  - formátované, neboli rozdělené na hlavičku a tělo zprávy. Zobrazovat jednotlivé položky hlavičky a zprávy odděleně.
- Snadno rozšiřitelné pro další protokoly. Využít například konfigurační soubor ve formátu .xml pro popis protokolu a jeho důležitých částí, v rámci aplikace zajistit serializaci/deserializaci.

Hlavní cíle práce vychází z konečných požadavků uvedených výše a jsou následující:

- validace formátu příchozího požadavku,
- správné určení návratového kódu (Response kódu),
- sestavení odpovědi na základě příchozího požadavku a uživatelských[1] parametrů.

[1]Pojem uživatel v rámci této práce zahrnuje vývojáře a členy testovacího oddělení, tedy uživatele, kteří mají širší zkušenosti v oblasti informačních technologií a oblasti bankovnictví z hlediska terminálu a platebních aplikací.

Jak již bylo uvedeno v kapitole 2, návratových kódů je poměrně velké množství a především pro určení některých návratových kódů je nutné mít velké množství specifických informací. Proto bude simulátor omezen pouze na základní, běžné, response kódy, u kterých nejsou potřeba žádná dodatečná data nebo historie takových dat. Určení správného návratového kódu může být také ovlivněno uživatelskými parametry, kdy pro účely testování může být například situace, aby následující požadavky byly zamítnuty, nebo aby byl v odpovědi nastaven specifický response kód. Simulátor dále nemá v žádném případě nahradit autorizační centrum a nemá tedy mít ani jeho plnohodnotné vyhodnocovací schopnosti a funkce. Jedná se o jednoduchý simulátor, který má za cíl usnadnit vývoj a testování nových protokolů a také usnadnit simulaci možných chybových stavů.

Simulátor bude rozdělen na několik logických částí, modulů, z nichž jeho jádro budou tvořit ty nejdůležitější:

- *Responder* - řídící modul, společný pro všechny protokoly, předává přijatou zprávu parseru, sestavuje konečnou odpověď a odesílá ji terminálu.
- Parser - parsování protokolů, validace prvního stupně, například validace hlavičky zprávy a přítomnost povinných položek.
- Validator - validace zprávy jako celku, tedy validace jednotlivých částí zprávy a jejích položek. Kontrola velikosti, formátu a typu dat.
- Response Builder - na základě výsledku parsování a validace dále rozhoduje o konečném návratovém kódu odpovědi.

Kromě výše zmíněných modulů bude simulátor obsahovat také tyto další části:

- Komunikátor - zajištění komunikace mezi terminálem a simulátor včetně všech nutných rutin.
- Logger - modul, který bude sloužit pro logování běhu simulátoru. Pokud bude logování na správných místech a bude zachycovat podstatné informace, může výrazně pomoci s vývojem simulátoru, ale také s odstraněním případných chyb.
- Utils - pomocné metody,
- a případně další součásti simulátoru.

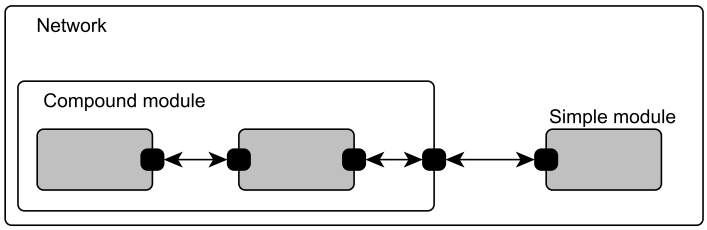
Pro budoucí vývoj a rozšíření simulátoru je nutné provést úvahu nad komunikací a předáváním dat mezi moduly, kterými mohou kromě jádra být GUI, ale také především databáze nebo skripty simulující autorizaci a vykonávající činnost nad rámec jádra simulátoru.

3.1 OMNeT++

OMNeT++[2] je modulární objektově orientované diskrétní simulační prostředí. Základním stavebním prvekm OMNeTu jsou moduly, které mohou být vzájemně propojené prostřednictvím bran (portů) a mohou být dále seskupovány do komplexních složených modulů. Moduly pak mohou obsahovat nejrůznější parametry a rozlišujeme tyto dva základní moduly[13]:

[2]OMNeT++ - Discrete Event Simulator. Dostupné z: https://omnetpp.org/

- Jednoduchý (single) modul - aktivní prvek modelu, obsahuje logiku a chování definované uživatelem v jazyku C++. Nejdůležitější částí každého jednoduchého modulu je implementace reakce na přijatou zprávu a její další zpracování.
- Složený (compound) modul - komplexní modul složený z jednoduchých modulů.



Obrázek 3.1: Moduly v prostředí OMNeT++.

Moduly mezi sebou vzájemně komunikují pomocí zasílání zpráv, které mohou obsahovat nejrůznější data a struktury. Lze definovat i vlastní typ zprávy, který bude obsahovat specifické informace. Zprávy jsou pak zasílané skrze brány jednotlivých modulů.

Další důležitou součástí celého konceptu jsou konfigurační soubory. Mezi nejdůležitější patří:

- .NED (Network definition file) - obsahuje popis struktury modulu a jeho parametry, jako jsou například brány a kanály s ostatními moduly.
- .ini (konfigurační soubor) - konfigurace a parametry simulace.
- .msg (Message definition) - uživatelem definované zprávy, které mohou být v rámci modelu přenášeny.

OMNeT++ obsahuje také vývojové prostředí založené na prostředí Eclipse, je volně šiřitelný a přenosný mezi běžně používanými operačními systémy Microsoft Windows, Linux a Mac OS.

V rámci návrhu a analýzy simulačního prostředí bylo zjištěno, že použití OMNeT++ pro potřeby finančního simulátoru by bylo příliš složité a nepřineslo by předpokládané výhody a zjednodušení. Proto nebyl OMNeT++ v konečném důsledku použit.

3.2 Jádro simulátoru

Jádro simulátoru budou představovat nejpodstatnější části potřebné ke zpracování autorizační zprávy a sestavení relevantní odpovědi. Pro tyto moduly je nutné určit jejich formu, tj. do jaké míry mohou být obecné, a kdy je naopak nutné, aby byly pokud možno co nejkonkrétnější. Dále musíme na základě analýzy zpracovávaných protokolů rozhodnout, které jejich části jsou pro nás signifikantní a je nezbytné je začlenit, a naopak, které součásti jsou pro účely simulátoru nepodstatné. Z hlediska odlišnosti protokolů ISO8583[8] a SPDH[1] se musíme při návrhu také soustředit na to, které moduly mohou být společné, a které je naopak nutné oddělit.

Vzhledem k odlišnostem zpracovávaných protokolů, které jsou popsány v tabulce 2.7, není moc součástí, které bychom mohli sloučit, a které by mohly být společné. Nicméně

jedním ze základních, a současně společných prvků, je přenos bankovních a finančních dat. Tato data jsou mnohdy stejná nebo mají velice podobnou formu. Například částka zpracovávané transakce musí být číslo, jednotlivé číslice se tedy musí skládat z hodnot z intervalu $\langle 0, 9 \rangle$. Stejných hodnot pak může nabývat například číslo účtu, zůstatek na účtu nebo nejrůznější autorizační a jiné kódy. Naopak číslo terminálu nebo číslo acquirera mohou nabývat alfanumerických hodnot, včetně speciálních znaků jako je například mezera nebo vykřičník. Velmi specifickým případem jsou poté položky, které mohou nabývat pouze určitých, předem daných, hodnot. Tedy jejich hodnoty se musí shodovat s povolených výčtem.

3.2.1 Konfigurace protokolů

Výše popsané skutečnosti jsou shodné pro celou řadu finančních protokolů, nejenom pro protokoly zpracovávané v rámci této práce, a můžeme tedy na základě těchto skutečností navrhnout univerzální popis protokolu a prostřednictvím tohoto popisu, pak dále zvolit jednotnou strukturu, která bude takovouto konfiguraci udržovat a umožní nám s ní pracovat. Jaká data je nutno definovat:

- Identifikátor pole - položka protokolu má vždy svůj jednoznačný identifikátor. Ten ji částečně popisuje a pomůže nám třídit a hledat další informace o poli.
- Formát pole - pro správné zpracování je nutné vědět, jaký formát má mít dané pole. Jestli se má jednat o číslo nebo jestli se má jednat o alfanumerickou hodnotu, atd.
- Velikost pole - velikost pole je nezbytná pro validaci, ale v případě protokolu ISO8583 je také nezbytná pro samotné parsování dat.
- Název pole - tato položka více méně není nutná, nicméně jedním z požadavků je i zobrazit rozparsovaná příchozí data. Zobrazení dat položky společně s identifikátorem by mohlo být dostačující, nicméně textový popis položky může být v tomto případě přínosem.
- Hodnoty pole - pokud pole nabývá předem daných hodnot, musíme tyto hodnoty samozřejmě znát. Proto je nutné uvést v takovém případě jejich výčet. Především je toto důležité z hlediska určení typu příchozí zprávy.

Přičemž pro atributy formát (format) a velikost dat(length) platí:

- Velikost dat(length) - pokud obsahuje pouze jednu hodnotu (číslo), dané pole má fixní velikost. Jestliže obsahuje dvě hodnoty oddělené dvěma tečkami, tak první číslo představuje minimální velikost pole a druhé pak maximální velikost pole. Pokud atribut obsahuje jedno číslo, kterému předchází dvě tečky, tak může mít pole velikost v intervalu $\langle 1, maxValue \rangle$.
- Formát (format) - tento atribut může nabývat pouze předem daných hodnot:
  - A - pole s tímto atributem může nabývat hodnot z intervalu $\langle a, z \rangle$ a intervalu $\langle A, Z \rangle$.
  - N - představuje numerického hodnoty.
  - S - hodnoty reprezentuje speciální znaky jako například mezera nebo vykřičník.
  - n - nibble, neboli polovina bytu.
  - b - binární data.

```
<?xml version="1.0" encoding="UTF-8"?>
<protocol>
    <field id="1" name="Field 1" format="N" length="4..8" />
    <field id="2" name="Field 2" format="A,N" length="..12" />
    <field id="A" name="Field A with subfields" format="SF" length="10" >
        <subfields>
            <subfield id="1" name="SF 1" length="4"/>
            <subfield id="2" name="SF 2" length="3"/>
            <subfield id="3" name="SF 3" length="3"/>
        </subfields>
    </field>
    <field id="B" name="Field B with values" format="N" length="4"
     values="fieldBval" />

    <Values id="fieldBval" enum="0110, 0120, 0130, 0400" />
</protocol>
```

Obrázek 3.2: Příklad konfiguračního souboru ve formátu xml.

– SF - tato hodnota identifikuje vnořené hodnoty pole.

Mimo vyjmenované hodnoty jsou povolené kombinace AN a ANS.

Některé položky mohou mít ale složitější formát a mohou samy nést další pole nebo se mohou různě členit. V takovém případě musíme myslet na to, že existuje možnost vnořování popisu dat v konfiguraci. Pro takový konfigurační soubor je vhodný formát xml [3], který si můžeme upravit podle našich potřeb a zadefinovat si vlastní elementy a jejich atributy. Příklad konfiguračního souboru je uveden v popisu 3.2.

Konfigurační souboru můžeme dále využít pro definování známých a podporovaných transakcí včetně jejich povinných položek. Zde je ovšem nutné již rozlišit oba protokoly. Jednak každý používá jiné položky pro identifikaci transakce, ale především pak mohou mít pro stejné transakce různé povinné pole. Tento fakt bude nutné zohlednit také při čtení a ukládání samotné konfigurace do vnitřní struktury simulátoru. Protokol SPDH určuje transakce na základě polí ve své hlavičce, konkrétně na základě polí **Message Type** a **Transaction Code**. Oproti tomu protokol ISO8583 využívá k identifikaci transakce hodnotu **Message Type ID** a **Processing Code**. Dalším rozdílem je také to, že protokol SPDH přenáší velké množství dat prostřednictvím svojí hlavičky, kdežto ISO8583 přenáší veškerá užitečná data mimo hlavičku. Proto některé transakce protokolu SPDH, například test linky, neobsahují žádná data mimo hlavičku. Rozšíření konfiguračního souboru o podporované transakce a povinné položky může mít například podobu z obrázku 3.3.

Pro každou podporovanou transakci je tedy nutné definovat následující položky:

- Identifikátor transakce - hodnoty, které identifikují transakci.
- Výčet povinných položek - položky, které se musejí objevit v požadavku. Zde je nutné myslet také na to, že mnohé transakce definují několik povolených kombinací položek. Například transakce prodej u protokolu SPDH dovoluje povinné položky ve složení částka a Track 2 nebo částka a Track 1, apod.

[3] eXtensible Markup Language. Dostupné z: https://www.w3.org/XML/

```
<?xml version="1.0" encoding="UTF-8"?>
<protocol>
    <spdhMessages>
        <message id="F,00" name="Normal Purchase" >
            <req values="B,q/B,2/B,q,6(E)/B,q,6(EIO)" />
        </message>
        <message id="F,01" name="Preauthorization Purchase" >
            <req values="B,i,q/B,i,2/B,i,q,6(E)/B,i,q,6(EIO)" />
        </message>
    </spdhMessages>
    <isoMessages>
        <message id="0100,31**0*" name="Balance" >
            <req values="3,11,22,24,25,41,42,45/3,11,22,24,25,35,41,42,55"/>
        </message>
        <message id="0200,00**0*" name="Sale" >
            <req values="3,4,11,22,24,25,41,42/3,4,11,22,24,25,35,41,55"/>
        </message>
    </isoMessages>
</protocol>
```

Obrázek 3.3: Příklad konfigurace transakcí a povinných položek zprávy ve formátu xml.

- Název transakce - identifikátor transakce nemusí být na první pohled jasný, proto je dobré mít k dispozici i název, případně popis, transakce.

3.2.2 Validace dat

Dalším společným prvkem může být validace přenášených dat. Jak jsme uvedli výše, obsah transakcí má několik společných rysů. Na základě informací z konfiguračního souboru, pak můžeme rozhodnout, zda jsou data validní nebo ne. Využít můžeme především údaj o velikosti konkrétní položky a také formát konkrétního pole. Samozřejmě některé položky mohou mít složitější, specifický nebo nezvyklý formát dat, který nelze jednoduše zachytit prostřednictvím konfiguračního souboru. Může jít třeba o Track 2, který nejenže může nabývat různých hodnot, může mít různé formáty, ale také se liší napříč finančními protokoly. Například pro protokol SPDH může vypadata Track 2 následovně:

- **;**PAN=YYMMdata**?** nebo **M**PAN=YYMMdata**?**, kde znaky **;** a **M** jsou začátek Tracku 2, znak = pak odděluje PAN od dalších dat, a symbol **?** je ukončovací znak.

V případě protokolu ISO8583 má potom Track 2 následující podobu:

- PAN**d**YYMMdata**f**, kde **d** odděluje PAN od dalších dat a **f** představuje ukončovací znak.

Proto takové položky, které jsou podobného formátu a jsou z hlediska autorizace podstatné, je nutné zpracovávat odděleně a informace z konfiguračního souboru využít pouze pro doplnění nebo získání takovýchto dat.

Sjednotit přístup pro oba protokoly tedy můžeme v případě jejich konfigurace, kde jsou data podobného formátu a je zde několik společných prvků. Částečně sjednotit pak

můžeme validaci dat, kde nám pomůže právě jednotný formát konfigurace. Validaci pak můžeme provést nad daty z hlediska jejich obsahu a velikosti.

3.2.3 Parsování dat

Další důležitou a nezbytnou součástí je parsování dat. Zde ovšem nelze uplatnit žádnou společnost vlastnost. Můžeme pouze říci, že finanční protokoly obvykle obsahují hlavičku a data, kde hlavička má pevně stanovený formát a data nikoliv. Parsování příchozích dat je tedy vhodné rozdělit na parsování hlavičky a parsování samotných dat.

Pro protokol SPDH platí, že bezprostředně za hlavičkou následují data, kdy každé datové položce předchází field separator, který má hodnotu 0x1C. Užitečná data se tedy nachází mezi oddělovači datových položek.

Situace u protokolu ISO8583 je poněkud složitější. Data nejsou nijak oddělena a následují bezprostředně za sebou a pro jejich správné čtení je nutné mít k dispozici jejich velikost. Nejprve ovšem musíme získat informaci o tom, jaké položky se v datech nachází. To zjistíme prostřednictvím bitové mapy a z konfigurace pak získáme velikost dat pro danou položku. Výjimku ovšem tvoří pole 55 (ICC System Related Data), které je ve formátu PDS (tabulka 2.4).

3.2.4 Responder

Nejdůležitější částí celého simulátoru pak bude bezpochyby responder, modul, který budu mít za úkol sestavení validní odpovědi. Pro sestavení takové odpovědi musíme určit signifikantní položky požadavku, které se musí objevit v odpovědi. Je nutné také určit, které položky je nutné vygenerovat, které pouze zopakovat, a které lze vynechat. Velmi důležitou funkcí je pak určení response kódu. Vzhledem k velké odlišnosti struktury obou zpracovávaných protokolů nelze určit společné prvky, a proto je nutné mít oddělené respondery.

Jak již bylo zmíněno, tak protokol SPDH přenáší velké množství dat prostřednictvím svojí hlavičky, která je povinnou součástí protokolu. Většina dat z hlavičky požadavku je v odpovědi zopakovaná. Výjimkou je datum a čas, který se generuje a v odpovědi se zasílá aktuální datum a čas simulátoru. Nejdůležitější částí hlavičky je pak response kód. Ten určíme na základě výsledků parsování a validace. Návratových kódů je celá řada, nicméně pro potřeby simulátoru stačí zpracovat jen ty základní, mezi než patří:

- 800 (Chyba formátu) - response kód, který je vracen při chybě formátu, tedy v případě, kdy dojde k chybě parsování, kdy se ve zprávě nachází duplicitní položky, anebo když v požadavku chybí povinné součásti.
- 801 (Nevalidní data) - tato hodnota je vrácena, pokud je některá z položek v nevalidním stavu.
- 050 (Zamítnuto) - pro případ obecné chyby, chybějící hodnoty v konfiguraci nebo neznámé hodnotě.
- 052 (Neznámá transakce) - pokud přijdou identifikátory, které nejsou validní nebo jsou neznámé.
- 000 a 007 (Schváleno) - v jiných případech dojde ke schválení transakce. Kód 000 se používá pro finanční transakce, to jsou transakce s hodnotou F v poli Message Type. Kód 007 se poté zasílá pro schválení administrativních transakcí, například pro test linky.

Minimum pro validní odpověď na SPDH požadavek tvoří hlavička protokolu. V praxi se nicméně zasílají i další důležitá data. Mezi taková data pak patří například Approval Code, Sequence Number, případně Track 2 nebo částka. Nejběžnější položky, které mohou být součástí odpovědi, jsou shrnuty v tabulce 3.1. Nejsou zde ovšem všechny položky, jelikož existují transakce, které vyžadují v odpovědi specifická data.

ID	**Název pole**	**Generované**	**Povinné**	**Poznámky**
H1	Device Type	Ne	Ano	
H2	Transmission Number	Ne	Ano	
H3	Terminal ID	Ne	Ano	
H4	Employee ID	Ne	Ano	
H5	Current Date	Ano	Ano	
H6	Current Time	Ano	Ano	
H7	Message Type	Ne	Ano	
H8	Message SubType	Ne	Ano	
H9	Transaction Code	Ne	Ano	
H10	Processing Flag 1	Ne	Ano	Vždy 0
H11	Processing Flag 2	Ne	Ano	Vždy 0
H12	Processing Flag 3	Ne	Ano	Vždy 0
H13	Response Code	Ano	Ano	000, 007, 800, 801, 050, 052
B	Amount	Ne	Ne	
F	Approval Code	Ano	Ne	
q	Track 2	Ne	Ne	
h	Sequence Number	Ano	Ne	
g	Response Text	Ano	Ne	

Tabulka 3.1: Možné položky v odpovědi protokolu SPDH.

Pro protokol ISO8583 bude situace v případě response kódu velmi podobná. Jeho určení bude probíhat opět na základě výsledků parsování a validace a budou podporovány následující hodnoty:

- 12 (Neznámá transakce) - v případě nevalidních nebo chybných identifikátorů transakce.
- 30 (Chyba formátu) - hodnota zahrnuje chybu parsování, chybějící povinné položky nebo nevalidní data.
- 50 (Zamítnuto) - při obecné chybě, chybějící hodnotě v konfiguraci nebo neznámé hodnotě bude použit zamítavý návratový kód.
- 00 (Schváleno) - v ostatních případech budou transakce schvalovány.

Pří sestavování odpovědi pro protokol ISO8583 je nutné dodržet správné pořadí polí a podle identifikátoru pole upravit bitovou mapu. Přičemž opět musíme rozlišovat položky, které je nutné do odpovědi zahrnout, které jsou z nějakého důvodu důležité, a které můžeme ignorovat. Nejčastěji se vyskytující položky v ISO8583 odpovědi jsou uvedeny v tabulce 3.2.

ID	Název pole	Generované	Povinné	Poznámky
H1	Message Type	Ano	Ano	
H2	Bitmap	Ano	Ano	
3	Processing Code	Ne	Ano	
4	Amount	Ne	Ne	
11	STAN	Ne	Ano	
12	Time	Ano	Ano	
13	Date	Ano	Ano	
24	NII	Ne	Ano	
37	RRN	Ano	Ano	
38	Auth. ID Response	Ano	Ne	
39	Response Code	Ano	Ano	00, 12, 30, 50
41	Terminal ID	Ne	Ano	

Tabulka 3.2: Možné položky v odpovědi protokolu ISO8583.

3.2.5 Uživatelem definovaná data odpovědi

Důležitou součástí simulátoru je také možnost alespoň částečně upravovat odpověď. A to minimálně možnost zvolit vlastní návratový kód. Tato funkcionalita musí být nastavitelná v průběhu chodu simulátoru. Jako vhodný formát pro takové nastavení je soubor ve formátu ini[4]. Tento konfigurační soubor bude čten pro každou dvojici požadavek-odpověď, tak aby mohl uživatel dynamicky měnit hodnoty odpovědi. Měnit hodnoty pro každou odpověď však může být nepohodlné, proto by bylo dobré mít i jinou možnost. Lze například využít pole částka, které je přítomno u většiny finančních transakcí, a na základě částky určit response kód pro odpověď. Musíme ovšem rozlišovat protokol, jelikož pro protokol SPDH obsahuje návratový kód tři číslice, a pro protokol ISO8583 pouze číslice dvě. Pokud bude chtít uživatel nastavit response kód na hodnotu například 50, při zadávání transakce nastaví částku tak, aby končila 50. Tedy třeba 0,50 Kč, 100,50 Kč, 2100,50 Kč a podobně. Nazpět pak bude nastaven takovýto návratový kód. V případě protokolu ISO8583 bude hodnota 50, v případě protokolu SPDH pak bude hodnota 050. Vždy se zpracují pouze poslední dvě, respektive tři číslice částky.

Pro ostatní položky, které bude možno nastavit, už je nutné přistupovat individuálně na základě daného protokolu. Pro nastavitelné položky nebude probíhat kontrola správnosti jejich hodnoty, jelikož nastavení chybné položky do odpovědi může být záměrem a předmětem testování chování terminálu pro určitý stav.

Uživatel musí mít také možnost vypnout použití nastavených položek a také musí být schopen nastavit pouze některou z nich. Proto je nutné v rámci konfigurace uživatelem definovaných položek odpovědi vyčlenit určitý přepínač nebo příznak, který bude značit, že se mají použít. Pokud bude tento příznak aktivní, tak se nadefinované položky použijí, ale pouze v případě, že budou mít nějakou hodnotu. Pokud bude položka prázdná, tak se nepoužije a buď bude z odpovědi vynechána, nebo bude použita vygenerovaná hodnota v rámci simulátoru.

Příklad konfiguračních souborů obsahujících uživatelem definovaná data odpovědi je uveden na obrázku 3.4. Položky *RSPCODEBYAMOUNT* a *USESPECIFIC* jsou typu bool a mohou nabývat pouze hodnot Y a N, kde Y představuje true a N pak false. Hodnota

[4]INI formát. Dostupné z: https://en.wikipedia.org/wiki/INI_file

```
[SPDH specific]
RSPCODEBYAMOUNT = N
USESPECIFIC = Y
RSPCODE = 069
AMOUNT =
RSPTEXT = AHOJ SVETE

[ISO specific]
RSPCODEBYAMOUNT = Y
USESPECIFIC = Y
RSPCODE = 69
AMOUNT = 1234
```

Obrázek 3.4: Příklad konfigurace uživatelem definovaných hodnot odpovědi.

položky *RSPCODEBYAMOUNT* určuje, zda se má použít částka transakce pro nastavení response kódu a hodnota položky *USESPECIFIC* pak určuje, zda použít specifické nadefinované hodnoty. Úpravou konfiguračních souborů uživatel nemůže ovlivnit chod simulátoru jako takového, má pouze možnost definovat některá data odpovědi.

Kapitola 4

Implementace

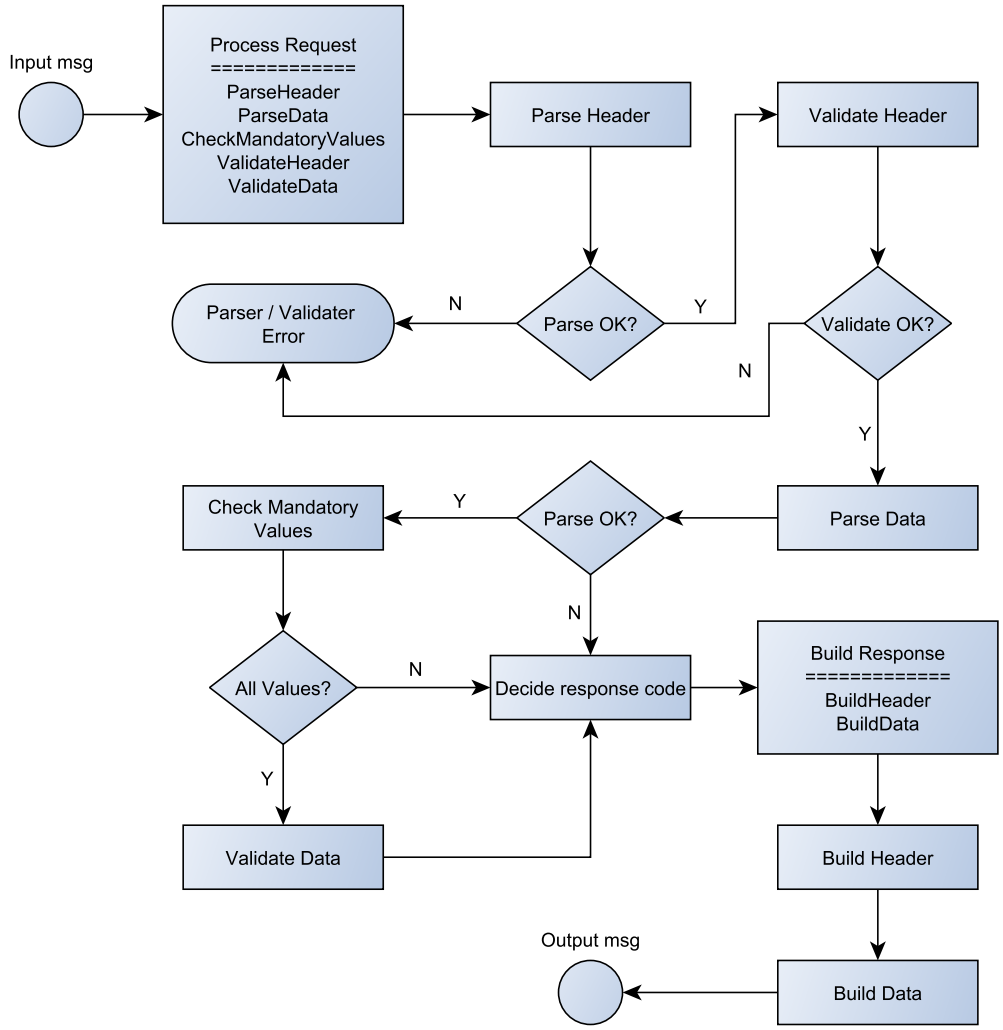
V předchozí kapitole 3 jsme popsali návrh simulátoru pro finanční protokoly. Tato kapitola se bude zabývat implementací výše zmíněného návrhu. Nejdříve si popíšeme proces zpracování příchozího požadavku a vytvoření validní odpovědi a následně se podíváme na některé implementační detaily. Simulátor je vyvíjen v jazyce C++ jako multiplatformní aplikace podporovaná na operačních systémech rodiny Microsoft Windows a Linux.

4.1 Zpracování příchozího požadavku

Zpracování příchozí požadavku, stejně jako sestavení odpovědi, bude mít na starost modul *Responder*. K tomu využívá modul *Parser*, který získá z příchozí zprávy příslušná data. *Parser* pak používá modul *Validator*, který validuje příchozí již zpracovaná data. Podle výsledku zpracování a validace dále *Responder* rozhodne o dalším postupu. Zpracování příchozího požadavku pak probíhá podle diagramu na obrázku 4.1. Poté, co jsou přijata data na úrovni komunikátoru (na socketu), tak dojde k vytvoření *Responderu*, který přijatou zprávu předá dále *Parseru*. *Parser* se nejdříve pokusí zpracovat hlavičku požadavku a následně ji zvalidovat. Pokud při parsování nebo validaci hlavičky dojde k chybě, již se nepokračuje dále, neprovádí se parsování dalších dat, nekontroluje se, zda zpráva obsahuje veškeré potřebné údaje, ani se neprovádí validace. *Responderu* je předána příslušná návratová hodnota podle níž rozhodne o tom zda a jak má být sestavena odpověď. Pokud proběhlo parsování a validace hlavičky v pořádku, tak se *Parser* pokusí zpracovat data požadavku. V případě úspěšného zpracování provede *Parser* kontrolu povinných položek a následně validaci. Výsledek kontroly je předán *Responderu*, který dále rozhodne o návratovém kódu a sestavení odpovědi.

Můžou ovšem nastat i situace, kdy přijde požadavek v tak špatném formátu, že simulátor není schopný určit, o jaký požadavek se vlastně jedná a zda jde o zprávu daného protokolu. V takovém případě simulátor nezasílá žádnou odpověď, a ukončí spojení s terminálem. V praxi je takové chování běžné, pokud autorizace obdrží neznámý požadavek, nereaguje na něj. Pokud obdrží požadavek, který umí zpracovat, ale je ve špatném formátu nebo obsahuje nevalidní data, tak se může zachovat dvěma způsoby:

- Nereagovat na takový požadavek.
- Odeslat odpověď se zamítavým response kódem. Případně zaslat i doplňující informace týkající se chyby.



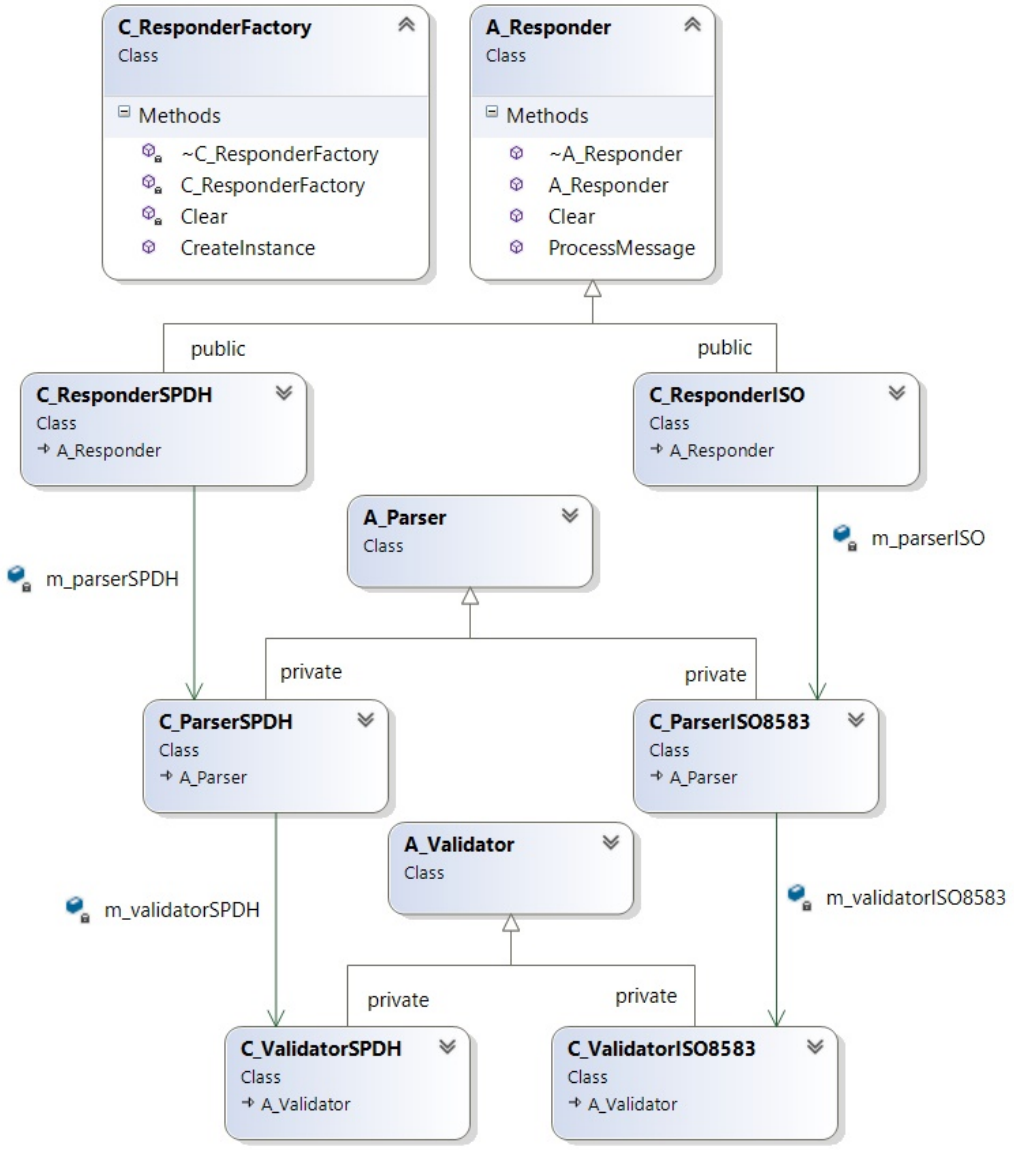
Obrázek 4.1: Proces zpracování příchozí zprávy a vytvoření odpovědi.

V případě simulátoru je snaha zasílat pokud možno vždy odpověď. Taková odpověď pak musí mít správný formát a validní response kód. Tedy pokud se nepodaří zpracovat a zvalidovat hlavičku příchozí zprávy, tak nedojde k odeslání odpovědi. Pokud se hlavičku podaří kompletně zpracovat, ale zpráva nebude obsahovat veškeré povinné údaje nebo dojde k chybě v rámci parsování nebo validace dat, bude odeslána odpověď a response kód bude zvolen na základě chybového kódu.

4.2 Vlastní implementace simulátoru

Základem celého simulátoru jsou tři již výše zmíněné moduly. Ty představují abstraktní třídy *A_Responder*, *A_Parser* a *A_Validator* a tvoří předky pro všechny další konkrétní třídy, zpracovávající konkrétní protokol. *Responder* třídy musí implementovat metodu *ProcessMessage* v rámci níž dochází ke zpracování požadavku a následnému vytvoření odpovědi. *Responder* pak obsahuje jako třídní atribut instanci konkrétního *Parseru* a *Parser* obsahuje instanci konkrétního *Validátoru*. O vytvoření správného *Responderu* se pak stará třída

C_ResponderFactory postavená na principu návrhového vzoru továrna[14], [2]. Tato třídy na základě identifikátoru protokolu vytvoří pro každý příchozí požadavek novou instanci daného *Responderu*. Struktura jádra simulátoru je naznačena na diagramu 4.2.



Obrázek 4.2: Digram tříd jádra simulátoru.

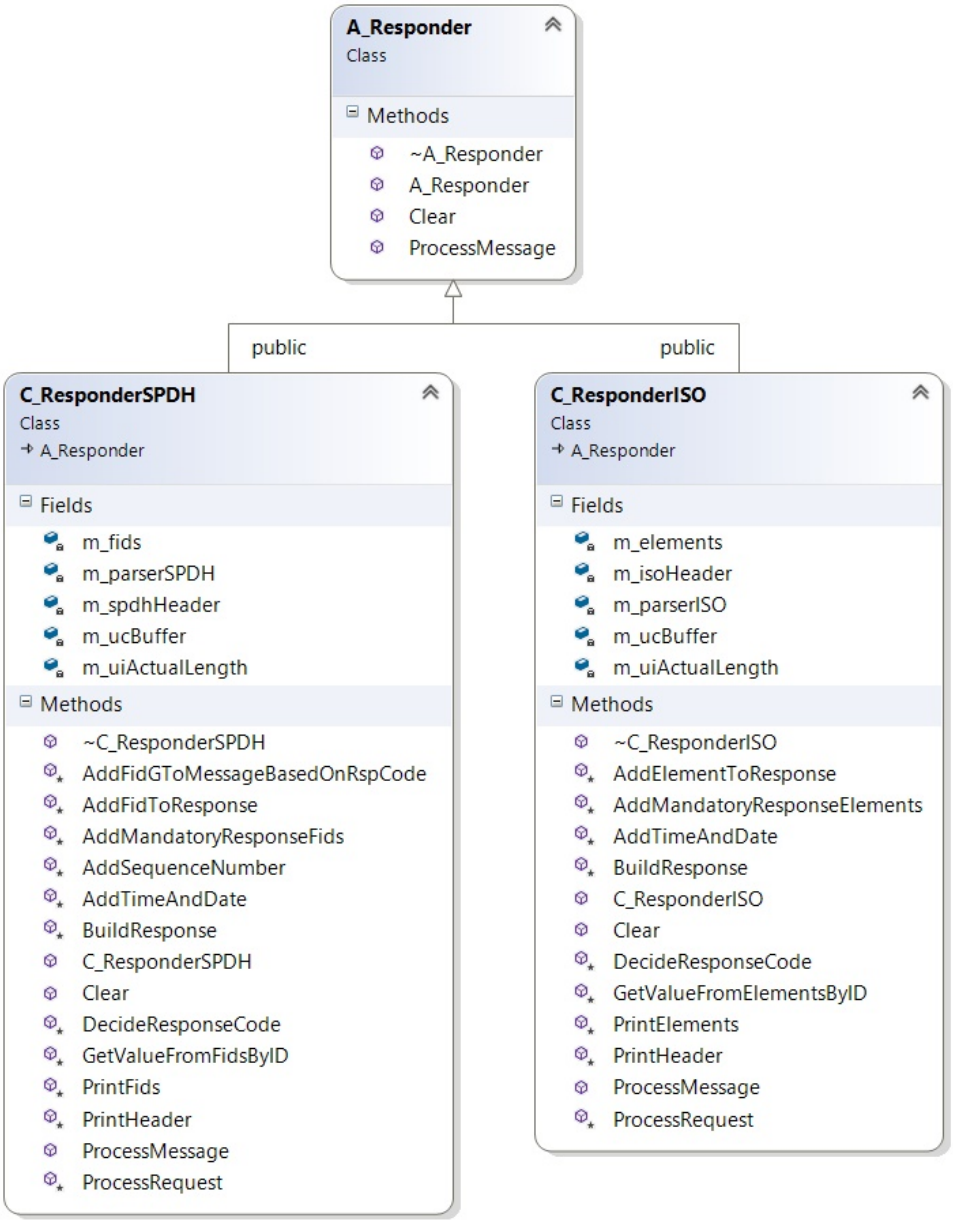
Responder

Nejdůležitějším prvkem celého simulátoru je *Responder* (diagram 4.3). Třída, která zajišťuje zpracování požadavku, ale především má za úkol sestavení odpovědi. Každá instance *Responderu obsahuje*:

- Strukturu popisují hlavičku příchozí zprávy.
- Vector uchovávající data protokolu.

- *Parser* daného protokolu.
- Buffer, který slouží pro vytvoření odpovědi.
- Aktuální velikost bufferu, která je použita jako offset při zpracování odpovědi.

Responder pak používá instanci *Parseru* a jeho public metodu *ParseAndValidateMessage* pro získání příslušného chybového kódu. *Parser* se také v rámci své činnosti postará o naplnění struktury popisující hlavičku a o inicializaci vektoru, který slouží pro uchování dat. *Parser* poté obsahuje instanci daného *Validátoru* a zpracování požadavku je tak delegováno z *Responderu* na *Parser*. *Responder* poté pouze na základě stavu parsování a validace rozhoduje o dalším postupu.



Obrázek 4.3: Diagram modulu Responder.

Mimo sestavení odpovědi zajišťuje *Responder* také výstup, tedy zobrazení příchozích dat, a to jak v čisté podobě, tak jak celá zpráva přišla, tak také ve větším detailu, kde je požadavek rozparsovaný a k datům je přidán i jejich identifikátor a název.

Konfigurace protokolu

Pro správnou funkčnost simulátoru je nutné zajistit, aby měl alespoň základní informace o zpracovávaném protokolu, aby věděl, jaké transakce jsou podporované, a aby měl přehled o podobě a formátu dat, které může očekávat. Protokol a všechna potřebná data jsou shrnuta v konfiguračním souboru ve formátu xml. Tento konfigurační soubor pak musí být přesunut do vnitřní struktury simulátoru. K tomu je určena třída *C_ProtocolData*, která slouží k uložení potřebných dat z konfiguračního souboru, přičemž obsahuje tyto atributy:

- Vector obsahující formát hlavičky.
- Slovník obsahující formát data.
- Vector obsahující podporované transakce a jejich povinné položky.

Vzhledem k faktu, že konfigurační souboru je poměrně rozsáhlý, a ke konfiguraci budeme přistupovat z více míst simulátoru, tak je třída *C_ProtocolData* navržena jako jedináček [14], [2], respektive pro každý zpracovávaný protokol bude existovat pouze jedna instance nesoucí načtenou konfiguraci. Tato instance se pak vytvoří při prvním přístupu k ní samotné. Včetně vytvoření nové instance dojde také ke zpracování konfiguračního souboru. Zde je využit pro parsování a práci s formátem xml modul RapidXml[1]. Při dalším přístupu už je vrácena konkrétní instance třídy *C_ProtocolData* s nastavením daného protokolu.

Komunikace s terminálem

Nedílnou součástí simulátoru je také komunikace s terminál nebo jiným klientem. Vlastní komunikace je implementována prostřednictvím statické třídy *C_CommunicationServer* a je realizována pomocí socketů. Při spuštění simulátoru dojde nejprve k vytvoření socketu na daném portu[2] a následně se čeká na data. I když se takové použití nepředpokládá, tak je komunikační server navržen jako konkurentní, dokáže tedy obsloužit více klientů současně. Vzhledem k faktu, že je simulátor multiplatformní, je nutné rozdělit implementaci komunikačního serveru na dvě části:

- Kompatibilní s operačním systémem Windows. Windows používá mírně odlišný přístup k socketům oproti Unixovým systémům. Další odlišností je zde omezení na maximální počet klientů a obsluha více klientů. Pro Windows je využívána funkce select, kde definujeme množinu socketů z nichž chceme číst, a čekáme na příslušnou událost.
- Komunikační server pracující pod operačním systémem Linux. Kromě odlišnosti při práci se sockety v operačním systému Windows zde není omezen počet klientů a pro příchozí spojení dochází k vytvoření nového procesu.

Bez ohledu na operační systém dojde po navázání spojení k vytvoření nové instance příslušného *Responderu*. Ten následně zpracovává přijatý požadavek a vytváří pro něj odpověď.

[1] RapidXml. Dostupné z: http://rapidxml.sourceforge.net/

[2] Standardně se však používá port 710 pro protokol ISO8583 a 1600 pro protokol SPDH.

Ostatní částí simulátoru

Mimo výše zmíněné součásti obsahuje simulátor ještě *Logger*, který zajišťuje výstup průběhu činnosti simulátoru do souboru. Mimo standardních výstupů jako je přijatý požadavek, odeslaná odpověď a další základní informace, můžeme v logu najít informace o chybových stavech, rozšířené informace o průběhu parsování a validaci dat a další. Tyto informace jsou důležité z hlediska prvotního testování a vývoje simulátoru. Mohou také poskytnou data při případném nestandardním chování a samozřejmě při implementaci nových částí simulátoru[3].

[3] Jako základ Loggeru byl použit framework, který je dostupný na této adrese: http://www.drdobbs.com/cpp/logging-in-c/201804215

Kapitola 5

Testování

Testování je nedílnou součástí každé práce, a před jeho samotným uskutečněním je nutné stanovit vhodnou testovací metodiku. Pro finanční simulátor probíhalo testování v několika krocích a v rámci různých fází celé práce. Bylo nutné otestovat jednak dílčí části simulátoru a jeho funkčnosti, nicméně bylo také nutné otestovat chování simulátoru jako celku, včetně jeho součástí. Samotné testování tedy probíhalo v následujících krocích:

1. Testování dílčích částí simulátoru - tedy *Parser*, *Validator*, celková logika určení response kódu. Tato část testování probíhala po celou dobu současně s vývojem.
2. Testování vůči předpřipraveným zprávám - bude znám požadavek a vůči němu odpověď. Simulátor musí zachovat povinnou strukturu odpovědi. Představuje prvotní testování, tedy otestování jádra simulátoru a jeho funkčnosti.
3. Testování konfigurace včetně jeho částí upravujících dotaz/odpověď.
4. Ověření funkčnosti vůči reálnému platebnímu terminálu.

V rámci této kapitoly se dále zaměříme na testování simulátoru vůči předpřipraveným zprávám, dále na testování součástí umožňující upravovat data odpovědi, také se podíváme na chování simulátoru vůči platebnímu terminálu. Na závěr se pak zaměříme na srovnání simulátoru s používanými aplikacemi a rozebereme některá plánovaná a možná rozšíření simulátoru.

5.1 Testování vůči předpřipraveným zprávám

Testování vůči předpřipraveným zprávám probíhalo za pomoci nástroje Hercules[1], který implementuje TCP klient aplikaci a dovoluje zpracovávat data v hexadecimální podobě. To se nám hodí nejen pro protokol ISO8583, ale také pro protokol SPDH, který obsahuje některé netisknutelné znaky. V této fázi testování bylo za cíl ověřit chování simulátoru především vůči nestandardním požadavkům, jako je například chyba formátu dat, nezaslání některého z povinných polí pro transakci, zaslání neexistující transakce nebo neznámého protokolu. V rámci této části bylo provedeno také testování konfigurace upravující odpověď. Pro testování byla vybrána pro každý protokol jedna zpráva, která byla upravena podle konkrétní testované situace.

Testování vůči předpřipraveným zprávám bude mít vždy následující podobu:

[1] Hercules. Dostupné z: http://www.hw-group.com/products/hercules/index_cz.html

- Originální požadavek[2] - na začátek sady testů bude zmíněn a popsán originální požadavek, který byl dále upravován.
- Popis testu - popis situace, která je testovaná.
- Předpokládaný výsledek - jakým výsledkem by měl test skončit.
- Upravený požadavek - upravený požadavek v hexadecimálním tvaru. Pokud budou data požadavku změněna, daná část bude označena červenou barvou.
- Odpověď vrácená simulátorem. Response kód, a případné další detaily, bude vyznačen modrou barvou.
- Výsledek testu včetně analýzy odpovědi.

Protokol ISO8583

Pro testování protokolu ISO8583 byla vybrána zpráva uvedená v příkladu v kapitole 2.1. Všechny autorizační zprávy budou zobrazeny v hexadecimálním tvaru.

Originální požadavek:
0200303C058020C0000000000000000000012345000024090241110615120051008400344761739001010010D15122211143804489313233343536373853 4E5454455354393837363534333 2

Jedná se o standardní prodej s částkou 123,45. Součástí požadavku je Track 2. Message Type má hodnotu 0200 a Processing Code hodnotu 000000.

Testovací scénář 1

Popis testu: Zaslání validního požadavku.
Předpokládaný výsledek: Schváleno, response code 00.
Upravený požadavek: Požadavek je stejný jako originální zpráva.
Vrácená odpověď:
0210303801000e8000000000000000000012345000024090241110600843939393939393939313430353234303937373030313233343536373 8
Výsledek testu: Simulátor požadavek schválil, zaslal response code 00.

Testovací scénář 2

Popis testu: Neexistující požadavek. Message Type má hodnotu 0289.
Předpokládaný výsledek: Zamítnuto s response kódem 12 (Neznámá transakce).
Upravený požadavek:
0289303C058020C0000000000000000000012345000024090241110615120051008400344761739001010010D15122211143804489313233343536373853 4E5454455354393837363534333 2
Vrácená odpověď:
0299303801000a8000000000000000000012345000024090241110600843939393939393939313430353132313233343536373 8
Výsledek testu: Požadavek je zamítnut s návratovým kódem 12.

[2] Směrovací informace a TPDU jsou pro všechny požadavky a odpovědi v rámci textu dále vynechány. Z hlediska sestavení validní odpovědi a práce simulátoru nejsou tyto informace podstatné.

Testovací scénář 3

Popis testu: Nevalidní data, v poli pro částku je zaslána hodnota 0x0000000ABCDE.
Předpokládaný výsledek: Zamítnuto, chyba formátu, response kód 30.
Upravený požadavek:
0200303C058020C0000000000000000ABCDE000024090241110615120051008400344761739001010010D151222111438044893132333435363738534E5454455354393837363534333 2
Vrácená odpověď:
0210303801000a80000000000000000000abcde000024090241110600843939393939393939313430353330313233343536373 8
Výsledek testu: Zamítnuto, response kód 30.

Testovací scénář 4

Popis testu: Nastaveno RSPCODEBYAMOUNT[3], zaslaná částka 123,45.
Předpokládaný výsledek: Hodnota response kódu podle posledních dvou číslic částky, tedy 45.
Upravený požadavek: Ne, požadavek je stejný.
Vrácená odpověď:
0210303801000e800000000000000000001234500002409024111060084393939393939393931343035323038323438343531323334354363738
Výsledek testu: Návratový kód je nastaven na 45, transakce by byla zamítnutá.

Testovací scénář 5

Popis testu: Nastaveno USESPECIFIC a data RSPCODE na hodnotu 51 a AUTHCODE na hodnotu 120556.
Předpokládaný výsledek: Hodnota Authentication ID Response nastavena na 120556 a návratový kód na 51.
Upravený požadavek: Požadavek je stejný jako originál.
Vrácená odpověď:
0210303801000e800000000000000000001234500002409024111060084393939393939393931343035313230353536353131323334354363738
Výsledek testu: Hodnoty jsou zaslané korektně a ve správném pořadí. Zelenou barvou je označeno Authentication ID Response.

Pokud simulátor přijme neznámý požadavek nebo se mu nepovede zpracovat hlavičku dat, tak nezasílá žádnou odpověď. V případě protokolu ISO8583 představuje neznámou zprávu například jakýkoli SPDH požadavek.

Protokol SPDH

Pro protokol SPDH byla zvolena, stejně jako u protokolu ISO8583, testovací zpráva z příkladu uvedeném v kapitole 2.2.Všechny požadavky budou uvedeny v hexadecimálním tvaru. Všechny odpovědi pak budou zobrazeny v ASCII formátu, kde znak 0x1C bude nahrazen znakem tečka (".").

[3]Uživatelem definovaná data odpovědi jsou předmětem sekce 3.2.5. V textu dále je považování slovní spojení "nastavení RSPCODEBYAMOUNT" nebo "nastavení USESPECIFIC" za definování těchto položek na hodnotu Y v rámci příslušného konfiguračního souboru pro daný protokol.

Originální požadavek:
392e3233504f532d5445524d494e414c2d303536393837363534313631323331323333539353
9464f30303030303030301c714d313233343536373839303132333435363738393d30313038
0a303f1C423132333435

Požadavek obsahuje Message Type F a Transaction Code 00, jedná se tedy o prodej na částku 123,45.

Testovací scénář 1

Popis testu: Zaslání validní zprávy.
Předpokládaný výsledek: Response text "Approved" a response kód 001.
Upravený požadavek: Požadavek není nijak upraven.
Vrácená odpověď:
9.23POS-TERMINAL-056987654170513175511F000000001.B12345.F23141590.qM1234567
890123456789=01080A0?.h0010010011.gApproved
Výsledek testu: Transakce je schválena, odpověď obsahuje daný response text a response kód.

Testovací scénář 2

Popis testu: Neznámá transakce. Message Type má hodnotu V (0x56).
Předpokládaný výsledek: Zamítnutá transakces s response kódem 052 a response textem "Invalid transaction".
Upravený požadavek:
392e3233504f532d5445524d494e414c2d303536393837363534313631323331323333539353
9564f30303030303030301c714d313233343536373839303132333435363738393d30313038
0a303f1C423132333435
Vrácená odpověď:
9.23POS-TERMINAL-056987654170513181846V000000052.B12345.h0010010011
.gInvalid transaction
Výsledek testu: Transakce je zamítnuta s návratovým kódem 053.

Testovací scénář 3

Popis testu: Nevalidní data. Místo částky je zaslán textový řetězec "AHOJ".
Předpokládaný výsledek: Návratový kód 801 společně s response textem "Invalid data".
Upravený požadavek:
392e3233504f532d5445524d494e414c2d303536393837363534313631323331323333539353
9464f30303030303030301c714d313233343536373839303132333435363738393d30313038
0a303f1C4241484f4a
Vrácená odpověď:
9.23POS-TERMINAL-056987654170513184420F000000801.BAHOJ.qM123456789012345678
9=01080A0?.h0010010011.gInvalid data
Výsledek testu: Response kód má hodnotu 801, transakce je zamítnuta.

Testovací scénář 4

Popis testu: Chybějící povinná položka pro transakci prodej. Chybí částka.
Předpokládaný výsledek: Zamítnutá transakce kvůli chybě formátu. Response kód 800

a response text "Format error".
Upravený požadavek:
392e3233504f532d5445524d494e414c2d303536393837363534313631323331323333539353
9464f303030303030301c714d313233343536373839303132333435363738393d30313038
0a303f
Vrácená odpověď:
9.23POS-TERMINAL-056987654170513185246F000000800.qM1234567890123456789=0108
0A0?.h0010010011.gFormat error
Výsledek testu: Transakce je zamítnutá, odpověď obsahuje návratový kód 800.

Testovací scénář 5

Popis testu: Nastaveno RSPCODEBYAMOUNT pro zaslanou částku 123,45.
Předpokládaný výsledek: Response kód podle posledních tří číslic. Hodnota tedy musí být 345.
Upravený požadavek: Požadavek není neupraven.
Vrácená odpověď:
9.23POS-TERMINAL-056987654170513201504F000000345.B12345.F12954882.qM12345678
90123456789=01080A0?.h0010010011.gApproved
Výsledek testu: Response kód má správnou hodnotu 345, nicméně součástí je i response text, který byl určen podle chybového kódu uvnitř simulátoru.

Testovací scénář 6

Popis testu: Byla nastavena hodnota USESPECIFIC, RSPCODE na 069, RSPTEXT na "AHOJ SVETE"a APPROVALCODE na hodnotu 85632010. Ostatní hodnoty byly prázdné.
Předpokládaný výsledek: Odpověď musí obsahovat návratový kód 069, response text musí mít hodnotu "AHOJ SVETE" a Approval Code musí být nastaven na 85632010.
Upravený požadavek: Požadavek nebyl nijak upraven.
Vrácená odpověď:
9.23POS-TERMINAL-056987654170513202230F000000069.B12345.F85632010.qM12345678
90123456789=01080A0?.h0010010011.gAHOJ SVETE
Výsledek testu: Návratový kód má hodnotu 069, ostatní upravené položky jsou také součástí odpovědi.

Stejně jako u protokolu ISO8583, pokud přijde pro protokol SPDH neznámý požadavek nebo nebude možné zpracovat hlavičku požadavku, tak pro takovou zprávu není odeslána žádná odpověď.

5.2 Testování vůči terminálu

Povinnou součástí této práce bylo také otestovat výsledný simulátor vůči reálnému platebnímu terminálu. Základním požadavkem pak je, aby terminál správně reagoval na zaslané odpovědi. Tedy aby simulátor svým chováním neodporoval chování autorizace. Metodika testování bude odlišná od té použité v přecházející sekci 5.1. Z terminálu totiž očekáváme většinou požadavky formálně správné a tedy se testuje spíše chování aplikace a zkoumaného protokolu. Navíc možnost vygenerovat z aplikace nahrané v terminálu chybný požadavek

je pouze její úprava a taková úprava je samozřejmě nežádoucí. Testování simulátoru vůči terminálu bude mít následující formát:

- Popis testu - popis zkoumané situace včetně transakce, jaká je z terminálu zaslána.
- Předpokládaný výsledek - jak by měl simulátor reagovat na požadavek.
- Vygenerovaná odpověď - odpověď zaslaná simulátorem na terminál, modrou barvou je zvýrazněn návratový kód.
- Výsledek transakce na straně terminálu.

```
################################################################################

Bytes received: 136

Raw request:
9.11S1APHPC6          170515155935FU00000001□B15000□F27656725□q;5470552704029420=18
11201168990600000?□h0010010011□gApproved□

SPDH header data
Device Type -> 9.
Tran number -> 11
Terminal ID -> S1APHPC6
Employee ID ->
Current date -> 170515
Current time -> 155935
Msg type -> F
Msg subtype -> U
Tran code -> 00
Processing flag 1 -> 0
Processing flag 2 -> 0
Processing flag 3 -> 0
Rsp code -> 001

SPDH fids data
Fid ID -> B; Fid name -> Amount 1; Fid value -> 15000
Fid ID -> F; Fid name -> Approvel Code; Fid value -> 27656725
Fid ID -> q; Fid name -> Track 2/Customer; Fid value -> ;5470552704029420=18112011689906000000?
Fid ID -> h; Fid name -> Sequence Number; Fid value -> 0010010011
Fid ID -> g; Fid name -> Response Display; Fid value -> Approved□


Raw response:
9.11S1APHPC6          170515155959FU00000001□B15000□F27656725□q;5470552704029420=18
11201168990600000?□h0010010011□gApproved□
Bytes sent: 136
```

Obrázek 5.1: Ukázka výstupu simulátoru.

Testovací data pro následující část byla získána pomocí logů z aplikace běžící na platebním terminálu a ze zachycené nešifrované komunikace mezi terminálem a autorizací prostřednictvím nástrojem Wireshark[4]. Stejné transakce pak byly zaslány a ověřeny vůči

[4]Nástroj Wireshark. Dostupné z: https://www.wireshark.org/

vyvíjenému simulátoru. Jednotlivé části komunikace (navázání spojení, zaslání data, ukončení spojení) mezi terminálem a simulátorem pak byly ověřeny na základě zachycené komunikace nástrojem Wireshark.

Protokol ISO8583

Pro otestování chování simulátoru vůči terminálu pro protokol ISO8583 byly vybrány základní transakce test linky, prodej a návrat. Prodej byl ještě navíc proveden i s uživatelsky definovanými hodnotami.

Testovací scénář 1

Popis testu: Test linky - transakce vykonávaná pro ověření dostupnosti linky.
Předpokládaný výsledek: Simulátor běží na nastaveném portu 710 pro protokol ISO8583. Předpokládaný výsledek je schválená transakce.
Vygenerovaná odpověď :
`08102020010002800000990000000062000730304 9534f5048504336`
Výsledek transakce: Transakce byla schválena, test linky proběhl úspěšně.

Testovací scénář 2

Popis testu: Prodej na částku 150,00 Kč.
Předpokládaný výsledek: Schváleno.
Vygenerovaná odpověď :
`0210303801000e800000000000000000015000000631548060515000739393939393939393135303533323034313530304 9534f5048504336`
Výsledek transakce: Transakce byla schválena.

Testovací scénář 3

Popis testu: Návrat na částku 500,00 Kč.
Předpokládaný výsledek: Transakce by měla být schválena.
Vygenerovaná odpověď :
`0230303801000e800000200000000000050000000065154851051500073939393939393939313530353134363730323030 49534f5048504336`
Výsledek transakce: Transakce byla schválena.

Testovací scénář 4

Popis testu: Transakce prodej na částku 100,43 Kč a zapnutým nastavením RSPCODEBYAMOUNT.
Předpokládaný výsledek: Terminál by měl transakci zamítnout a zobrazit zprávu "Zadržte kartu - kráděž".
Vygenerovaná odpověď :
`0210303801000e800000000000000000010043000067154928051500073939393939393939313530353337313530333433 49534f5048504336`
Výsledek transakce: Transakce byla zamítnuta a byla zobrazena daná zpráva.

Testovací scénář 5

Popis testu: Transakce prodej s nastavenými hodnotami USESPECIFIC a RSPCODE na 51.
Předpokládaný výsledek: Zamítnutá transakce s textem "Nedostatek financí".
Vygenerovaná odpověď :
`0210303801000e800000000000000000010000000681551140515000739393939393939393135303534393434383135314953 4f5048504336`
Výsledek transakce: Transakce byla zamítnuta.

Protokol SPDH

V rámci testování simulátoru a protokolu SPDH byla vybrány transakce test linky, prodej a storno. Prodej byl následně zopakován s uživatelsky definovanými hodnotami.

Testovací scénář 1

Popis testu: Test linky, simulátor poslouchal na portu 1600.
Předpokládaný výsledek: Schváleno s návratovým kódem 007.
Vygenerovaná odpověď :
`9.00S1APHPC6 170515155848AU95000007.F84707040.h0010010011.gApproved`
Výsledek transakce: Terminál obdržel schvalovací odpověď s response kódem 007.

Testovací scénář 2

Popis testu: Prodej na částku 150,00 Eur.
Předpokládaný výsledek: Schváleno.
Vygenerovaná odpověď :
`9.12S1APHPC6 170515160117FU00000001.B15000.F35986172.q;5470552704029420=1811 2011689906000000?.h0010010011.gApproved`
Výsledek transakce: Transakce byla schválena.

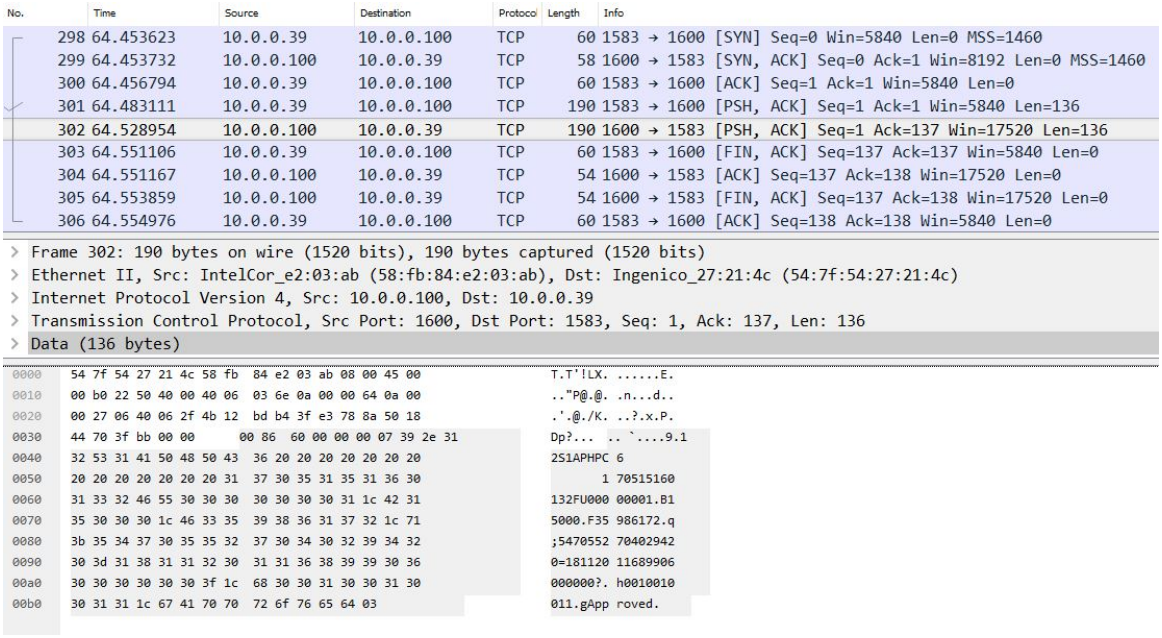
Testovací scénář 3

Popis testu: Storno předcházející transakce. Message SubType má správně hodnotu U (User reversal).
Předpokládaný výsledek: Schváleno, s návratovým kódem 001.
Vygenerovaná odpověď :
`9.12S1APHPC6 170515160132FU00000001.B15000.F35986172.q;5470552704029420=1811 2011689906000000?.h0010010011.gApproved`
Výsledek transakce: Transakce byla schválena s návratovám kódem 001.

Obrázek 5.2 pak zachycuje komunikaci mezi terminálem s IP 10.0.0.39 a simulátorem s IP 10.0.0.100 v rámci transakce popisované v testovacím scénáři 3. V horní části obrázku je zobrazen průběh celé komunikace. Ve spodní části je pak zvýrazněna odpověď, kde můžeme vidět jednotlivé části odpovědi a také, že byla transakce schválena.

Testovací scénář 4

Popis testu: Prodej na částku 100,69 Eur s nastavenou hodnotou RSPCODEBYAMOUNT.
Předpokládaný výsledek: Transakce bude zamítnuta s návratovým kódem 069.



Obrázek 5.2: Zachycená komunikace s odpovědí na transakci z testovacího scénáře 3.

Vygenerovaná odpověď :
`9.17S1APHPC6 170515185417FO00000069.B10069.F59070341.q;5470552704029420=1811 2011689906000000?.h0010010011.gApproved`
Výsledek transakce: Transakce byla zamítnuta s příslušným response kódem.

Testovací scénář 5

Popis testu: Prodej s nastavenými hodnotami USESPECIFIC, RSPCODE na 111 a RSPTEXT obsahoval řetězec "TEST TERMINAL TEXT".
Předpokládaný výsledek: Zamítnuto s návratovým kódem 111 a textem "TEST TERMINAL TEXT".
Vygenerovaná odpověď :
`9.16S1APHPC6 170515160705FO00000111.B50000.F85632010.q;5470552704029420=1811 2011689906000000?.h0010010011.gTEST TERMINAL TEXT`
Výsledek transakce: Transakce byla zamítnuta s response kódem 111 a byl zobrazen příslušný text.

Simulátor generuje pro oba implementované protokoly validní odpovědi, které terminál dokáže zpracovat a správně interpretovat.

5.3 Rozšiřitelnost simulátoru

Důležitou součástí vyvíjeného simulátoru je především jeho rozšiřitelnost o další protokoly. Dostupné nástroje nahrazující autorizaci neposkytují žádnou možnost rozšíření, jsou určené pouze pro daný protokol a veškeré informace o podobě zpracovávaného protokolu obsahují uvnitř a neumožňují je žádným způsobem modifikovat. V praxi pak může nastat situace, že

libovolná autorizace používá pro komunikace vlastní upravený protokol, který je většinou založený na protokolech ISO8583 a SPDH.

Taková úprava může být pouze triviální a může spočívat pouze v redefinici některých polí zprávy nebo v rozšíření možných hodnot některé z položek. Například může autorizace využívat pro komunikaci protokol SPDH, který ovšem nepoužívá hodnotu F v poli Message Type, ale používá vlastní hodnotu O a redefinuje pole U pro vlastní použití. V takovém případě stačí upravit konfigurační soubor, zavést novou hodnotu pro Message Type a podle specifikace autorizace upravit pole U. Taková změna se obejde bez úpravy implementace, stačí pouze upravit daný konfigurační soubor.

Nicméně se může jednat i o rozsáhlejší úpravu protokolu. Není výjimkou, kdy autorizace přidávají další vlastní hlavičku nebo kompletně redefinují celý protokol, kdy je zachována pouze struktura protokolu, ale data už jsou velmi odlišná. V takovém případě je již nutné provést kompletní analýzu konkrétního protokolu, upravit a přidat jeho zcela novou implementaci včetně konfiguračního souboru a celé logiky jeho zpracování.

5.4 Porovnání simulátoru s dostupnými programy

V praxi se ve společnosti Sonet pro testování činnosti aplikace na platebním terminálu využívají dvě oddělená prostředí, ISO SmartHost a SPDH SmartHost, každé pro jeden protokol. Obě zmíněná prostředí fungují na stejném principu, první zmíněné zpracovává protokol ISO8583 a druhé zmíněné pak protokol SPDH. Srovnání obou prostředí s vyvíjeným simulátorem je součástí tabulky 5.1. Používaná prostředí jsou stejně jako simulátor konkurentní, tedy schopné zpracovat více požadavků v jednom okamžiku a umožňují uživateli také definovat vybrané položky odpovědi v rámci svého nastavení.

	Simulátor	**ISO/SPDH SmartHost**
Konkurentní	Ano	Ano
Rozšiřitelný	Ano	Ne
Uživatelem definovaná data odpovědi	Ano	Ano
Multiplatformní	Ano	Ne
Detail transakce	Ano	Ne
GUI	Ne	Ano

Tabulka 5.1: Porovnání simulátoru a používaných prostředí.

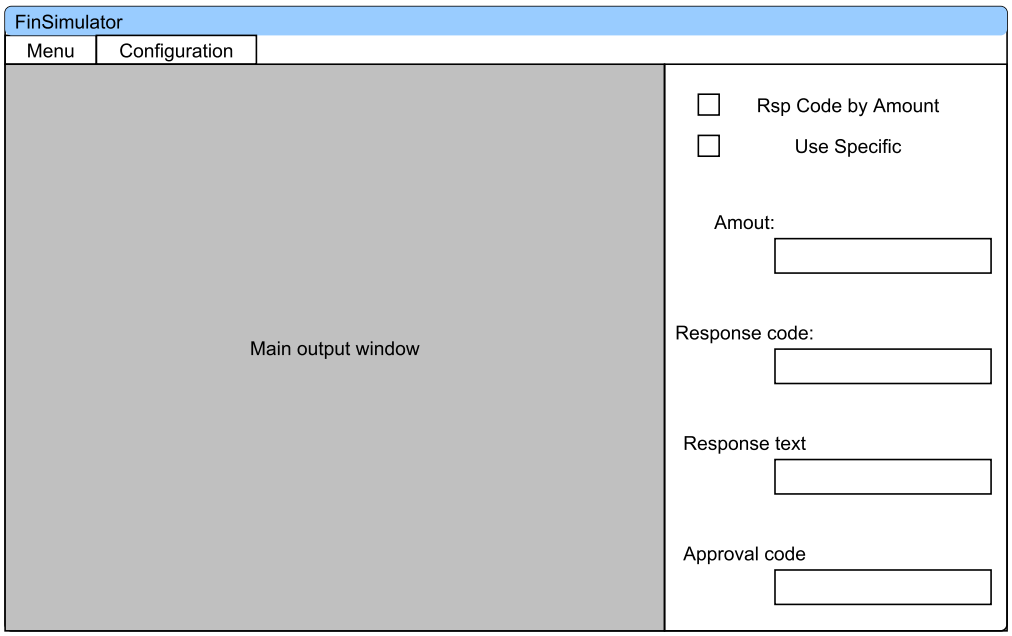
Nicméně v dalších aspektech už se simulátor a používaná prostředí liší. Ať už se jedná o rozšiřitelnost prostředí o další protokoly, úpravu stávajících protokolů nebo o zobrazení detailu transakce. ISO a SPDH SmartHost umí zobrazovat pouze některé vybrané detaily, kdežto simulátor umí zobrazit celý zpracovaný požadavek včetně identifikátorů a krátkého popisu pole. Další výhodou simulátoru pak může být také to, že je podporován na operačních systémech Windows a Linux. Naopak nevýhodou vůči používaným prostředím je chybějící grafické uživatelské rozhraní, které může uživatelům ulehčit a zpříjemnit práci s danou aplikací. ISO a SPDH SmartHost však již nejsou delší dobu podporována a dále vyvíjena. I toto je jedním z důvodů, proč je potřeba nový simulátor pro finanční protokoly.

5.5 Plánovaná rozšíření

Simulátor je v současné době na začátku vývoje, kdy je k dispozici jeho první verze. Implementace dalších součástí simulátoru bude pokračovat i v budoucnu a vývoj nových částí bude zaměřen především na současné nedostatky a požadavky společnosti Sonet.

Grafické uživatelské rozhraní

Největším současným nedostatkem simulátoru je jistě chybějící grafické uživatelské rozhraní, které by zpříjemnilo a usnadnilo využití simulátoru. Simulátor má charakter spíše pasivního prostředí, kde není ve většině případů interakce mezi ním a uživatelem aktivní, simulátor v podstatě uživateli pouze zobrazuje přijatá a odeslaná data. Vzhledem k charakteru simulátoru tedy není GUI nezbytné, nicméně je nesporné, že jeho použití je pro uživatele příjemnější a pohodlnější. Takové uživatelské rozhraní, pak může mít podobu zobrazenou na obrázku 5.3.



Obrázek 5.3: Ukázka rozložení prvků grafické uživatelského rozhraní.

Hlavní část bude tvořit výstup pro zpracovávaná data, tedy pro aktuálně příchozí požadavek a odpověď. Vpravo pak bude možnost nastavovat nejčastější uživatelské hodnoty odpovědi, kde každý protokol bude mít vlastní rozložení prvků. V rámci menu Configuration pak bude možné nastavit i další hodnoty odpovědi.

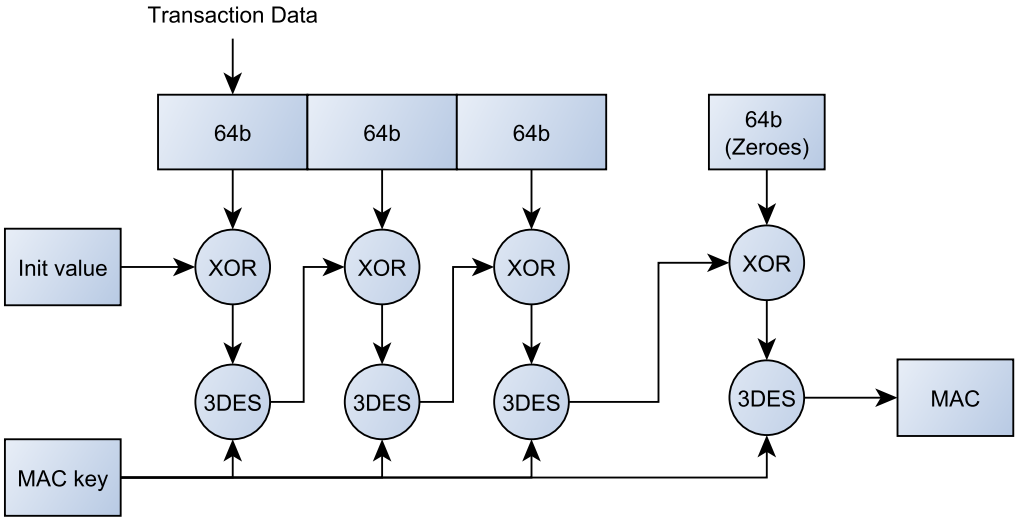
Použití grafického uživatelského rozhraní se pak předpokládá pouze na operačním systému Windows, na kterém probíhá vývoj i testování ve společnosti Sonet. Pro implementaci pak bude použit framework Qt[5].

[5]Framework Qt. Dostupné z: https://www.qt.io/product/

Výpočet MACu a PIN bloku

Velmi významným rozšířením simulátoru může být schopnost ověřit a spočítat MAC a ověřit hodnotu PIN blok. V obou případech se využívá pro správu klíčů princip Master/Session, kdy autorizace i terminál mají stejný Master klíče (můžeme ho také nazvat jako šifrovací klíč) a na základě tohoto klíče jsou pak zašifrovaná náhodně vygenerovaná data, která představují Session klíč (pracovní klíč). Session klíč je pak generován pro každou transakci a využívá se pro generování MAC hodnoty nebo pro šifrování PIN bloku.

Hodnota MAC se používá pro ověření, zda nebyla data nějakým způsobem porušena nebo přepsána. Pokud se na straně autorizace nepodaří ověřit MAC, je transakce samozřejmě zamítnuta a je vrácen příslušný response kód. Pro výpočet hodnoty MAC a jeho ověření na straně simulátoru musíme znát pole, z kterých se počítá, ale také musíme znát Session klíč použitý pro vygenerování MAC. Hodnota MAC se pak počítá podle normy ANSI X9[8]. Možný průběh výpočtu je naznačen na diagramu 5.4.



Obrázek 5.4: Výpočet MAC.

Nad vstupním bufferem je postupně několikrát provedena operace XOR a výpočet blokové šifry DES nebo 3DES. Výsledná podoba algoritmu je pak závislá na konkrétní implementaci. V praxi se mohou šifry DES a 3DES i kombinovat. Na konci algoritmu ale vždy získáme hodnotu MAC, kterou můžeme porovnat s hodnotu přijatou v rámci přenášené zprávy.

V rámci PIN bloku se pak zasílá zašifrovaná hodnota PINu. Pro jeho ověření musíme opět znát Session klíč použitý pro zašifrování PIN blok, ale musíme znát také správný PIN, abychom mohli vygenerovat PIN blok a ověřit vůči příchozímu. Pro generování PIN bloku se používá algoritmus dostupný v rámci normy ISO9564-1[9] a využívá se formát 0, kde se PIN blok skládá ze dvou části:

- Předpřipravený PIN ve formátu CLXXXXXXXXXXXXXX, kde C představuje hodnotu 0, L počet číslici PINu, prvních L hodnot X je nahrazeno PINem a zbývající část je nahrazena libovolnou hodnotou, která závisí na konkrétní implementaci. Jedna položka představuje jeden nibble, celé pole pak osm bytů.

- Předpřipravený PAN ve formátu ZZZZPPPPPPPPPPPP, kde Z představuje hodnotu 0 a P dvanáct posledních číslic PANu v nibble tvaru. Celé pole má velikost osm bytů.

Nad takto připravenými hodnotami je provedena operace XOR a výsledek, pokud má být zaslán ve zprávě, je poté zašifrován pracovním klíčem. Pokud má dojít k ověření, tak se hodnota spočteného PIN bloku porovná s příchozím dešifrovaným PIN blokem nebo se provede za pomoci opačného postupu získání čisté podoby PINu a porovnání jeho hodnoty se skutečným PINem.

Rozšíření simulátoru umožňující ověřit hodnotu MAC a PIN bloku může být realizováno jako jednorázové ověření, kdy uživatel zadá prostřednictvím nastavení požadovaná pole a simulátor pro aktuálně příchozí transakci provede dané ověření. Pro další transakci by bylo nutné hodnoty opět upravit. Toto rozšíření však může být realizováno i jako komplexnější řešení, kdy bude mít simulátor k dispozici používané Master klíče a bude sám generovat pro terminál pracovní klíče. V tomto případě by bylo možné na úrovni simulátoru ověřovat za určitých okolností (simulátor zná data, z kterých se počítá MAC, zná PIN) všechny příchozí transakce. Takové řešení je ovšem výrazně složitější a vyžaduje navržení správy klíčů na úrovni simulátoru.

Historie transakcí

Dalším užitečným rozšířením může být ukládání zpracovaných transakcí a jejich zpětné prohlížení. V rámci tohoto rozšíření je nutné zvolit vhodný formát pro ukládání transakcí včetně formátu, v jakém budou transakce ukládány. Zda se uloží pouze přijatý požadavek, který se při zpětném načtení rozparsuje, proběhne validace a vytvoření odpovědi, nebo zda se uloží požadavek včetně odpovědi a při načtení se data pouze rozparsují, anebo zda se uloží kompletní vzorek dat, tedy požadavek v čisté podobě, zpracovaný požadavek a odpověď, a při čtení záznamu se provede pouze jejich zobrazení. Toto rozšíření ovšem vyžaduje přítomnost grafického uživatelského rozhraní.

Kapitola 6

Závěr

Cílem této práce bylo vytvořit simulátor pro finanční protokoly ISO8583 a SPDH, který bude snadno rozšiřitelný a bude umožňovat definovat uživatelem vybraná data odpovědi. Takové prostředí je z hlediska vývoje a testování aplikace na platebním terminálu nezbytné. V rámci teoretické částí práce bylo nutné nejdříve vymezit a definovat některé pojmy z oblasti bankovnictví a pojmů týkajících se terminálu a platebních karet. Z tohoto hlediska bylo také důležité definovat pojem autorizace a simulátor a jejich rozdíly. Tedy, že simulátor nemá v žádném případě sloužit jako náhrada za autorizaci, ale jeho primárním účelem je pomoci při vývoji a nasazování nových protokolů, odhalení chyb při zpracování protokolů stávajících a umožnění simulovat nejrůznější chybové stavy. Nezbytnou součástí byla také analýza výše zmíněných protokolů, zjištění jejich formátu, získání informací o tom, jak by se měly tyto protokoly chovat, ale také určení jejich společných a rozdílných rysů. Na základě těchto informací pak vznikl návrh simulátoru a byly vytvořeny potřebné konfigurační soubory pro zpracování protokolů.

V praktické částí práce jsem poté implementoval v jazyce C++ navržený simulátor. Simulátor je rozdělen do několika logických celků, z nichž nejdůležitějším je *Responder*, který zajišťuje za pomoci *Parseru* zpracování příchozího požadavku. Dále se také stará o sestavení validní odpovědi. Validaci požadavku pak zajišťuje na základě informací získaných prostřednictvím konfiguračních souborů *Validátor*.

Testování probíhalo v několik fázích. Jednak za pomoci nástroje Hercules na předpřipravených zprávách, kdy jsem ověřil základní funkčnost simulátoru, především pro nejrůznější chybové stavy. Další částí testování pak bylo testování vůči reálnému terminálu, kde bylo nutné ověřit, zda simulátor s terminálem správně komunikují, ale také, zda simulátor zasílá validní odpovědi a zda je terminál schopen tyto odpovědi zpracovat a správně interpretovat. Součástí testování bylo také krátké srovnání implementovaného simulátoru s dosud používanými prostředími, ty sice na rozdíl od simulátoru disponují grafickým uživatelským rozhraním, nicméně jejich funkčnost nelze nijak rozšířit, neexistuje k nim žádná podpora a jejich další vývoj již neprobíhá.

Simulátor v žádném případě nemůže nahradit konkrétní autorizaci, vždy se bude jednat jen o jednoduchý nástroj sloužící pro prvotní testování nového protokolu a testování již zavedených standardních protokolů. Vývoj plnohodnotné autorizace ke každému novému protokolu by byl finančně a časově velice náročný. Vývoj simulátoru bude pokračovat i nadále, ať už se bude jednat o implementaci výše zmíněných rozšíření nebo o přidání nových protokolů.

Literatura

[1] ACI: *Standard POS Device Message Specifications Manual*. 2015, release 6.0 version 10.

[2] Alexandrescu, A.: *Moderní programování v C++: návrhové vzory a generické programování v praxi*. Computer Press, 2004, ISBN 80-251-0370-6.

[3] EMV Integrated Circuit Card Specifications for Payment Systems: *Book 1: Application Independent ICC to Terminal Interface Requirements*. 2011, version 4.3.

[4] Česká národní banka: *Druhy a typy platebních karet*. [Online; navštíveno 29.12.2016]. URL http://www.cnb.cz/miranda2/export/sites/www.cnb.cz/cs/statistika/predpisy_CNB_statistika/predpisy_menove_bank_stat/vykazy_metodika_2005/download/05_6_BA0092.pdf

[5] Hendy, M.: *Multi-application Smart Cards: Technology and Applications*. Cambridge University Press, 2007, ISBN 9780521873840.

[6] Hypercom: *Message specifiaction*. 2005, version 3.44.

[7] ISO: *ISO/IEC 7813:2006* . 2006, iSO/IEC JTC 1/SC 17. URL http://www.iso.org/iso/catalogue_detail.htm?csnumber=43317

[8] ISO 8583: *ISO 8583-1:2003(en)*. 2003, iSO/TC 68/SC 7. URL https://www.iso.org/obp/ui/#iso:std:iso:8583:-1:ed-1:v1:en

[9] ISO 9564-1: *ISO 9564-1:2011(en)*. 2011, iSO/TC 68/SC 2. URL https://www.iso.org/obp/ui/#iso:std:iso:9564:-1:ed-3:v1:en

[10] Juřík, P.: *Encyklopedie platebních karet*. Grada Publishing, a. s., 2003, ISBN 80-247-0685-7.

[11] Komerční banka, a.s.: *Pokyny pro provádění transakcí platebními kartami*. [Online; navštíveno 16.12.2016]. URL https://www.kb.cz/file/cs/o-bance/dokumenty-ke-stazeni/kb-20111201-pokyny-pro-provadeni-transakci-platebnimi-kartami.pdf?3047d0d71768073358dfb06370d7650c

[12] mesec.cz: *Bezkontaktni platby*. [Online; navštíveno 7.1.2017]. URL http://www.mesec.cz/bankovni-ucty/platebni-karty/bezkontaktni-platby/pruvodce/

[13] OMNeT++: *Simulation Manual*. 2016, version 5.0. URL https://omnetpp.org/

[14] Pecinovský, R.: *Návrhové vzoy*. Computer Press, 2013, ISBN 978-80-251-1582-4.

[15] penize.cz: *Platební karty a jejich druhy*. [Online; navštíveno 16.12.2016].
URL http://www.penize.cz/80265-platebni-karty-a-jejich-druhy

[16] Schlossberger, O.: *Elektronické platební prostředky*. Profess Consulting, 1997, ISBN 978-80-85235-51-7.

[17] Sdružení pro bankovní karty: *Statistiky*. [Online; navštíveno 29.12.2016].
URL http://www.bankovnikarty.cz/pages/czech/profil_statistiky.html

[18] Sdružení pro bankovní karty: *Vydání karet v ČR*. [Online; navštíveno 29.12.2016].
URL http://www.bankovnikarty.cz/pages/czech/profil_karty.html

[19] Sekerka, B.: *Banky a bankovní produkty*. Profess Consulting, 1997, ISBN 978-80-85235-51-7.

[20] Smith, I.: *What is EMV?* Hypercom.

[21] Sonet, s.r.o.: *Slovník pojmů*. 2015, release 1.0 version 1.

Příloha A

Obsah CD

Obsah CD:

- `Doc` - dokumentace ve formě html stránky vygenerovaná programem Doxygen[1].
- `Release` - zkompilovaný zdrojový kód simulátoru včetně konfiguračních souborů.
- `Sources` - adresář obsahující zdrojový kód, konfigurační soubory, nástroj Hercules a projekt pro Microsoft Visual Studio.
- `TextSources` - zdrojové soubory, obrázky k textu diplomové práce.
- `xvymla01_DP.pdf` - výsledná práce ve formátu pdf.
- `Readme.txt` - manuál k simulátoru

[1] Doxygen. Dostupné z: http://www.stack.nl/~dimitri/doxygen/

Příloha B

Manuál

Simulátor nevyžaduje žádný speciální dodatečný software. Je nutné mít k dispozici pouze konfigurační soubory. Před spuštěním je pak žádoucí upravit konfigurační soubor `config.ini` a zvolit vybraný protokol a port. Po spuštění (soubor `inSimulator`) simulátor nastaví komunikační server, vytvoří socket na příslušném portu a čeká na příchozí požadavek. Mezi příchozí požadavky je pak možné upravovat ini soubor pro daný protokol (`ISO8583config.ini` nebo `SPDHconfig.ini`) a měnit tak data odpovědi. V případě spouštění pod operačním systémem Linux je vhodné využít terminálovou konzoli. Při manipulaci se simulátorem a jeho konfiguračními soubory je nutné dodržet předem danou adresářovou strukturu - simulátor musí být vždy ve stejném adresáři jako je adresář ConfigFiles obsahující konfigurační soubory.

Spuštění simulátoru

- Windows:
  1. Nastavit příslušný protokol a port v souboru `ConfigFiles/config.ini`.
  2. Spustit soubor `finSimulator.exe`.
- Linux:
  1. Nastavit příslušný protokol a port v souboru `ConfigFiles/config.ini`.
  2. Spustit terminálovou konzoli.
  3. Přepnout se do adresáře obsahujícího simulátor.
  4. Spustit simulátor příkazem `./finSimulator`.

Konfigurační soubory

- `ConfigFiles/config.ini`
- `ConfigFiles/ISO8583protoConfig.xml`
- `ConfigFiles/ISO8583config.ini`
- `ConfigFiles/SPDHprotoConfig.xml`
- `ConfigFiles/SPDHconfig.ini`

Podporované protokoly

- `SPDH`
- `ISO8583`

Simulátor byl otestován na operačních systémech Windows 7 Home Premium 32bit, Windows 8 64bit, Windows 10 Home 64bit, Fedora Workstation 22 64bit, Ubuntu 15.10 64bit.

V případě potřeby je součástí CD a odevzdaného archivu nachystaný projekt pro Microsoft Visual Studio.

Příloha C

Konfigrační soubory

Simulátor v současné době vyžaduje pro správnou funkčnost minimálně pět konfiguračních souborů. Jeden, který je obecný, a určuje pro který protokol byl simulátor spuštěn, a na jakém protokolu má poslouchat. Další konfigurační soubory jsou již specifické pro každý protokol, kdy je vždy nutné mít pro jeden protokol dva konfigurační soubory. Jeden, který popisuje jeho strukturu a podporované transakce a druhý, který obsahuje uživatelem definovaná data odpovědi.

```
[Simulator]
;Possible values - SPDH, ISO8583
PROTOCOL = SPDH
TCPPORT = 1600
```

Obrázek C.1: Konfigurační soubor definující protokol a číslo portu.

Společný konfigurační soubor je ve formátu ini a jeho podoba je zobrazena na obrázku C.1. Tento konfigurační soubor umožňuje definovat pouze číslo portu a protokol, přičemž jsou známé pouze protokoly SPDH a ISO8583.

Další konfigurační soubory jsou již specifické. Hlavním je soubor ve formátu xml, který popisuje strukturu protokolu. Soubor je rozdělen na několik částí:

- Hlavička protokolu,
- Data protokolu,
- Podporované transakce,
- Definice výčtových typů protokolu.

Poslední konfigurační soubor pak slouží k definování uživatelských položek odpovědi. Část tohoto souboru je stejná pro oba protokoly, část je naopak rozdílná podle použitých polí protokolu a podle jeho potřeb. Pokud jsou nastavené obě hodnoty RSPCODEBYAMOUNT a USESPECIFIC, tak se v případě response kódu použije částka transakce.

Protokol SPDH

```
<?xml version="1.0" encoding="UTF-8"?>
<protocol>
    <info name="SPDH" format="ascii" date="02-02-2017" />
    <header>
        <headerValue name="Device Type" format="AN" length="2" />
        <headerValue name="Transmission Number" format="N" length="2"/>
        <headerValue name="Terminal ID" format="ANS" length="16"/>
        <headerValue name="Employee ID" format="ANS" length="6"/>
        <headerValue name="Current Date" format="" length="6"/>
        <headerValue name="Current Time" format="" length="6"/>
        <headerValue name="Message Type" format="AN" length="1" possibleValue="msgType" />
        <headerValue name="Message SubType" format="AN" length="1" possibleValue="msgSubType" />
        <headerValue name="Transaction Code" format="N" length="2" possibleValue="tranCode" />
        <headerValue name="Response Code" format="N" length="3"/>
    </header>
    <fields>
        <field id="A" name="Customer Billing Address" format="N" length="..20" />
        <field id="B" name="Amount 1" format="N" length="..18" />
        <field id="C" name="Amount 2" format="N" length="..18" />
        <field id="6" name="Variable 6" format="SF" length="0" >
            <subfields format="sfid">
                <subfield id="E" name="POS entry mode" length="3"/>
                <subfield id="I" name="Transaction currency code" length="3"/>
                <subfield id="O" name="EMV request data" length="..136"/>
                <subfield id="P" name="EMV Additional Request data" length="..64"/>
                <subfield id="Q" name="EMV response data" length="..64"/>
                <subfield id="R" name="EMV Reversal Data/EMV Add Response Data" length="..258"/>
            </subfields>
        </field>
    </fields>
    <messages ids="msgType, tranCode" >
        <message id="F00" name="Normal Purchase" >
            <req mandatoryValues="Bq/B2/BDq8(A)/BD28(A)/Bq6(E)/Bq6(EIO)" />
        </message>
        <message id="A95" name="Handshake request" >
            <req mandatoryValues="" />
        </message>
    </messages>
    <possibleValues>
        <possibleValue id="msgType">
            <value id="A" name="Administrative transaction"/>
            <value id="F" name="Financial transaction"/>
            <value id="L" name="Pass-through Administrative transaction"/>
            <value id="M" name="Pass-through Financial transaction"/>
        </possibleValue>
    </possibleValues>
</protocol>
```

Obrázek C.2: Část konfiguračního souboru protokolu SPDH.

```
[SPDH specific]
RSPCODEBYAMOUNT = y
USESPECIFIC = N
RSPCODE = 111
AMOUNT =
RSPTEXT = TEST TERMINAL TEXT
SEQNUMBER =
APPROVALCODE = 85632010
```

Obrázek C.3: Konfigurační soubor s uživatelem definovanými daty pro protokol SPDH.

Protokol ISO8583

```
<?xml version="1.0" encoding="UTF-8"?>
<protocol>
    <info name="ISO8583" format="binary" date="12-04-2017" />
    <header>
        <headerValue name="Message Type ID" format="n" length="4" possibleValue="msgType" />
        <headerValue name="Bit Map" format="b" length="64" />
        <headerValue name="Secondary Bit Map - unused" format="b" length="0" />
    </header>
    <fields>
        <field id="2" name="Primary Account Number" format="n" length="..19" />
        <field id="3" name="Processing Code" format="n" length="6" possibleValue="procCode" />
        <field id="4" name="Amount" format="n" length="12" />
        <field id="11" name="System Trace Number" format="n" length="6" />
        <field id="12" name="Local Transaction Time" format="n" length="6" />
        <field id="13" name="Local Transaction Date" format="n" length="4"/>
        <field id="24" name="NII" format="n" length="3" />
        <field id="53" name="Security Control Info" format="n" length="16" >
            <subfields>
                <subfield name="Terminal Serial Number" format="n" length="12"/>
                <subfield name="Unused" format="n" length="4"/>
            </subfields>
        </field>
        <field id="54" name="Additional Amounts" format="AN" length="..120" />
        <field id="55" name="ICC System Relatd Data" format="ANS" length="..255" />
        <field id="60" name="Private Use" format="ANS" length="..999" />
    </fields>
    <messages ids="msgType, procCode" >
        <message id="020000**0*" name="Sale" >
            <req mandatoryValues="3,4,11,22,24,25,35,41,42/3,4,11,22,24,25,41,42,45/3,4,11,22,24,25,35,41,42,55" />
        <message id="080099**0*" name="Handshake" >
            <req mandatoryValues="3,11,24,41" />
        </message>
    </messages>
    <possibleValues>
        <possibleValue id="msgType">
            <value id="0100" name="Authorization Request"/>
            <value id="0110" name="Authorization Request Response"/>
            <value id="0200" name="Financial Transaction Request"/>
            <value id="0210" name="Financial Transaction Request Response"/>
            <value id="0220" name="Financial Transaction Advice"/>
            <value id="0230" name="Financial Transaction Advice Response"/>
            <value id="0400" name="Reversal Request"/>
            <value id="0410" name="Reversal Request Response"/>
        </possibleValue>
    </possibleValues>
</protocol>
```

Obrázek C.4: Část konfiguračního souboru protokolu ISO8583.

```
[ISO specific]
RSPCODEBYAMOUNT = N
USESPECIFIC = Y
RSPCODE = 51
AMOUNT =
AUTHCODE =
```

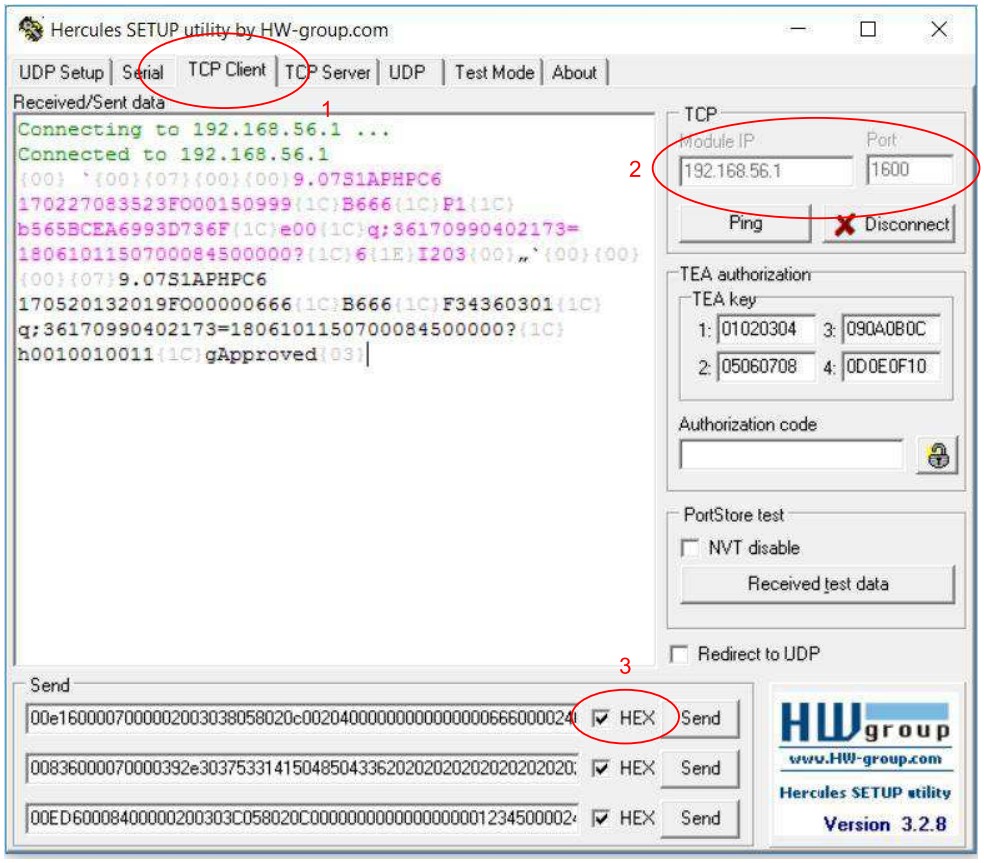
Obrázek C.5: Konfigurační soubor s uživatelem definovanými daty pro protokol ISO8583.

Příloha D

Použití nástroje Hercules

Pokud nemáme k dispozici terminál, je vhodné pro otestování simulátoru použít nástroj Hercules[1], který umožňuje zadávat testované zprávy i v hexadecimální podobě.

Můžeme použít zprávy uvedené v sekci Testování vůči předpřipraveným zprávám. Takové zprávy je nutné ještě upravit a přidat na začátek každé zprávy sedm bytů TPDU a směrovací data. Nicméně v tomto případě mohou být libovolná, například 0x01020304050607.



Obrázek D.1: Použití nástroje Hercules.

[1] Hercules. Dostupné z: http://www.hw-group.com/products/hercules/index_cz.html

V rámci programu Hercules (obrázek D.1) je nutné zvolit záložku TCP Client (bod 1 na obrázku D.1), dále nastavit příslušnou IP adresu a port simulátoru (bod 2 na obrázku D.1) a také před zasláním zprávy zvolit HEX (bod 3 na obrázku D.1) data. Fialovou barvou je pak ve výpisu zobrazena odeslaná zpráva a černou barvou přijatá odpověď.