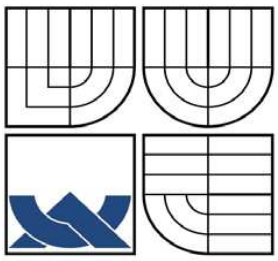
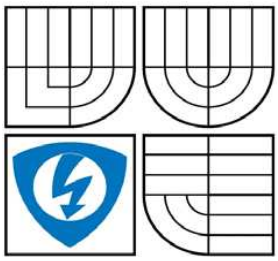


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE PROTOKOLU ACP DO OPERAČNÍHO SYSTÉMU L4

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

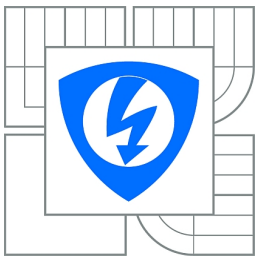
AUTOR PRÁCE
AUTHOR

Bc. TOMÁŠ KOLARÍK

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. KAREL BURDA, CSc.

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Tomáš Kolarík ***ID:*** 78932
Ročník: 2 ***Akademický rok:*** 2011/2012

NÁZEV TÉMATU:

Implementace protokolu ACP do operačního systému L4

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte a popište univerzální protokol řízení přístupu ACP (Access Control Protocol). Dále nastudujte a popište operační systém L4. Na tomto základě navrhněte koncept implementace protokolu řízení přístupu ACP do operačního systému L4. Svůj návrh zdůvodněte, prakticky zrealizujte, otestujte a zhodnoťte.

DOPORUČENÁ LITERATURA:

[1] Burda, K.: Univerzální rámec pro řízení přístupu v počítačových sítích. Elektrorevue 2011/9. 6 s.
[2] L4Ka Team: L4 eXperimental Kernel. Reference Manual. Universitat Karlsruhe, Karlsruhe 2006.

Termín zadání: 6.2.2012 ***Termín odevzdání:*** 24.5.2012

Vedoucí práce: doc. Ing. Karel Burda, CSc.
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Práce se zabývá implementací protokolu ACP, který slouží k řízení přístupu pro operační systém založený na mikrojádře L4. Teoretická část práce se zabývá možnostmi řízení přístupu v počítačových sítích. Pozornost je se přitom soustřeďuje na AAA systémy, které řízení přístupu umožňují. Následně je podrobně popsán protokol ACP, typy jeho zpráv a reakce na ně. Druhá část teoretické přípravy je věnována operačním systémům kde se podrobněji zabývá jejich architekturou a službami. Následně je blíže popsána rodina mikrojader L4, její filozofie a vlastnosti. Také je blíže popsané aplikační rozhraní L4Re a možnosti jeho rozšíření. Praktická část se zabývá implementačním návrhem systému pro podporu protokolu ACP v počítačích. Obecný návrh je následně použitý při reálné implementaci ACP protokolu do prostředí operačního systému L4 postaveném na platformě L4Re. Je zde udělaný podrobný návod na tvorbu a překlad softwaru pro tuto platformu. Jsou tady popsané použité postupy při implementaci a popis jednotlivých modulů a funkcí. Závěr práce obsahuje informace o způsobe testování a vlastnostech implementace.

Klíčová slova

řízení přístupu, AAA protokoly, ACP protokol, implementace ACP, operační systémy, služby operačního systému, mikrojádro L4, Fiasco.OC, L4Re, ACP server, AC portál

Abstract

This thesis deals with the implementation of ACP protocol which serves to manage the access for operation system based on L4 microkernel. The theoretical part of the thesis deals with methods of access management in computer networks. It focuses primarily on AAA systems which make access management possible. Furthermore it describes in detail the ACP protocol, the types of messages and their feedback. The next theoretical part is dedicated to operation systems and in particular to their architecture and services. Then we get a closer look at L4 microkernel family, their philosophy and properties. We continue with a detailed description of the L4 application interface and its ways of expansion. The practical section deals with the implemented concept of system for ACP protocol support in computers. General concept is then applied in real implementation of ACP protocol into the L4 operation system environment based on the L4 platform. To assist, I also included a detailed tutorial explaining the modeling and compilation of software for this platform. At this point we describe the methods used at the implementation and the description of particular modules and features. The end of the thesis concludes the information about the ways of testing and the implementation properties.

Keywords

access control, AAA protocols, ACP protocol, ACP implementation, operating systems, operating system services, L4 microkernel, Fiasco.OC, L4Re, ACP server, AC portal

Citace

KOLARÍK, T. *Implementace protokolu ACP do operačního systému L4.* Brno: FEKT VUT v Brně, 2012. 48 s. Vedoucí práce doc. Ing. Karel Burda, CSc.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Implementace protokolu ACP do operačního systému L4 jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

podpis autora

Obsah

Úvod

Počítačové systémy sú v súčasnosti rozšírené v takmer všetkých oblastiach života. Ich prítomnosť užívateľ mnohokrát ani nevníma, ale pritom by bez ich služieb nemohol vykonávať požadovanú činnosť. Počítačové systémy poskytujú rôzne služby, ktoré môžu využívať buď ľudia alebo ďalšie počítačové systémy. Práve pre túto vlastnosť, t. j. možnosť zdieľania služieb mnohými užívateľmi, je čím ďalej tým viac kladený dôraz na riadenie prístupu k týmto službám. V súčasnosti je tento problém najčastejšie spojený s používaním počítačových sietí a internetu. Ich využitie sa v mnohých oblastiach stáva postupne nevyhnutným a aj preto rastú nároky na ich bezpečnosť. Ďalším dôvodom, pre ktorý je potrebné zaoberať sa riadením prístupu je monitoring využívania služieb a ich prípadné spoplatnenie.

Aby bolo riadenie prístupu možné realizovať je potrebné mať v systéme k dispozícii nástroje na autentizáciu, autorizáciu a účtovanie. Pomocou týchto nástrojov sme schopní zabezpečiť obmedzenie prístupu aj jeho spoplatnenie. Jedným z často používaných prostriedkov na dosiahnutie týchto cieľov je protokol RADIUS. Je označovaný ako AAA protokol, čo je skratka z anglických výrazov autentication, autorisation a accounting. Jeho mohutné rozšírenie ukázalo potrebu zaoberať sa aktívne AAA protokolmi. Protokol ACP vyvíjaný na univerzite patrí taktiež do skupiny AAA protokolov. Zatiaľ však neexistuje v praxi žiadna naplno využiteľná implementácia tohto protokolu. Preto sa bude táto práca zaoberať návrhom implementácie do operačného systému typu L4. Podľa organizácie Open Kernel Labs je operačný systém L4 celosvetovo nasadený na vyše 1,5 miliardy zariadení [13]. Preto by podpora protokolu ACP mohla prispieť k jeho rozšíreniu a zároveň zvýšeniu bezpečnosti počítačových systémov.

Nasledujúca kapitola vysvetlí problém riadenia prístupu, spôsob jeho riešenia pomocou AAA systémov. Ďalej sa bude zaoberať rôznymi konfiguráciami AAA systémov a nakoniec bude podrobne popísaný ACP protokol.

1 Riadenie prístupu

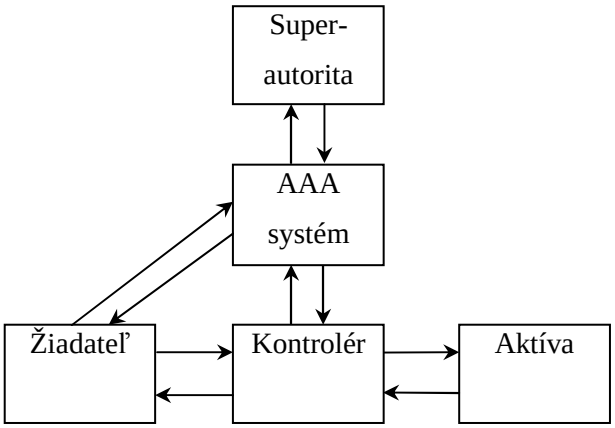
Sieťové technológie v súčasnosti užívateľom umožňujú prístup k rôznym službám. V mnohých prípadoch je však potrebné prístup k nim riadiť. A to z dôvodu zachovania ich dôvernosti alebo potreby spoplatnenia ich využívania či iných dôvodov. K tomuto účelu boli vyvinuté sieťové nástroje, ktoré zabezpečujú:

- autentizáciu - overenie identity užívateľa,
- autorizáciu - vydávanie oprávnenia na používanie požadovaných služieb,
- účtovanie - zbieranie informácií o využívaní služieb užívateľom.

Prostriedky využívané v oblasti riadenia prístupu sú často označované skratkou AAA (z anglického authentication, authorization, accounting).

1.1 Proces riadenia prístupu

Proces prístupu k aktívam ovplyvňujú nasledujúce entity. Žiadateľ je osoba požadujúca prístup k aktívu. Správca alebo majiteľ, ktorý má na starosti prideľovanie práv žiadateľom na prístup k aktívam sa bude ďalej označovať ako super-autorita. AAA systém je samostatný aktívny prvok v sieti alebo skupina prepojených prvkov, ktorý sa stará o autonómne riadenie prístupu, tým že zabezpečuje autentizáciu žiadateľa na základe informácií od super-autority. Ďalej na základe žiadateľovej identity ho autorizuje na prístup k aktívam a zároveň vedie záznamy o týchto prístupoch. Kontrolér je brána pre žiadateľa k aktívam, riadi k nim prístup na základe oprávnenia od AAA systému a informuje AAA systém o narábaní žiadateľa s aktívami [2].



Obrázok 1.1: Tok informácii v procese riadenia prístupu

1.2 Autentizácia autorizácia a účtovanie

Autentizácia je proces overovania identity užívateľa. Používa sa za účelom jednoznačnej identifikácie užívateľa pri jeho snahe o prístup k službám. Na základe jeho identity je potom

možné služby patrične upraviť, sprístupniť, či zamedziť prístup. Nevyhnutnou podmienkou k tomu, aby mohol byť užívateľ zaradený do prístupového zoznamu je nutné, aby došlo k vzájomnej výmene informácií medzi budúcim užívateľom a autoritou. V rámci tejto komunikácie je užívateľovi pridelená identita, t.j. jeho označenie v systéme. Ďalej sú mu pridelené prístupové práva a nakoniec vyjednaný dokazovací a overovací faktor. Dokazovacím faktorom (napr. heslom) dokazuje užívateľ svoju identitu, overovacím faktorom (napr. hašom hesla) autorita overuje identitu užívateľa [2].

Autorizácia je pridelenie oprávnenia pre vykonávanie určitej činnosti. Autorizáciu vykonáva autorita, čo je osoba poverená touto úlohou. Vydávanie oprávnenia môže byť vykonávané osobou alebo môže byť zautomatizované. U zautomatizovanej autorizácie sa rozhoduje o prístupe na základe pravidiel stanovených autoritou. Väčšinou sa viažu na osobu žiadateľa a prípadné ďalšie informácie, ako je stav účtu atď. V sieťových AAA systémoch sa stretávame hlavne s automatizovanou autorizáciou. Osobná sa vykonáva iba v prípade zavádzania nového užívateľa alebo pri úprave jeho práv, či jeho vyradzovaní [2].

Účtovanie slúži k zhromažďovaniu a uchovávaniu informácií o prístupe užívateľov k aktívam. Na ich základe môže poskytovateľ vyžadovať platby za poskytnuté služby. Ďalej sa tieto informácie dajú využiť na odhalenie nežiadúceho nakladania s aktívami, ako aj efektívnosť ich využívania. Informácie sú získavané od kontroléra, cez ktorý užívatelia pristupujú k aktívam [2].

1.3 Konfigurácia AAA systému

Vnútorne možno AAA systém rozdeliť na nasledujúce časti:

1. Autentizátor zabezpečuje identifikáciu žiadateľa. Jeho identitu následne poskytne druhej časti AAA systému nazývanú autorita.
2. Autorita na základe identity užívateľa a jeho požiadavky rozhodne o podmienkach prístupu. Tieto podmienky sú predané kontroléru, ktorý v niektorých prípadoch býva priamo súčasťou AAA systému.
3. Účtovateľ je poslednou základnou časťou vnútornej štruktúry. Ten zbiera a uchováva záznamy o prístupe žiadateľov k aktívam.

Súčasťou AAA systémov sú aj ďalšie prvky: napr. databáza prístupových práv, avšak pre vysvetlenie princípu AAA systémov sú nepodstatné [2].

Podľa rozsahu integrácie jednotlivých častí AAA systému rozlišujeme nasledujúce tri konfigurácie:

- distribuovaný AAA systém – každá z častí AAA systému je tvorená samostatným sieťovým zariadením,
- centralizovaný AAA systém – kontrolér je samostatný sieťový prvok, autentizátor, autorizátor a účtovateľ sú integrované do AAA servera,
- kompaktný AAA systém – autentizátor, autorizátor, účtovateľ a kontrolér sú integrované do jediného zariadenia, toto zariadenie budeme označovať ako AAA portál.

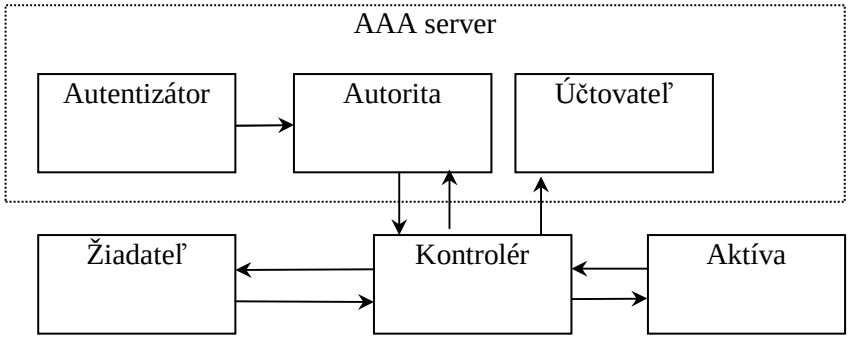
U distribuovaného systému najskôr prebehne autentizácia voči autentizátoru. Ten informuje autoritu, ktorá rozhodne o prístupových právach a odošle ich kontroléru. Kontrolér informuje

užívateľa o možnostiach prístupu a zároveň účtovateľa o prístupoch užívateľa k aktívam. Táto konfigurácia sa nepoužíva pre jej náročnosť na správu a hardvér [2].

Najčastejšie sa môžeme stretnúť s centralizovanou konfiguráciou, kde sa sústreďuje autentizátor, autorizátor a účtovateľ v jednom sieťovom prvku. Zatiaľ čo kontrolérov môže byť v sieti hneď niekoľko. Táto konfigurácia umožňuje jednoduché a centralizované riadenie prístupu. Komunikácia v tomto prípade prebieha len medzi kontrolérom, AAA serverom a žiadateľom. Spôsob predávania informácií pri tejto konfigurácii záleží na variante tzv. sprostredkovateľskej entity. Na jej základe rozlišujeme tieto možnosti konfigurácie[2]:

- sprostredkovateľom je AAA server,
- sprostredkovateľom je kontrolér,
- sprostredkovateľom je žiadateľ.

Sprostredkovateľ vytvára premostenie v komunikácií medzi dvoma ďalšími stranami, čím sa zníži počet komunikačných kanálov. Najčastejšie sa v praxi stretávame s kontrolérom ako sprostredkovateľom. V tomto prípade sa na začiatku pri autentizácii medzi žiadateľom a AAA serverom kontrolér chová pasívne a iba postupuje správy od žiadateľa do AAA servera a naopak. Až po získaní autorizačnej správy z AAA servera a o možnostiach prístupu žiadateľa nastaví možnosť prístupu podľa rozsahu práv a pošle o tom hlásenie žiadateľovi. Týmto spôsobom sa v dnešnej dobe najčastejšie riadi prístup do sietí. Dôvodom je skutočnosť že žiadateľ na začiatku nemôže priamo komunikovať s AAA serverom priamo, ale môže komunikovať s hraničným prvkom, ktorý je vlastne kontrolérom na riadenie prístupu.



Obrázok 1.2: Časti AAA systému [3]

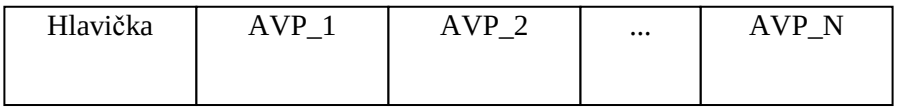
Jednotlivé prvky AAA systému medzi sebou komunikujú po sieti, k čomu využívajú komunikačné protokoly. Ich popis definuje štruktúru a význam jednotlivých správ, ich sekvencie a reakcie entít na ne. Ak protokol dovoľuje sieťovým spôsobom zabezpečiť autentizáciu, autorizáciu a účtovanie, označuje sa ako AAA protokol. Medzi najznámejšie AAA protokoly patrí RADIUS, TACACS+ a Diameter [2]. V ďalšej časti sa budeme venovať ACP protokolu na riadenie prístupu.

1.4 ACP protokol

ACP je protokol pre jednotnú implementáciu rôznych metód riadenia prístupu k aktívam sieťových zariadení. AC portály (ďalej len portály) používajú protokol ACP k riadeniu prístupu, buď k vlastným aktívam alebo vyjednávajú prístup k aktívam iných zariadení. Portál je softvérová výbava sieťového zariadenia, nastaviteľná podľa potreby systému alebo hardvérových prostriedkov zariadenia. Portál zariadenia žiadajúceho aktíva sa označuje ako žiadateľ a portál zariadenia poskytujúceho aktíva sa označuje ako poskytovateľ. Proces vyjednávania prístupu sa označuje ako transakcia. Protokol umožňuje, buď priamu komunikáciu žiadateľa s poskytovateľom alebo umožňuje túto komunikáciu tranzitovať cez iné portály pokiaľ neexistuje priame spojenie. Uvedené spoje môžu byť, buď trvalé alebo dočasné. Trvalé budú väčšinou v rámci jednej organizácie, či zariadenia. Dočasné budú napríklad medzi prvkami viacerých organizácií. Popísané riešenie vyžaduje sieťovú adresáciu, podporu smerovania a možnosť tranzitovania správ[3].

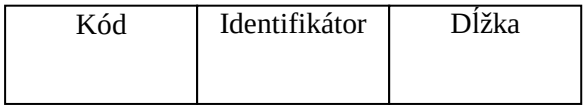
1.4.1 Štruktúra ACP správ

ACP správy sú zložené z hlavičky a z AVP položiek. Ich počet môže byť N kde N = 0, 1, 2, 3, … Správa kde N = 0 sa nazýva prázdna správa. Veľkosti jednotlivých polí sú udávané v oktetoch, čo je dátová jednotka o veľkosti 8 bitov pokiaľ nie je uvedené inak.



Obrázok 1.3: Formát ACP správy [3]

Hlavička správy pozostáva z polí Kód (Code), Identifikátor (Identifier) a Dĺžka (Length).



Obrázok 1.4: Formát hlavičky ACP správy [3]

Kód hlavičky umožňuje rozlíšiť, že ide o správu protokolu ACP a udáva typ správy v rámci ACP protokolu. Pomocou kódu hlavičky sa dajú rozlišovať protokoly EAP a ACP v spoločnom spoji [3].

Pole Identifikátor v hlavičke obsahuje unikátne číslo, ktorým sa v danom spoji rozlišujú správy určitej transakcie od správ iných transakcií. Hodnotu poľa Identifikátor pre každú transakciu v danom spojení určuje portál, ktorý v danom spoji vyšle prvú správu tejto transakcie t.j. správu Start. Portály v každom spoji určujú hodnotu poľa Identifikátor nezávisle, preto môže nastať situácia, že zvolia pri vzájomnej komunikácií rovnaké číslo Identifikátor, vtedy sa budú správy odlišovať podľa bitu M0.

Pole Dĺžka v hlavičke udáva celkovú dĺžku správy vrátane záhlavia v oktetoch. Pokiaľ portál príjme správu s nesprávnou dĺžkou, musí transakciu zrušiť. Zrušenie sa vykoná

zmazaním prevádzkových informácií súvisiacich s danou transakciou. Portál na prípadné ďalšie správy zrušenej transakcie nereaguje a tak dôjde k prerušeniu spojenia. Zrušenie transakcie vo všetkých ostatných zúčastnených portáloch nastane po vypršaní časového limitu. Poskytovateľ môže taktiež transakciu ukončiť zaslaním prázdnej správy typu Finish.

P	R3 .. R0	M2 .. M0
1 bit	4 bity	3 bity

Obrázok 1.5: Rozdelenie bitov v položke Kód hlavičky ACP správy [3]

Význam jednotlivých bitov poľa Kód je nasledujúci:

- P bit odlišuje správu ACP od správy protokolu EAP, preto musí vždy nadobúdať hodnotu 1.
- Bity R3 .. R0 sú rezervné, ich hodnota musí byť nulová.
- Bity M2 .. M0 určujú typ správy. Bit M0 súčasne udáva smer prenosu správy. Správy s M0 = 0 sú správy od žiadateľa k poskytovateľovi, správy s M0 = 1 v opačnom smere.

1.4.2 Typy ACP správ

Aktuálna verzia ACP protokolu definuje šesť druhov správ. Ich výmenou, medzi žiadateľom a poskytovateľom prebieha transakcia, pri ktorej si dojednajú autentizačnú metódu aj aktíva, o ktoré má žiadateľ záujem. Jednotlivé typy sú odlíšené pomocou bitov M2 .. M0 v položke Kód v hlavičke [3].

- Start (M2 .. M0 = [000]) – je vždy prvou správou transakcie a vysiela ju žiadateľ. Môže obsahovať požiadavky na sprístupnenie konkrétnych aktív a taktiež môže obsahovať návrh na typ autentizácie.
- Offer (M2 .. M0 = [001]) – správa je vysielaná poskytovateľom. Je v nej uvedená ponuka dostupných aktív alebo podporované spôsoby autentizácie. Správa je posielaná pokiaľ dané údaje neboli poslané v správe Start.
- Specification (M2 .. M0 = [110]) – správa, ktorou žiadateľ reaguje na správu Offer. Slúži na výber konkrétnych aktív alebo typu autentizácie z ponuky zaslanej správou Offer.
- Request (M2 .. M0 = [101]) – správa je odosielaná poskytovateľom v rámci autentizácie. Touto správou sa začína proces autentizácie.
- Response (M2 .. M0 = [010]) – správou žiadateľ reaguje na správu Request v rámci autentizačného procesu.
- Finish (M2 .. M0 = [111]) – táto správa uzatvára transakciu, odosiela ju vždy poskytovateľ. Obsahuje informácie o poskytnutí, či neposkytnutí aktíva a prípadné ďalšie informácie. Správa Finish spôsobuje v každom uzle vymazanie údajov potrebných k prenosu správ danej transakcie.

Obsah jednotlivých správ je uložený v príslušných AVP štruktúrach. Správy typu Start a Finish môžu byť prázdne, ostatné by mali obsahovať aspoň jednu AVP. Typom a významom jednotlivých AVP sa venuje nasledujúca kapitola.

1.4.3 Formáty a rozdelenie AVP

AVP (skratka z anglického Attribute-Value Pair) je dátová štruktúra využívaná na prenos užitočných dát v protokole ACP. Štruktúra AVP je znázornená na nasledujúcom obrázku.

Obrázok 1.6: Formát AVP dátovej štruktúry [3]

Význam jednotlivých polí je nasledujúci [4]:

- Pole Typ definuje význam dát v poli Hodnota, má dĺžku 1 oktet.
- Pole Dĺžka definuje dĺžku poľa Hodnota v oktetoch. Podľa typu AVP správy má táto položka dĺžku 1 oktet (typ SAVP) alebo 2 oktety (typy LAVP a CAVP).
- Pole Hodnota obsahuje samotné dáta.

AVP sa rozdeľujú na nasledujúce tri typy:

- SAVP – krátke AVP, tento typ obsahuje jediný typ dát o dĺžke kratšej než 2^8 oktetov. Hodnota Typ týchto AVP je v rozsahu 0-127.
- LAVP – dlhé AVP, tento typ obsahuje jediný typ dát o dĺžke kratšej než 2^{16} oktetov. Tento typ je určený na prenos dlhších blokov dát. Hodnota Typ týchto AVP je v rozsahu 128-191.
- CAVP – kontajnerové AVP, obsahuje jedno alebo viac ACP ľubovoľného typu v ľubovoľnej kombinácií. Celková dĺžka všetkých obsiahnutých AVP musí byť kratšia než 2^{16} oktetov. Tento formát slúži k zoskupovaniu AVP podľa rôznych kritérií, napr. rovnaký typ AVP. Hodnota Typ týchto AVP z rozsahu 192-255.

Pokiaľ žiadateľ alebo poskytovateľ prijíma správu AVP, ktorú nedokáže spracovať musí transakciu zrušiť. Zrušenie sa vykoná zmazaním prevádzkových informácií súvisiacich s danou transakciou. Zrušenie transakcie vo všetkých ostatných zúčastnených portáloch nastane po vypršaní časového limitu. Poskytovateľ môže taktiež transakciu ukončiť zaslaním prázdnej správy typu Finish.

1.4.4 Typy AVP

Jednotlivými typmi AVP je možné určiť a preniesť zrozumiteľným a prehľadným spôsobom informácie potrebné pre riadenie prístupu. Typy sú rozdelené podľa obsahu prenášaných dát do skupín.

1.4.4.1 Mená entít

Tieto AVP umožňujú prenášať v textovej podobe mená entít, pomocou ich typu je taktiež možné rozlíšiť rolu entity. Tieto AVP sa označujú reťazcom NAME_AAA_B, kde AAA je skratkou role entity (SUP – žiadateľ, PRO – poskytovateľ, AUT – autentizátor, ACC – účtovateľ). Znakom na mieste B v zápise sa rozoznáva lokálna alebo globálna platnosť, napr.: NAME_SUP_G (Typ AVP = 0) označuje globálne meno žiadateľa a môže obsahovať hodnotu napríklad vo forme e-mailovej adresy. Typy AVP sú z rozsahu 0-7 [4].

1.4.4.2 Adresy zariadení

Tieto AVP umožňujú prenášať adresy zariadení v číselnej podobe. Pre značenie platia rovnaké pravidlá ako u skupiny mená entít a označujú sa reťazcom ADDR_AAA_B. Typ adresy sa rozlišuje na základe dĺžky AVP, kde dĺžka 16 označuje IPv6 adresu, dĺžka 6 označuje MAC adresu, dĺžka 4 adresu IPv4 a dĺžka 2 označuje číslo TCP/UDP portu. Typy AVP sú z rozsahu 16-23 [4].

1.4.4.3 Kódy metód

Nasledujúce AVP umožňujú prenášať kódy autentizačných metód. Predpokladá sa hlavne využitie metód založených na protokole EAP, preto je definovaný SAVP typ EAP (Typ AVP = 32) Hodnota o veľkosti väčšinou 1 oktet v tomto prípade vyjadruje typ EAP metódy. Pre lokálne definované metódy je definovaný typ LAM (Typ AVP = 33). V tomto prípade kódy metód definuje správca [4].

1.4.4.4 Kódy aktív

Nasledujúce AVP umožňujú prenášať kódy aktív, tie je možné obecne kódovať globálne alebo lokálne. Pre globálne kódy zatiaľ neexistuje príslušný štandard, globálne AVP sa označujú ASSET_G (Typ AVP = 36). Lokálne kódovanie aktív je definované správcom systému, označujú sa ako ASSET_L (Typ AVP = 37). Môže byť definované pomocou hierarchického stromu, kedy každý oktet hodnoty ACP vyjadruje číslo vetvy na danej úrovni stromu až po jeho list, ktorým je samotné aktívum. Poskytovateľ tak môže prehľadným spôsobom ponúkať aktíva postupne v interakcii so žiadateľom. Ak mu v každej správe Offer pošle úroveň, na ktorej si môže užívateľ vybrať, buď služby alebo sa môže vnárať do ďalších úrovní pomocou výberu vetvy, ktorou sa má pokračovať [4].

1.4.4.5 Ďalšie typy AVP

Je definovaných mnoho ďalších druhov AVP, ktorými sa môžu prenášať texty, zoznamy AVP štruktúr, návratové hodnoty transakcie, varianty ACP protokolu, kryptografické primitíva. Nakoniec je možné pomocou na to vyhradených AVP prenášať celé správy protokolov RADIUS, Diameter a Kerberos, či samotné AVP protokolov RADIUS a Diameter zabalené do AVP protokolu ACP. Popis všetkých typov AVP je k dispozícii v RFC dokumentoch k jednotlivým protokolom.

1.4.5 Transakcie ACP a ich spájanie

Protokol ACP umožňuje komunikujúcim stranám dohodnúť požadované aktíva, autentizačnú metódu, vykonať autentizáciu, doručiť prístupové parametre a vykonávať účtovanie. Strany komunikujú striedavým posielaním správ, čiže strana môže vyslať správu až po prijatí správy z opačnej strany. Ochranu proti chybám, segmentáciu a ďalšie služby zaisťuje príslušný komunikačný modul portálu. Transakcia má nasledujúci priebeh:

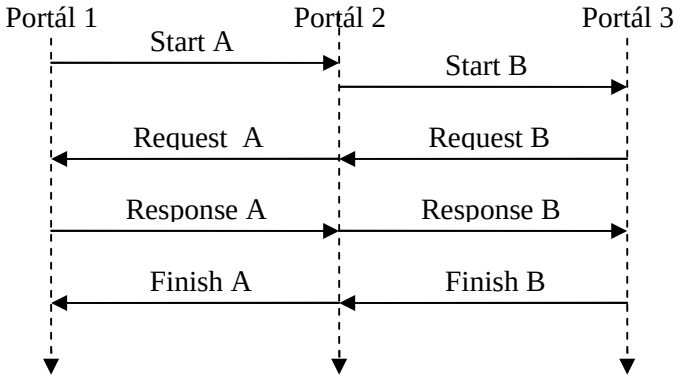
1. Žiadateľ pošle poskytovateľovi správu typu Start.
2. Ak poskytovateľ nemá dostatok informácií na vykonanie transakcie, začne fázu vyjednávania. V tejto fáze dohodne príslušný spôsob autentizácie a požadované

aktíva pomocou správ typu Offer a Specification. Po získaní potrebných informácií Poskytovateľ prejde do fázy autentizácie.

3. Autentizáciu Poskytovateľ začne správou Request a očakáva správnu odpoveď typu Response. Týchto výmen môže nastať niekoľko.
4. Na základe výsledku autentizácie a autorizácie Poskytovateľ ukončí transakciu správou Finish.

Prepojenie týchto elementárnych transakcií je možno urobiť dvoma spôsobmi. Transakcie sa dajú zreťaziť alebo do seba vkladať.

Zreťazenie transakcií umožňuje vytvárať sekvencie transakcií, kedy po ukončení jednej sa spustí transakcia iná ako následok predošlej transakcie. Vkladanie transakcie nastáva ak na skončenie jednej transakcie musí prebehnúť transakcia iná.



Obrázok 1.7: Príklad vloženia transakcie [4]

1.4.6 Protokol ACP-VSA

Protokol ACP je flexibilne navrhnutý tak, aby umožňoval vytvoriť niekoľko variant chovania pri transakciách. Transakciu v každej variante však riadi Poskytovateľ. Komunikujúce strany sa po zahájení transakcie môžu dohodnúť aj na inej variante, avšak východisková je varianta ACP-VSA. Varianta ACP-VSA, je založená na systéme jednoduchých odpovedí.

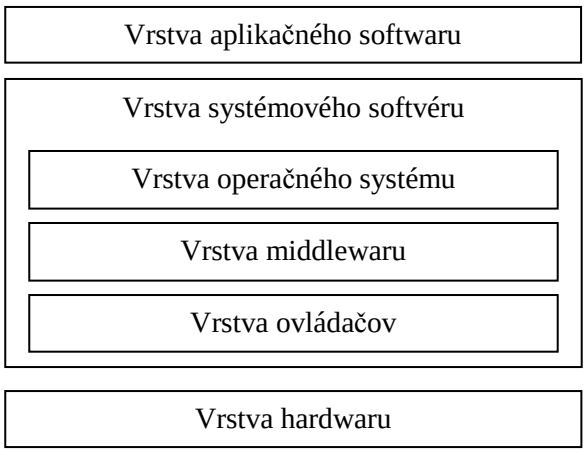
Komunikácia v tejto variante prebieha nasledovne. Poskytovateľ po prijatí správy typu Start začne fázu vyjednávania. Tú vedie spôsobom, že posiela takú ponuku alebo dotaz žiadateľovi, na ktoré môže v každej odpovedi posielať práve jedno AVP. Posielaná ponuka vždy obsahuje N rôznych AVP rovnakého typu, z ktorých si žiadateľ môže jednu vybrať. V prípade dotazu, poskytovateľ zasiela prázdne AVP a prípadne ďalšie AVP, ak sú potrebné k zisteniu správnej odpovedi na dotaz. Žiadateľ vyplní správnou odpoveďou prázdne AVP a pošle ho v správe Specification späť. Ak pošle žiadateľ viac ako 1 AVP, poskytovateľ musí danú transakciu ukončiť zaslaním prázdnej správy Finish. Postup vyjednávania zo strany Poskytovateľa definuje správca v príslušnom správnom module. Po vyjednaní parametrov prebehne výmena autentizačných správ. Po autentizácii sa posiela správa Finish s výsledkom a transakcia sa týmto ukončí [4].

2 Operačné systémy

Každý počítačový systém obsahuje základnú sadu programov nazývanú operačný systém. Operačný systém (skrátene OS) vytvára prostredie, v ktorom sú spúšťané aplikácie. Aplikáciám a jej tvorcom prináša mnoho výhod, od prenositeľnosti na rôzne architektúry procesorov cez rôzne služby, ktoré by inak museli byť priamo implementované v aplikácii. Ďalšou výhodou OS je, že často vytvára užívateľsky prívetivé prostredie, ktoré zvyšuje efektivitu práce a spríjemňuje prácu užívateľovi.

Operačný systém je softvér, ktorý plní dve základné úlohy. Predstavuje abstrakčnú vrstvu pre vrchnú časť softvéru, aby bol čo možno najmenej závislý na použitom hardvéru. Tým sa taktiež uľahčuje vývoj middlewaru a aplikačného softvéru umiestneným nad OS. Ďalšou úlohou je správa hardvérových a softvérových prostriedkov. Zabezpečuje, aby boli využívané efektívne a zároveň bezpečne. [16]

V tejto kapitole bude čiastočne priblížená rozsiahla problematika operačných systémov. V úvode budú priblížené hardvérové nástroje využívané pri behu OS. Ďalej bude pozornosť sústredená predovšetkým na OS s mikrojadrom a koniec bude venovaný operačným systémom L4.

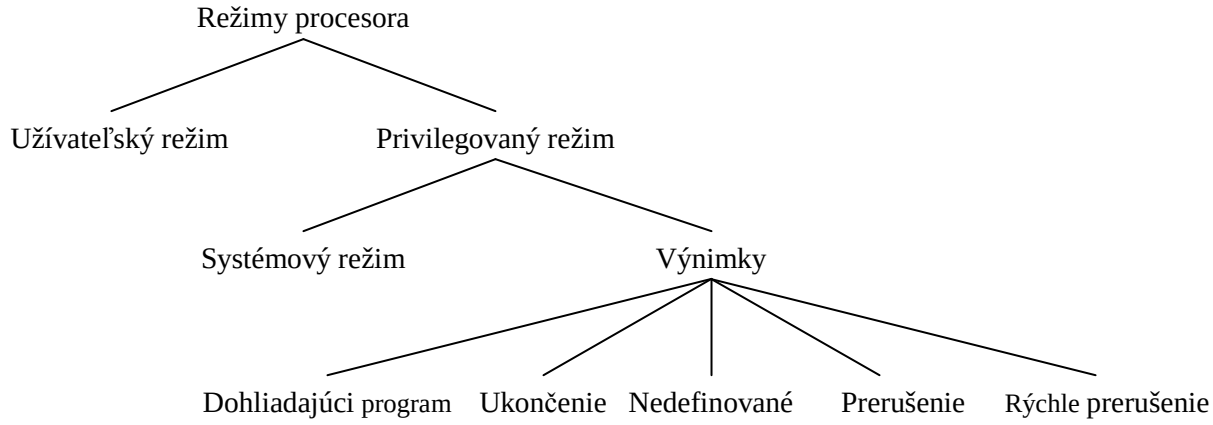


Obrázok 2.1: Príklad umiestnenia OS v architektúre vstavaného systému

2.1 Režimy procesora

Prevažná väčšina súčasných procesorov podporuje činnosť vo viacerých režimoch. Rozlišujú sa štandardne dva typy režimov, a to privilegovaný a užívateľský. Procesory ARM môžu bežať v jednom zo šiestich rozdielnych privilegovaných režimov mimo užívateľského režimu. Týchto šesť režimov je vytvorených pre zjednodušenie a zrýchlenie obsluhy prerušení a výnimiek. Ak sa procesor nachádza v privilegovanom režime, niektoré registre sú zamenené za fyzicky iné. Vyhradené registre pre daný konkrétny režim namiesto tých, ktoré sa používajú v užívateľskom režime. Aplikačný program obvykle beží v užívateľskom režime. Ten je neprivilegovaný, preto neumožňuje aplikačnému programu vykonávať všetky

operácie, napríklad nie je povolená zmena režimu procesora. Ak sa v užívateľskom režime procesor pokúsi zmeniť režim prepísaním príslušného registra, nastane výnimka. Na rozdiel od privilegovaných režimov, v ktorých je povolené meniť režim procesora ľubovoľným spôsobom. Takzvaný systémový režim využíva rovnaké registre ako užívateľský režim, avšak nie sú v tomto režime aplikované žiadne obmedzenia. Tento režim využíva operačný systém. [15] [17]



Obrázok 2.2: Režimy ARM procesora [17]

2.2 Delenie operačných systémov

Jednotlivé OS sa od seba značne líšia, či už rozsahom služieb, podporovanou architektúrou a inými viac či menej zásadnými vlastnosťami. Na základe týchto rozdielov sa dajú OS rozdeliť podľa niekoľkých parametrov ako napríklad podľa počtu užívateľov, podľa počtu súčasne bežiacich procesov podľa času odozvy systému a ďalších parametrov. Aj napriek tejto rôznorodosti majú všetky OS spoločné to, že obsahujú jadro. Jadro je časť systému, ktorá sa pri zavádzaní spúšťa ako prvá, a je najdôležitejšou časťou OS z pohľadu jeho fungovania. Niektoré OS pozostávajú iba zo samotného jadra (napr. FreeRTOS). Jadra OS sa dajú rozdeliť do štyroch kategórii [18]:

- Monolitické jadrá – sú používané hlavne u Unixoých systémov. Všetky služby sú súčasťou jadra vrátane ovládačov zariadení pozri obr. 2.4, vyznačuje sa vysokým výkonom. [1]
- Mikrojadrá – obvykle vytvárajú iba malú sadu základných abstrakcií, ako je správa pamäte, správa procesov a medziprocesová komunikácia. Ostatné služby bežia ako procesy mimo jadra pozri obr. 2.3.
- Hybridné jadrá – sú podobné mikrojadrám, avšak obsahujú v jadre pridaný kód, ktorý tak môže bežať omnoho rýchlejšie než v užívateľskom režime. Je to kompromis medzi monolitickým typom jadra a mikrojadrom. Tento typ jadra má napríklad Windows NT, Windows XP a ďalšie.

- Exojadrá – je stále len experimentálny prístup k návrhu jadra OS. Ich funkcionalita je obmedzená len na ochranu a zdieľanie surového hardvéru. Ostatné služby sú poskytované knižnicami operačného systému.

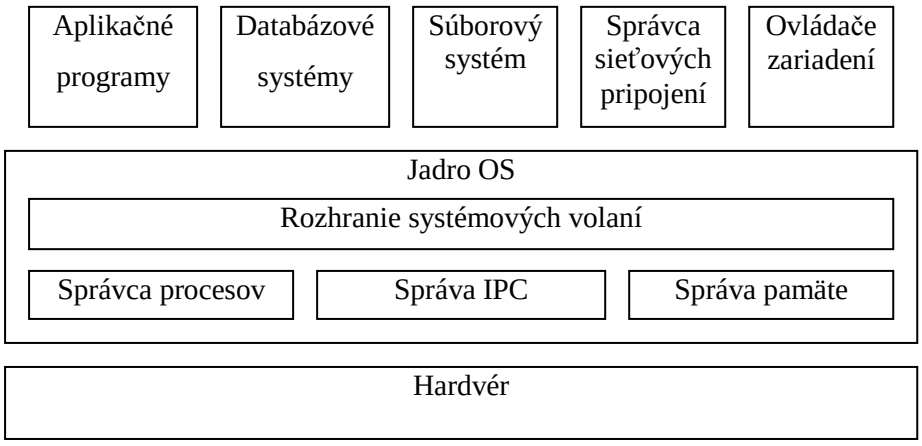
Adresový priestor jadra OS je oblasť pamäte, kde je spustené jadro a kde vykonáva svoje služby. Tento priestor je prístupný pre užívateľské programy iba prostredníctvom systémových volaní. Ak sú nároky jadra na tento priestor dostatočne malé a celý sa vojde do rýchlej cache pamäte procesora, výrazne sa tým zvýši výkon celého operačného systému.

2.3 Služby jadra operačného systému

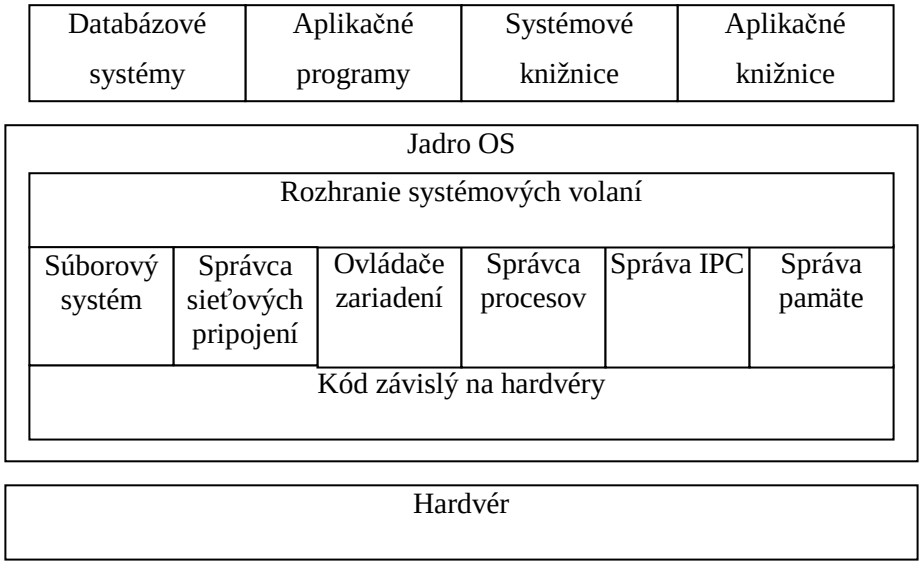
Aj keď sa jadrá značne líšia v rozsahu poskytovaných služieb, všetky poskytujú niektoré alebo všetky tieto služby:

- Správa procesov – prostriedok, ktorým OS riadi a pristupuje k ostatnému softvéru a prideľuje mu procesor. Ďalšia funkcia, ktorá často spadá pod správu procesov je detekcia výnimiek, prerušení a chýb.
- Správa pamäte – pamäťový priestor pre uchovávanie programu a programových dát. Pamäťový priestor je však pre všetky časti systému spoločný, preto sa prístup k nemu musí riadiť a regulovať. Bez obmedzení prístupu do pamäte by chyba v programe mohla spôsobiť poškodenie dát v inom programe alebo samotnom OS. Taktiež by mohlo dochádzať k úmyselným útokom či odpočúvaniam bez aplikovanej ochrany.
- Správa I/O systému - vstupno/výstupné zariadenia sa zdieľajú medzi viacerými procesmi, a preto je nutné zabezpečiť riadenie prístupu a alokáciu prostriedkov na prístup.

Delenie systému na výpočtovú, pamäťovú a vstupno/výstupnú je možné vidieť u von Neumanovej architektúry výpočtových systémov. Tá však popisuje fyzické časti na rozdiel od jadra OS. OS vytvára abstrakcie týchto fyzických častí. Tým, že jadro vytvára tieto abstrakcie, umožňuje tak ich zdieľanie medzi niekoľkými procesmi, z ktorých môže byť vytvorený aplikačný softvér.



Obrázok 2.4: Príklad architektúry systému s mikrojadrom L4



Obrázok 2.4: Príklad architektúry unixového systému s monolitickým jadrom [1]

2.4 Správca procesov

Správca procesov sa dá považovať za najdôležitejšiu časť operačného systému. Pozostáva z niekoľkých služieb ako vytváranie, pozastavovanie, spúšťanie, ukončovanie a odstraňovanie procesov.

Niekoľko súčasne aktívnych procesov je tak možné uchovávať v pamäti. Ak existuje v systéme iba jeden procesor, súčasne sa môže vykonávať iba jeden proces. Ostatné procesy sú pozastavené a čakajú na pridelenie procesora. Rozhodnutie, ktorý z procesov sa bude následne vykonávať je úlohou plánovača procesov. Dispečer pridelí procesor jednému procesu a odoberie druhému. Toto predávanie procesora medzi procesmi sa volá zmena kontextu. [5]

2.4.1 Plánovač procesov

Plánovač procesov má na starosti plánovanie, ktorý z procesov bude pri nasledujúcej zmene kontextu spustený. Plánovače používané v OS môžeme v zásade rozdeliť na preemptívné a nepreemptívné.

Pri nepreemptívnom plánovaní OS nemôže procesu násilne odobrať procesor, ale čaká až proces sám procesor uvoľní, bez ohľadu na čas využívania procesora či prioritu. V tomto prípade musí byť zabezpečené, že žiadny proces neuviazne v nekonečnej slučke.

Pri preemptívnom plánovaní si riadi čas zmeny kontextu samotný OS, a tak musí proces počítať so zmenou kontextu v ktoromkoľvek okamihu, alebo ho taktiež môže vyvolať.

Plánovače pri rozhodovaní, ktorý z procesov bude spustený používajú rôzne algoritmy. Ich škála je široká a každý z nich má svoje výhody a nevýhody. Špeciálnym prípadom sú plánovače real-rime operačných systémov, v nich sa pri plánovaní kladú špecifické požiadavky hlavne na dobu spracovania vstupných dát a reakciu na ne. Tieto systémy sú vhodné pre riadiace a kontrolné systémy.

2.5 Správca pamäte

Správca pamäte riadi prideľovanie a uvoľňovanie pamäte. Ukladá nariadenia a vytvára mechanizmy pre prácu s pamäťou ako aj pre jej ochranu. Stránkovanie, segmentácia a virtuálna pamäť sú podstatnými nástrojmi pre správu pamäte [9].

2.5.1 Virtuálna pamäť

Virtuálna pamäť je abstrakcia fyzickej pamäte. Virtuálna pamäť vystupuje ako logická vrstva medzi aplikačnými požiadavkami a jednotkou správy pamäte (MMU). Virtuálna pamäť prináša mnoho výhod ako napríklad:

- Súbežný beh viacerých procesov.
- Je možné spustiť aplikáciu s väčšími pamäťovými nárokmi než je fyzicky k dispozícii.
- Proces môže spúšťať kód ktorý je len čiastočne nahraný do pamäte.
- Je možné programovať aplikácie bez viazanosti na konkrétny hardvér.

Hlavným prostriedkom virtualizácie pamäte je virtuálny adresový priestor. Ten používajú procesy pri svojej činnosti, fyzické umiestnenie však môže byť v skutočnosti rozdielne. Ak proces pristupuje do virtuálnej pamäte procesor a MMU zaisťujú správne vyhľadanie fyzickej adresy, na ktorú sa má pristúpiť [9].

2.6 Medziprocesová komunikácia IPC

Medziprocesová komunikácia je hlavným prostriedkom k výmene informácií medzi jednotlivými procesmi. Procesy si navzájom pomocou IPC môžu vymieňať dáta, môžu sa pomocou nich synchronizovať a IPC taktiež vytvára prostriedky k bezpečnému zdieľaniu prostriedkov systému. Na tieto jednotlivé úlohy slúžia nasledujúce nástroje [9].

- Fronty správ – sú často implementovaným nástrojom medziprocesovej komunikácie. Prostredníctvom nich si procesy môžu vzájomne vymieňať dáta vo forme balíkov. OS v tomto prípade zabezpečuje doručovanie balíkov správnym procesom a taktiež definuje formu a množstvo balíkov pre jednotlivé fronty. Mikrojadrá často využívajú posielanie správ ako hlavný synchronizačný mechanizmus.
- Semafory – sa používajú predovšetkým pri zdieľaní pamäte ak je potrebný navzájom výlučný prístup a na synchronizáciu procesov. Systémy často podporujú viac druhov semaforov podľa potrebných vlastností, ako napríklad binárne semafory, mutexy, načítacie semafory.
- Signály – informujú proces o prebehnutej asynchrónnej udalosti. Sú generované buď softvérovo alebo hardvérom. Proces po prijatí signálu preruší vykonávanie aktuálneho programu a začne vykonávať obslužnú rutinu pre obsluhu daného signálu. Po jeho ukončení sa vráti na pôvodné miesto, kde bol program prerušený. Pre svoju asynchronnú povahu sa signály často využívajú na obsluhu prerušení.

2.7 Rodina mikrojadier L4

Mikrojadrá sú typické kladením dôrazu na minimálnu veľkosť kódu jadra a tým pádom kódu bežiaceho v privilegovanom režime procesora [11]. Typicky poskytujú len nevyhnutnú sadu služieb. Týmto prístupom sa dá dosiahnuť vysoká stabilita systému a odolnosť voči útokom. Stabilita je dosiahnutá vďaka malému rozsahu kódu, na ktorom je závislý beh celého systému. Takýto kód je možné dôkladne otestovať a formálne verifikovať na rozdiel od rozsiahlych monolitických jadier, v ktorých jediná chyba môže spôsobiť zlyhanie celého systému. Odolnosť je taktiež následkom vylúčenia mnohých častí z jadra systému, kedy pri útoku na ne alebo pri ich samotnom zlyhaní nie je ohrozený beh samotného jadra.

L4 je označenie rodiny mikrojadier druhej generácie. Pri návrhu konceptu L4 bol kladený dôraz na výkonnosť. Mikrojadrá prvej generácie sa totiž vyznačovali slabými výkonovými parametrami, a tým boli v značnej nevýhode oproti iným typom jadier. Analýzou chovania mikrojadra boli identifikované úzke miesta obmedzujúce výkon. To umožnilo zamerať pozornosť na čo najväčšiu priepustnosť týchto miest pri návrhu architektúry, ako aj pri jeho implementácii. Prvá implementácia L4 bola realizovaná Jochenom Liedtke, autorom predchádzajúceho mikrojadra L3. Bola napísaná v jazyku symbolických inštrukcií assembler pre dosiahnutie maximálneho výkonu [10]. Toto prvé funkčné mikrojadro s návrhom zameraným na výkon sa následne stalo predmetom záujmu mnohých inštitúcií, z ktorých niektoré sa zaoberajú vývojom a skúmaním mikrojadier dodnes. Tieto následne vyvíjané jadrá obsahujú v názve označenie L4 a tak tvoria spoločnú rodinu s rovnakým zameraním na výkon. V súčasnosti sa pracuje na vývoji jadier 3. generácie, ktoré sa mimo výkonu zameriavajú na bezpečnosť [12].

2.8 Všeobecné vlastnosti mikrojadra L4

Mikrojadro poskytuje minimálnu sadu funkcií, ktoré nad ním umožňujú vystavať komplexný a plnohodnotný systém. Popisované rozhranie a funkcie vychádzajú z najnovšej špecifikácie API rozhrania, označenej ako X.2. Základné abstrakcie, ktoré jadro ponúka sú nasledujúce:

1. Obsluha adresného priestoru.
2. Vytváranie vláken procesov.
3. Medziprocesovú komunikáciu IPC.
4. Unikátne identifikátory.

Tieto prostriedky umožňujú vytvárať ďalšie softvérové komponenty, ktoré už nemusia bežať v privilegovanom režime a pritom môžu plniť funkcie operačného systému. Servery [8] sú vlákna bežiace v užívateľskom režime procesora, ktoré poskytujú špecifickú službu. Servery, ktoré neplnia funkciu aplikačného softvéru ale poskytujú podporné služby pre aplikačný softvér môžeme zaradiť do operačného systému.

2.8.1 Pamäť a adresný priestor

K pamäti sa pristupuje prostredníctvom virtuálneho pamäťového priestoru. Ten je rozdelený na stránky, ktoré sú priebežne mapované na fyzickú pamäť RAM alebo ROM. Pomocou jednotky MMU sa virtuálne adresy prekladajú na adresy fyzické. Rovnako v L4 existujú aj virtuálne registre, ktoré poskytujú rozhranie na rýchlu výmenu dát medzi jadrom a vláknami. Virtuálne registre sú pevne namapované buď do fyzickej pamäte alebo priamo na fyzické registre. Existujú tri druhy virtuálnych registrov[8]:

- Registre na ovládanie vlákna – TCR.
- Registre na zasielanie správ – MR.
- Zásobníkové registre – BR.

L4 definuje špeciálne funkcie pre prácu s týmito registrami, prístup k nim iným spôsobom môže negatívnym spôsobom ovplyvniť fungovanie systému.

Rozhranie L4 umožňuje medzi jednotlivými pamäťovými priestormi daných procesov výmenu dát pomocou stránok. To je možné urobiť dvoma spôsobmi. Prvý umožňuje zdieľať danú stránku oboma procesmi. Funkcia sa nazýva MapItem. Druhým spôsobom môže proces stránku poskytnúť inému s tým, že sa vzdá prístupu k nej. Funkcia sa nazýva GrantItem.

2.8.2 Vlákna

Vlákno [8] je základná spustiteľná abstrakcia v L4. Vlákna v L4 sú nenáročné a jednoduché na správu. Spolu s rýchlou medziprocesovou komunikáciou vedú k efektívnosti celého systému. Vlákno je asociované s konkrétnym adresným priestorom. Každé vlákno má svoju sadu TCR registrov, ktoré uchovávajú stav vlákna. Časť z nich je určená pre samotné vlákno.

Úloha je súbor vláken zdieľajúcich spoločný adresný priestor. L4 taktiež rozlišuje medzi privilegovanými a neprivilegovanými vláknami. Privilegované sú také, ktoré zdieľajú adresný priestor s takým vláknom, ktoré bolo vytvorené jadrom počas bootovania. Niektoré systémové volania môžu volať iba privilegované vlákna.

2.8.3 Identifikácia prvkov

Vlákna sú identifikované jedinečným identifikátorom (UID). Vlákna môžu mať súčasne dva identifikátory, jeden lokálny a jeden globálny. Lokálny je jedinečný v rámci adresného priestoru. Globálny je jedinečný v rámci systému. Globálnym identifikátor môže byť použitý ľubovoľným vláknom, zatiaľ čo lokálny iba vláknom z rovnakého adresného priestoru. Na rozdiel od vláken, adresné priestory nemajú svoje identifikátory. Namiesto toho sa dá ľubovoľný adresný priestor identifikovať pomocou UID ľubovoľného vlákna patriaceho do tohto priestoru.

2.8.4 Komunikácia

Posielanie správ je základný spôsob komunikácie medzi vláknami a úlohami v L4, nazývaný taktiež ako medziprocesová komunikácia (inter-process communication – IPC). Posielanie správ umožňuje vláknam komunikovať s inými vláknami v rozdielnych adresných priestoroch. Toto posielanie správ je jadrom L4, slúži na prenos dát medzi vláknami. IPC je

v L4 blokujúce, to znamená, že každé poslanie správy musí vyvrcholiť v tzv. rendez-vouz, kedy posielajúce vlákno vykonáva funkciu vysielania správy a prijímacie práve funkciu príjmu. Preto je to aj dobrým nástrojom synchronizácie vláken. Časovo obmedzeným čakaním na príjem správy sa dá jednoducho docieliť uspanie vlákna na presne danú dobu [6].

Správy pozostávajú z dvoch častí, z povinnej a nepovinnej. Povinná je značka správy, tá obsahuje nastavenia správy a jej označenie. V nastaveniach je informácia o dĺžke správy a type dát, ktoré prenáša. Nepovinne môže správa prenášať typované a netypované dáta. Netypované slúžia na prenos ľubovoľných dát o veľkosti maximálne 63 Wordov, typované dáta na prenos reťazcov formou odkazu a na mapovanie pamäte. Jednotlivé správy sú prenášané prostredníctvom MR a BR registrov [8].

2.9 Dostupné systémy založené na L4

V tejto časti budú popísané dostupné implementácie L4 systémov. Mnoho z nich sa už nevyvíja alebo ich nahradili novšie projekty, ako sú napríklad L4Ka::Hazelnut, L4/MIPS a L4/Alpha [7].

OKL4 – je následník a pokračovateľ NICTA::Pistachio-embedded projektu, ktorý je komerčne vyvíjaný a podporovaný spoločnosťou Open Kernel Labs (OKL). Systém podporuje celú škálu procesorových architektúr a jeho variabilita ho umožňuje nasadiť od vstavaných systémov až po viacprocesorové podnikové systémy.

L4Ka::Pistachio – je posledné L4 mikrojadro vyvíjané skupinou System Architecture Group na Univerzite Karlsruhe v spolupráci s DiSy skupinou z Univerzity New South Wales, Austrália. Je to vôbec prvá dostupná implementácia jadra s L4 API verzie 4, ktorá je plne 32 a 64 bitovo čistá, podporuje viac procesorov a super rýchlu lokálnu IPC.

Fiasco – je implementácia mikrokernelu pre x86 architektúru s L4 binárnym rozhraním vytvoreným Michaelom Hohmuthom. Fiasco je navrhnuté ako preemptívne real-time jadro, a je základom pre DROPS systém. Vylepšenou verziou sa stalo mikrojadro s označením Fiasco.OC, ktoré bolo rozšírené o prístupovú politiku v podobe kapabilít a ďalšie rozšírenia. Pre toto jadro bolo následne vybudované prostredie L4Re, ktoré nahradzovalo prostredie vytvorené pre Fiasco, L4env. Pomocou týchto prostriedkov bol vytvorený operačný systém TUD:OS.

Codezero – je hypervízor postavený na mikrojadre L4. Podporuje ARM architektúru a virtualizuje systémy postavené na Linuxe a Androide. Je dostupný pod GPLv3 licenciou alebo komerčne. Systém je zameraný na virtualizáciu mobilných operačných systémov.

seL4 – je mikrojadro tretej generácie, ktoré stavia na výhodách mikrojadra L4, ako sú malá veľkosť, vysoký výkon a slobodná politika. Rozširuje jeho možnosti o vstavaný capability model, ktorý vytvára mechanizmy zaručujúce bezpečnosť na úrovni operačného systému ako aj aplikačnej úrovni.

L4.verified – sa zameriava na kompletnú verifikáciu jadra L4, to je matematického dôkazu, že implementácia zodpovedá špecifikácii. K tomuto účelu bolo zvolené seL4 jadro. Dôkaz pozostával z formalizácie API rozhrania seL4, ktorá bola následne použitá na samostatné overenie zhody vlastností.

2.10 Platforma L4 runtime enviroment

Platforma L4 runtime enviroment (ďalej len L4Re) vytvára základnú sadu služieb a abstrakcií. Tie sú užitočné pri implementácii a behu užívateľských aplikácií nad mikrojadrom Fiasco.OC.

2.10.1 Mikrojadro Fiasco.OC

Je najnižšie položený kód v rámci abstrakčnej úrovne bežiaci v rámci operačného systému L4. Mikrojadro je jediný program bežiaci v privilegovanom režime procesora. Nezahrňuje komplexné služby ako napríklad spúšťanie programov, radiče zariadení, či súborový systém. Tie sú implementované na úrovni užívateľských programov nad samotným mikrojadrom. Časť týchto služieb je obsiahnutá v L4Re.

Služby mikrojadra Fiasco.OC sú implementované formou objektov jadra. Úloha tak vlastní odkaz na objekt jadra vo svojej jadrom chránenej oblasti pamäte. Takýto odkaz sa v systéme označuje ako kapabilita. Kapabilita sa dá považovať za odkaz na volanie funkcie objektu v objektovo orientovanom prostredí. Ak úloha vlastní kapabilitu, môže ju poskytnúť inej úlohe s rovnakými alebo menšími právami na tento objekt.

Z pohľadu návrhu kapability prinášajú flexibilitu v systémových štruktúrach. Vlákno pristupujúce k službe prostredníctvom kapability nezaujíma, kde je služba implementovaná. Je tak možné vytvárať služby implementované v rámci jadra ako aj v rámci užívateľského priestoru. Preto je možné ich navzájom nahradzovať, či vymieňať bez toho, aby to klient pozoroval. Samotné jadro vytvára celú sadu kapabilít, ktoré následne dáva k dispozícii jednotlivým úlohám [14].

2.10.2 Popis L4Re

L4Re pozostáva zo sady knižníc a serverov. Rozhrania knižníc rovnako ako aj serverov sú kompletne objektovo orientované. Vytvárajú implementácie prototypov tried definovaných špecifikáciou L4Re. Knižnice a funkcionality pre beh procesov obsiahnuté v L4Re:

- knižnica jazyka C,
- pthreads ,
- libstdc++,
- zdieľané knižnice,
- infraštruktúra virtuálneho súborového systému,
- C a C++ prostredie,
- libsdl,
- infraštruktúra pre programovanie klient/server aplikácií.

Služby poskytované L4Re:

- spúšťanie programov pomocou skriptu,
- vstupne/výstupné radiče,
- správa platformy a zariadení vrátane ACPI a PCI,
- prostredie pre vytváranie radičov zariadení DDE,

- GUI multiplexor.

Systém s minimálnou konfiguráciou postavený na L4Re potrebuje tieto tri časti, aby bol spustiteľný:

- Fiasco – mikrojadro,
- Sigma0 – koreňový stránkovač,
- Moe – koreňová úloha.

Sigma0 počiatočne vlastní všetky systémové prostriedky, avšak obvykle sa Sigma0 používa hlavne na riešenie výpadkov stránok pre Moe koreňovú úlohu. Moe poskytuje dôležité služby normálnym užívateľským aplikáciám. Funguje napríklad ako zavádzač programov do pamäte. Ďalej slúži na mapovanie oblastí pamäte pri správe virtuálnej pamäte a ako alokátor pamäťového priestoru. [14]

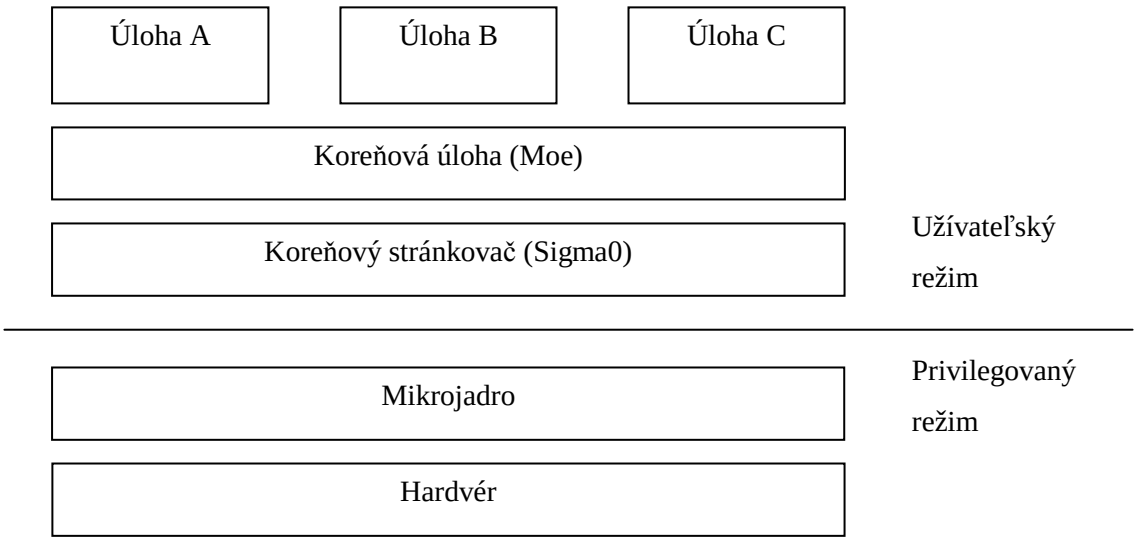
Medzi štandardné služby L4Re patria Ned, Io, Mag a Rtc. Tie však nie sú pre beh systému nevyhnutné.

Ned – je dôležitou službou L4Re. Ned je štandardný inicializačný proces, ktorý riadi konfiguráciu a spustenie reálneho systému. Spúšťa a konfiguruje systém na základe Lua skriptov.

Io – tvorí centralizovaný systém správy prostriedkov vzťahujúcich sa k hardvérovej platforme a periférnym zariadeniam. Vytvára prenositeľné abstrakcie pre periférne zariadenia a ich prostriedky. Zároveň sa stará o sprostredkovanie prístupu k týmto zdrojom iným aplikáciám, napríklad radičom zariadení.

Mag – je štandardný grafický multiplexor pre grafický hardvér. Mag umožňuje bezpečné multiplexovanie grafického a vstupného hardvéru medzi niekoľkými aplikáciami. Tieto aplikácie môžu obsahovať plnohodnotné grafické rozhranie.

Rtc – je štandardný multiplexor pre prístup k hardvéru reálneho času.



Obrázok 2.5: Základná štruktúra systému postavenom na L4Re [14]

3 Všeobecný návrh implementácie protokolu ACP

Táto časť sa venuje rozboru a návrhu implementácie ACP protokolu vo všeobecnej rovine. Rozhodol som sa najskôr vytvoriť úplný všeobecný návrh, pretože ACP je nový protokol, ktorý sa v praxi zatiaľ nepoužíva. Všeobecný úplný návrh je dobrým východiskom pre ďalšie možné implementácie. Za implementáciu protokolu sa považuje jeho podpora v systéme, to znamená, že systém s daným protokolom musí vedieť komunikovať. Aby to bolo možné, musí byť vytvorený softvér k tomu určený. Zo špecifikácie protokolu ACP vyplýva, že tento softvér sa pre tento protokol má označovať ako AC portál.

3.1 Špecifikácia AC portálu

Táto časť má za cieľ zhromaždiť známe informácie a požiadavky na AC portál, ktoré vyplývajú z popisu ACP protokolu.

- Portál pozostáva z jadra a z voliteľných modulov, ktorých môže mať viac rovnakého typu.
- Podpora komunikačných modulov. Tie umožnia portálu komunikovať prostredníctvom vybraného spoja napr.: spoje typu USB, TLS, UDP atď.
    - Musí riešiť stratu niektorej správy alebo prenosovú chybu.
- Podpora autentizačných modulov. Tie umožnia realizáciu vybrané autentizačné metódy, napr.: EAP-TLS.
- Podpora správnych modulov. Tie umožnia správcovi definovať algoritmus riadenia prístupu k vybranému aktívu daného zariadenia:
    - musí umožňovať definovať prístupovú politiku k jednotlivým aktívam,
    - v súlade s príslušnou metódou riadenia prístupu definuje chovanie modulu v priebehu transakcie a obsahuje informácie potrebné k vykonaniu transakcie napr.: vyžadované autentizačné metódy, smerovacie údaje a pod.,
    - musí definovať sled správ pre danú transakciu.
- Portál musí korektným spôsobom dekódovať a generovať správy ACP protokolu.
- Musí komunikovať pomocou striedavo prenášaných správ.
- Musí platiť, že komunikáciu vždy riadi poskytovateľ.
- Musí byť schopný tranzitovať transakcie.
- Portál musí byť schopný generovať jedinečné hodnoty Identifier pre ním vytvárané transakcie.
- Musí po prijatí správy Start pre danú transakciu spustiť časovač, ktorý sa nuluje s príchodom každej novej správy danej transakcie.
- Musí ukončovať transakciu výmazom všetkých informácií súvisiacich s touto transakciou.

- Musí ukončiť transakciu ak:
    - zistí nesprávnu dĺžku dát ACP správy,
    - nedorazí požadovaná správa do vypršania časovača a na ďalšie správy tejto transakcie portál nebude reagovať,
    - ak dorazí požadovaná správa viackrát,
    - ak prijme AVP správu, ktorú nedokáže spracovať.
- Musí definovaným spôsobom na základe údajov pre budovanie spojenia a údajov v správe Start zistiť, ktorému portálu má danú správu Start tranzitovať.
- Musí poskytovať možnosť existenciu viacerých aktív.
- Musí správcovi umožniť zadefinovať lokálne kódy prenášané pomocou AVP štruktúr.
- Portál musí implementovať aspoň ACP-VSA variantu ACP protokolu, keďže sa jedná o východziu variantu.

Vymenované požiadavky na portál by mali spĺňať predovšetkým implementácie AC portálu poskytujúcich aktíva. AC portál, ktorý je umiestnený na koncovom zariadení, ako napríklad čipová prístupová karta, môžu obsahovať zjednodušenú implementáciu AC portálu, keďže nemusia obsahovať nástroje umožňujúce tranzit transakcií.

3.2 Analýza a návrh

V tejto časti budú analyzované požiadavky zapísané v špecifikácií. Budú zohľadnené ich následky, navrhnuté riešenia a alternatívy. Následný návrh má za úlohu vyčleniť jednotlivé logické časti portálu, ktoré sa následne budú podrobnejšie analyzovať. Návrh bude všeobecný, preto bude použiteľný pre ľubovoľnú platformu či programovací jazyk.

Portál bude tvorený z niekoľkých vzájomne previazaných a komunikujúcich častí, čo vyplýva z požiadaviek zásuvných modulov. Jednotlivé moduly sa po spustení spoja s jadrom ACP. To bude v systéme vždy len jedno. Systém pozostáva z nasledujúcich častí:

- Jadro portálu,
- Správny modul,
- Komunikačný modul,
- Autentizačný modul.

V tejto práci bude pozornosť sústredená iba na variantu portálu na strane poskytovateľa služieb. Na strane žiadateľa bude portál čiastočne odlišný. V nasledujúcich odstavoch bude urobený podrobnejší rozbor jednotlivých častí. Najskôr bude popísané jadro portálu a potom jednotlivé moduly.

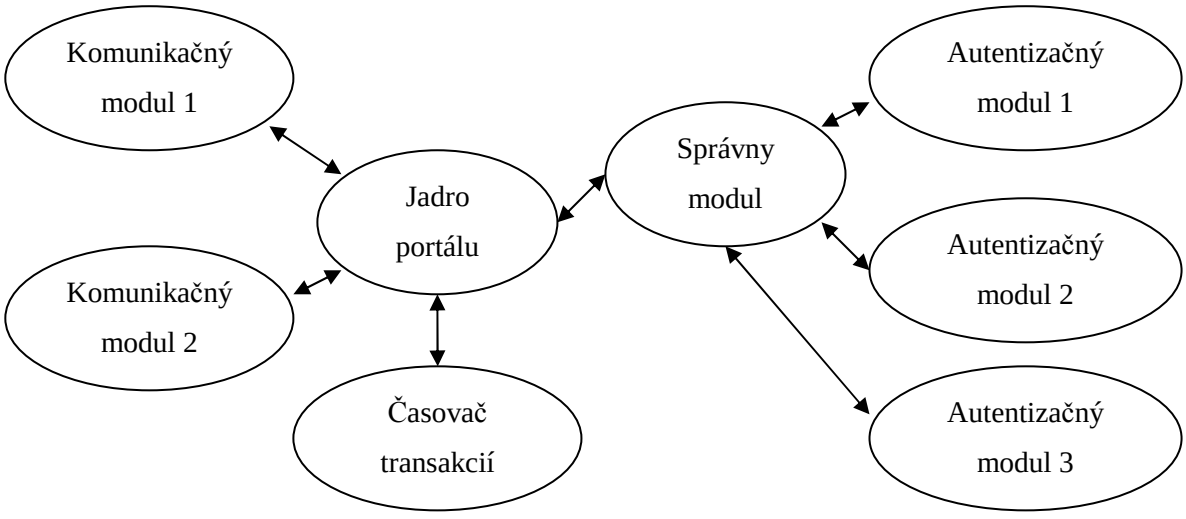
3.2.1 Jadro portálu

V špecifikácií sa jadro priamo nespomína, ale všetko, čo nepatrí do modulov bude patriť do jadra systému. Zároveň bude jadro centrálnym prvkom komunikácie. Cez jadro pôjdu všetky prichádzajúce a odchádzajúce dáta.

Jadro bude vytvárať dátové štruktúry popisujúce jednotlivé transakcie tzv. deskriptory a spravovať ich. Novým transakciám bude generovať jedinečné identifikátory pomocou generátoru náhodných čísel. Bude predávať prijaté dáta do jednotlivých typov modulov. Ďalej bude generovať ACP správy a predávať ich komunikačným modulom na odoslanie. Prijaté ACP správy bude kontrolovať a nezhody patričným spôsobom riešiť. Najčastejšie zrušením transakcie.

Časovač nebude súčasťou hlavnej obsluhy jadra, ale bude to samostatný proces. Ten sa bude pravidelne v rovnaných časových intervaloch aktivovať a inkrementovať časovače jednotlivých transakcií. Hlavná časť jadra bude pri každej prijatej správe danej transakcie jej časovač nulovať. Pokiaľ by časovač presiahol hraničnú hodnotu pre dané rozhranie jadro transakciu zruší tým, že ju zaradí do fronty zrušených transakcií. Tam budú iba deskriptory a ostatné dáta sa zmažú. Zrušené transakcie tam budú čakať ešte istú dobu, než budú úplne zmazané. Toto opatrenie splní požiadavku na dočasné nereagovanie na zrušenú transakciu. Pokiaľ transakcia skončí korektne bude okamžite celá zmazaná.

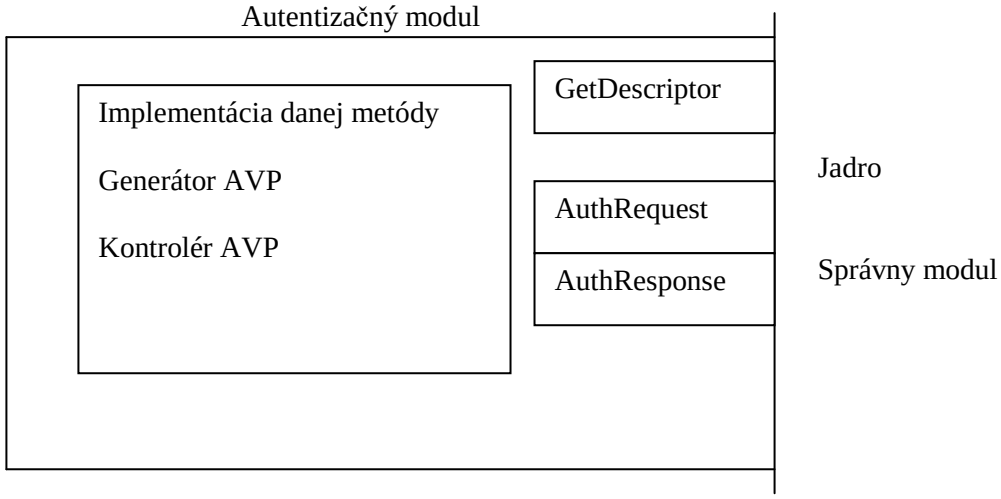
Jadro bude obsahovať rozhrania k všetkým druhom modulov. Prístup k jednotlivým modulom bude prevádzaný prostredníctvom zoznamov modulov. Tie budú súčasťou jadra a na ich základe bude jadro s jednotlivými modulmi komunikovať. Pre každý typ modulov bude existovať samostatný zoznam. Na ich základe bude taktiež jadro schopné zistiť svoju prevádzkyschopnosť.



Obrázok 3.1: Komunikačné väzby medzi jednotlivými časťami portálu

3.2.2 Autentizačný modul

Autentizačné moduly musia umožniť realizáciu vybrané autentizačné metódy, napr.: EAP-TLS. To je ich hlavná a jediná úloha. Návrh autentizačného modulu je na nasledujúcom obrázku.



Obrázok 3.2: Štruktúra Autentizačného modulu

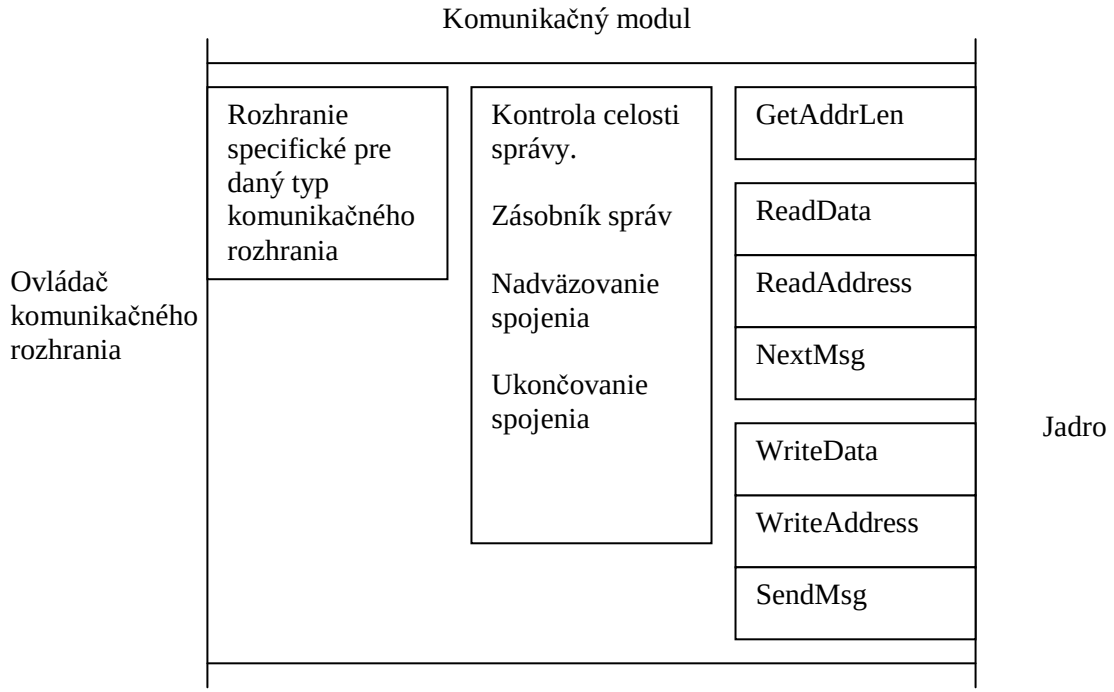
Každý autentizačný modul bude schopný na základe požiadavky AuthRequest zo správneho modulu spustiť proces autentizácie. Po prijatí tejto správy poznačí do deskriptora informáciu pre jadro o fáze autentizácie tejto transakcie. V prípade potreby získania ďalších informácií od žiadateľa modul posiela správy priamo jadru, aby sa zrýchlilo a zefektívnilo fungovanie portálu. Po skončení autentizácie pošle konečný výsledok vždy správnemu modulu s výsledkom autentizácie.

Jadro po prijatí správy Response predá jej AVP autentizačnému modulu správou AuthRequest. Autentizačný modul na základe úplnosti údajov vyplnených v deskriptore a prijatých AVP je schopný riadiť proces autentizácie. Autentizačný modul v stave pred odoslaním správy AuthResponse skontroluje, či spracoval všetky prijaté AVP. Ak nie vráti chybu. Jadro správy prijaté od modulu vkladá do ACP správy typu Request a odosiela. Ak autentizačný modul pri spracovávaní AVP narazí na neznáme AVP vráti do správneho modulu informáciu o chybe. Metódou GetDescriptor je možné zistiť, či je modul aktívny, a ktorú transakciu práve obsluhuje.

Rozhranie autentizačného modulu bude teda spoločné pre jadro a správny modul. Modul bude obsahovať ďalšie rozhrania a to rozhranie na prístup k zoznamu užívateľov a overovacím faktorom. Toto rozhranie sa však môže meniť pre jednotlivé implementácie modulu. Toto rozhranie môže byť riešené prostredníctvom volaní do databázy alebo aj lokálnym zoznamom. Záleží na konkrétnej metóde a module.

3.2.3 Komunikačný modul

Zo špecifikácie ACP plynú pre komunikačný model nasledujúce úlohy. Musia umožniť portálu komunikovať prostredníctvom vybraného spoja napr.: spoje typu USB, TLS, UDP, EAPoL. Musia riešiť stratu niektorej správy alebo prenosovú chybu. Návrh modulu je na nasledujúcom obrázku.



Obrázok 3.3: Štruktúra komunikačného modulu

Bolo by možné, aby existovali komunikačné moduly pre viacero rozhraní, to by však zvyšovalo nároky na rozhranie k jadru a zároveň by sa mohlo stávať, že zvolená sada rozhraní v jednom module nebude využitá celá a bude zbytočne zaťažovať systém. Preto je vhodné, aby pre jednotlivé rozhrania existovali samostatné komunikačné moduly. Každé rozhranie je niečím špecifické. Bude to vyžadovať možnosť nezávislého nastavenia jednotlivých modulov samostatnými konfiguračnými súbormi, ktoré spracujú jednotlivé moduly. Jednotlivé portály môžu byť namapované na fyzické rozhrania alebo virtuálne. Jednotlivé rozhrania sa budú zabezpečovať dáta proti chybám v takom rozsahu, v akom to vykonáva samotný prenosový protokol, ktorý využívajú.

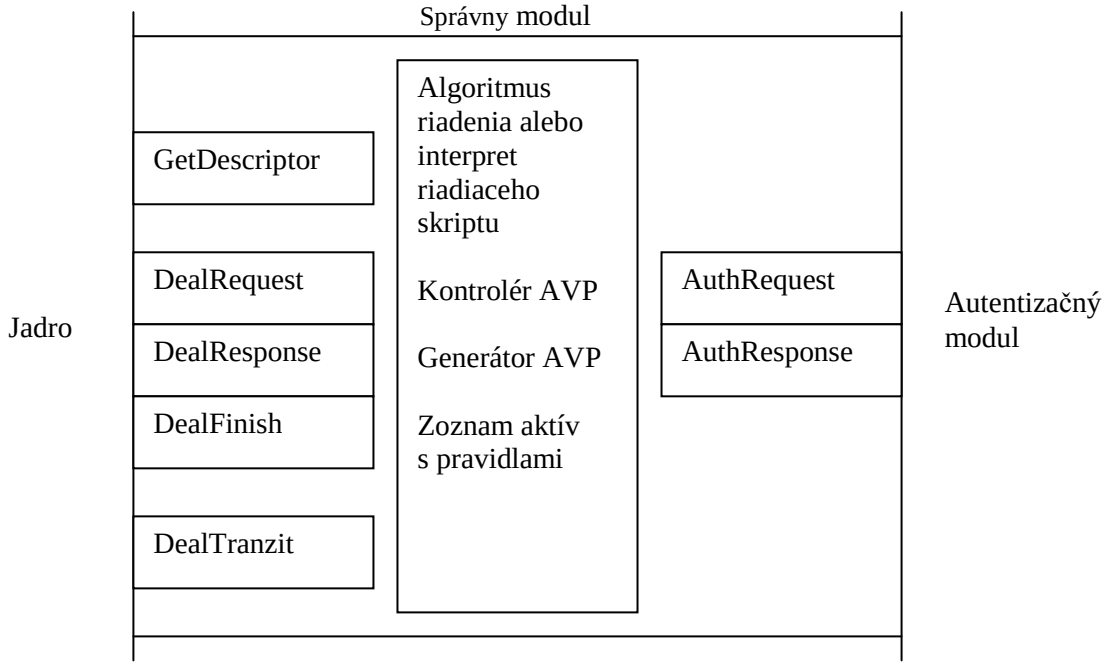
Rozhranie pre komunikáciu s jadrom bude umožňovať prenos celých ACP správ. Jednotlivé moduly budú mať dve rozhrania. Jedno pre vysielanie a druhé pre príjem ACP správ.

Prijímacie rozhranie na strane jadra bude implementovať metódu NextMsg, ktorá vráti dĺžku prijatej ďalšej správy. Ďalším zavolaním NextMsg sa aktuálna správa zahodí. Týmto spôsobom bude môcť jadro zahadzovať správy s príliš veľkou dĺžkou. Následne pomocou volania ReadData vyčíta dáta do poskytnutej pamäte. Rovnakým spôsobom sa vyčíta adresa.

Volaním SendMsg a nezadaním adresy sa pripraví modul na odosielanie dát. Následným nahraním dát a adresy sa po zavolaní SendMsg vykoná samotné zaradenie do fronty na odoslanie dát. Modul je následne pripravený na ďalšie odosielanie.

3.2.4 Správny modul

Na správny modul je v špecifikácii kladených najviac požiadaviek. Musí umožniť správcovi definovať algoritmus riadenia prístupu k vybranému aktívu daného zariadenia. Musí umožniť zadefinovať prístupovú politiku k jednotlivým aktívam. V súlade s príslušnou metódou riadenia prístupu definuje chovanie v priebehu transakcie. Obsahuje informácie potrebné k vykonaniu transakcie, napr.: vyžadované autentizačné metódy, smerovacie údaje a pod. Návrh tohto modulu znázorňuje nasledujúci obrázok.



Obrázok 3.4: Štruktúra správneho modulu

Na základe týchto požiadaviek je zrejmé, že sa bude jednať o najzložitejší typ modulu. Keďže riadi celkový priebeh transakcie, tak sa musí v portáli nachádzať vždy minimálne jeden takýto modul. Môže ich však obsahovať viac. Jeden z nich bude stanovený ako počiatočný, ktorému sa budú posielať dáta po prijatí správy Start. Tento počiatočný modul môže následne predať funkciu riadenia transakcie inému správnemu modulu podľa v ňom definovaných pravidiel. Správny modul bude riadiť proces vyjednávania požadovaného aktíva ako aj autentizačnej metódy. Správne moduly budú štandardne samostatné procesy, ktorých chovanie je pevne dané v programe. Bude však možné vytvoriť aj programovateľný správny modul, ktorý bude fungovať ako interpret vhodného skriptovacieho jazyka. Metódou GetDescriptor bude možné zistiť, či je modul aktívny, a ktorú transakciu práve obsluhuje.

Správny modul bude mať dva štandardné rozhrania. Jedno smerom k jadru a druhé k autentizačným modulom. Rozhranie k jadru budú využívať správne moduly aj medzi sebou na predávanie riadenia. Ďalšie rozhrania budú slúžiť na prístup k zoznamu aktív, prístupovým právam a autentizačným metódam. Ďalej k štruktúre záznamov práce s aktívami, tzv. účtovacie záznamy.

3.2.5 Transakcie

Komunikácia medzi jednotlivými modulmi a jadrom umožní fungovanie portálu a obsluhu transakcií. Jednotlivých scenárov, ktoré môžu nastať je veľké množstvo. Niektoré sú zachytené v sekvenčných diagramoch pozri príloha 2 a niektoré sú podrobnejšie popísané v tejto kapitole.

Prijatím správy Start z komunikačného rozhrania jadro skontroluje v zozname existujúcich transakcií pre dané rozhranie, či už daná transakcia neexistuje. Ak áno, tak bude daná transakcia považovaná za útok a bude zrušená zaradením do fronty zrušených transakcií. Pokiaľ daná transakcia neexistuje bude vytvorená. Transakcia je definovaná deskriptorom, ktorý ju popisuje. Obsahuje všetky potrebné informácie pre priebeh transakcie. Komunikačný modul sa ukladať nemusí, pretože zoznamy sú zviazané s jednotlivými komunikačnými modulmi. Takto pripravený deskriptor predá správnemu modulu správou DealRequest spolu s AVP štruktúrami, ktoré boli so správou prijaté. Ak je správcom v danom správnom module povolený príjem AVP v správe Start, správny modul začne proces vyjednávania. Inak vráti deskriptor správou DealResponse jadru s informáciou, že transakcia zlyhala spolu s prípadnými AVP o dôvode ukončenia transakcie. Jadro vtedy vygeneruje správu Finish a dané AVP do nej vloží.

V prípade úspešného spustenia vyjednávania jadro po prijatí správy DealResponse vygeneruje ACP správu Offer a pošle ju s AVP žiadateľovi. Správny modul môže predať riadenie autentizačnému modulu alebo inému správnemu modulu. Urobí tak predaním deskriptoru danému modulu. Ten sa k danému deskriptoru prihlási a prípadne si môže poznačiť, od koho a v akom stave dostal deskriptor. Implementáciou zásobníka v deskriptore by bolo možné hlboké vnáranie v predávaní riadenia.

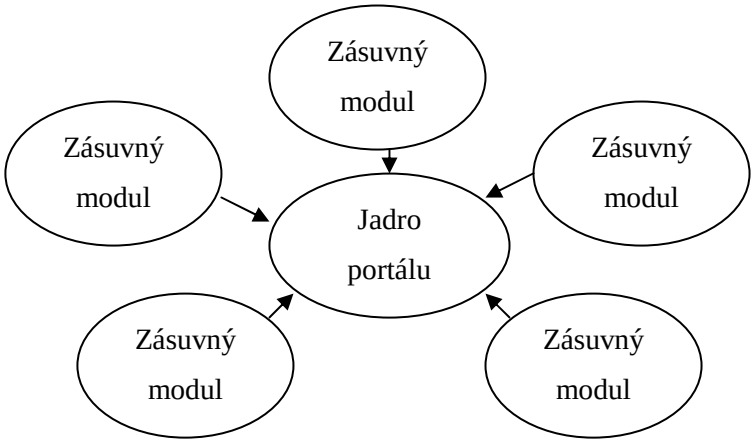
3.2.6 Tranzitovanie transakcií

Jadro po prijatí správy DealTranzit od správneho modulu overí, či je transakcia nová. To znamená, že nemá vypršaný časovač a má prázdny deskriptor. Ak áno vytvorí nový špeciálny deskriptor pre tranzitovanie. Ten sa na základe informácií od správneho modulu pripojí do zoznamu transakcií na nové rozhranie, kde sa bude správa tranzitovať. V tomto deskriptore bude krížový odkaz na deskriptor pôvodnej transakcie. Následne vygeneruje správu Start. Vygeneruje identifikátor novej transakcie, vloží do nej AVP od správneho modulu a prepošle na nové rozhranie. Ak sa vráti odpoveď, tak sa bude transakcia vyhľadávať v zozname tranzitných deskriptorov. Po jeho nájdení sa na základe krížového odkazu dostane k pôvodnému deskriptoru a tým pádom k adrese pôvodného odosielateľa, na ktorú bude správa preposlaná.

3.2.7 Zavádzanie portálu a jeho spustenie

Spustenie portálu bude riadené správcom definovaným konfiguračným súborom. V ňom bude zadefinovaný počet a typ jednotlivých modulov, ktoré budú pripojené na portál. Ako prvé bude spustené vlákno jadra portálu, ktoré na základe konfiguračného súboru zistí počty, typy jednotlivých portálov a ich umiestnenie napríklad na lokálnom úložisku. Ďalej

v ňom budú informácie potrebné na overenie identity jednotlivých portálov. Pomocou týchto informácií umožní ich pripojenie a spoluprácu s jadrom. Po pripojení a overení identity aspoň jedného komunikačného a správneho modulu sa aktivuje obsluha ACP protokolu.



Obrázok 3.5: Zásuvné moduly iniciujú komunikáciu a registrujú sa v jadre.

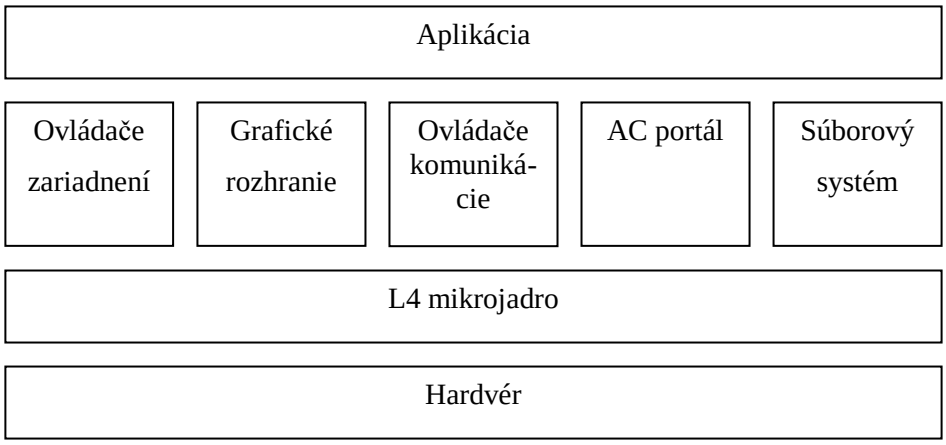
Jednotlivé moduly môžu spadať do bezpečnostne kritickej oblasti softvérovej výbavy zariadenia. Preto ich bude potrebné zabezpečiť proti nežiaducim zmenám. Ich kód bude musieť byť digitálne podpísaný nejakou dôveryhodnou autoritou, s ktorou sa dokáže samotné jadro spojiť. Jednotlivé moduly po svojej aktivácií pristúpia k procesu vlastnej autentizácie. Tým jadro zistí, že modul je pripravený a po jeho autentizovaní ho zapíše do zoznamu modulov a začne ho používať. V prípade pridania modulu za behu systému, požiadaním o vlastnú autentizáciu sa prihlási jadru a jadro ho môže dynamicky pridať do zoznamu. V jadre budú implementované dva zoznamy modulov. Jeden pre komunikačné moduly a druhý pre moduly správne a autentizačné. Oba zoznamy budú obsahovať rovnaké položky obsahujúce označenie modulu, typ modulu a jeho označenie vlákna UID. U komunikačných modulov bude UID globálne, kým u ostatných bude UID lokálne. Lokálne UID má výhodu v tom, že pomocou neho sa môže pristupovať k vláknu iba v rámci daného adresného priestoru. Preto nehrozia útoky na dané moduly inými vláknami z ostatných adresných priestorov. Správne moduly budú mať tento zoznam k dispozícii, aby na jeho základe mohli predávať riadenie transakcie buď autentizačným alebo iným správnym modulom.

4 Implementácia AC portálu do operačného systému L4

Táto kapitola sa bude zaoberať konkrétnou implementáciou a využitím služieb mikrojadra L4 v AC portáli. Implementáciou AC portálu sa rozšíria možnosti využitia operačného systému. Na začiatok bude urobený rozbor možností implementácie. Z nich bude jedna vybratá a implementovaná do zvolenej varianty operačného systému L4. Bude bližšie popísané implementačné prostredie a nástroje použité na preklad zdrojových kódov. Na konci bude popis testovania a overenia funkčnosti vytvorenej implementácie.

4.1 Možnosti implementácie

V tomto prípade bude za OS považované mikrojadro L4 s rozširujúcimi servermi plniacimi úlohy štandardne zaradzované do operačného systému ako napr.: ovládače komunikačných rozhraní. Navrhovaná implementácia sa bude zameriavať na jej rozšíriteľnosť, tak ako je to zo špecifikácie požadované.



Obrázok 4.1: AC portál ako server mikrojadra L4

Existuje niekoľko variant, kde by v rámci architektúry systému postavenom na L4 mikrojadre mohol byť AC portál implementovaný. Mohol by byť implementovaný ako:

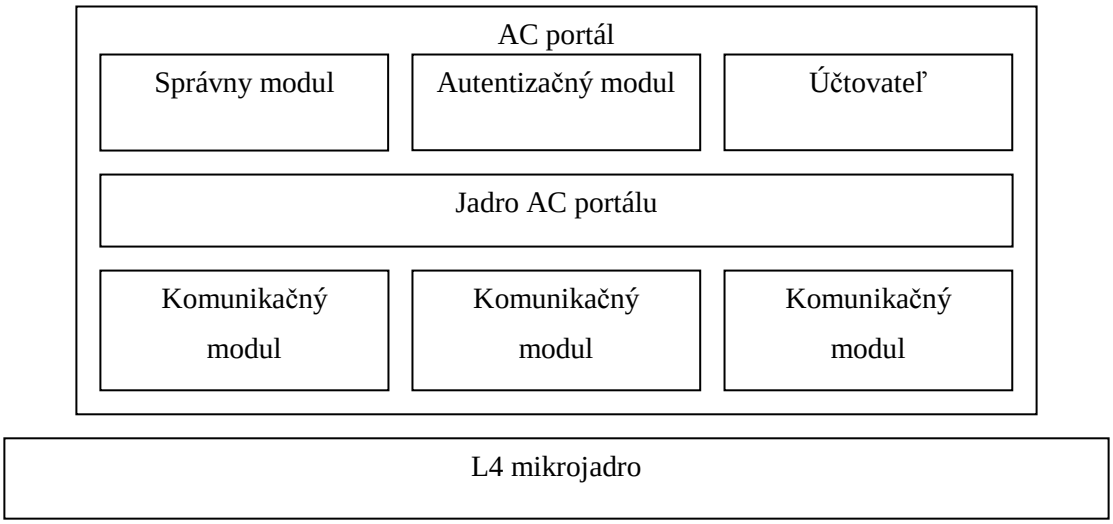
- súčasť jadra,
- server,
- aplikácia.

Pokiaľ by sa mala implementácia urobiť do samotného jadra operačného systému, s najväčšou pravdepodobnosťou by sa nezanedbateľným spôsobom zvýšila pamäťová náročnosť jadra. Čím by sa zhoršila možnosť ho celé umiestniť do procesorovej cache pamäte a tým by utrpela jeho primárna vlastnosť - výkon. Ďalším problémom, s ktorým by sa pri implementácii tejto varianty muselo počítať je, že samotné jadro štandardne nepodporuje

žiadne komunikačné rozhranie. Tak by bola táto funkcionalita v samotnom jadre závislá na existencii komunikačných serverov, čo by prevrátilo závislosti jadra a serverov. Ďalším problémom, ktorý s touto variantou súvisí je, že jadro beží v privilegovanom režime. Pri chybe, či útoku na AC portál by tak mohlo dôjsť k znefunkčnení celého jadra a tým pádom aj celého systému.

Ďalšou možnosťou je implementácia AC portálu ako serveru OS. Servery obvykle vytvárajú balík služieb, ktorý je v monolitických systémoch súčasťou jadra. Tie medzi sebou komunikujú pomocou IPC. Servery už nie sú súčasťou jadra, preto pri ich znefunkčnení dochádza iba k strate konkrétnej služby a systém sa môže z tejto situácie zotaviť. Jednotlivé komunikačné rozhrania sú taktiež implementované ako servery, preto táto varianta nenarušuje koncept operačného systému.

Poslednou možnosťou je implementácia AC portálu ako aplikácie. Aplikácie sa v systéme s mikrojadrom odlišujú od serverov predovšetkým v tom, že neposkytujú ďalším aplikáciám žiadne služby. Preto sa implementácia AC portálu bude vzhľadom k jej zameraniu poskytovať služby a vystupovať v systéme ako server.



Obrázok 4.2: Architektúra AC portálu

4.2 Výber operačného systému L4

Keďže samotné L4 mikrojadro obsahuje len tie najzákladnejšie funkcie, nedá sa samotné používať ako plnohodnotný OS, ale iba s ďalším rozširujúcim softvérom. Preto sa musí nájsť vhodná implementácia samotného jadra, pre ktoré existujú dostatočné podporné nástroje, aby sa dalo pre implementáciu ACP protokolu použiť a následne otestovať.

Za najvhodnejšiu implementáciu bolo vybrané mikrojadro Fiasco.OC vyvíjané Technickou univerzitou v Drážďanoch. Toto jadro poskytuje dobrú podporu štandardne rozšírených procesorov. Má veľmi dobre spracovanú vývojovú dokumentáciu a je k dispozícii emailová podpora. Ďalšou výhodou je existencia aplikačného prostredia L4Re. Táto nadstavba označovaná rozširuje mikrojadro o možnosť načítavania programov do pamäte, pokročilejšiu správu pamäte. Ďalej implementuje prostredie pre beh aplikačných

programov, ktorým poskytuje rôzne služby a umožňuje im zdieľať knižnice. Dôležitým argumentom pre výber boli tiež dostupné ukážkové programy spolu s návodmi na inštaláciu a spustenie.

4.3 Príprava implementačného prostredia

Táto časť sa venuje príprave prostredia pre samotnú implementáciu AC portálu.

Ako vývojové prostredie bol zvolený operačný systém Linux. Keďže vývoj pre L4 operačný systém vyžaduje GNU nástroje Binutils, prekladač, Perl a Make a v doporučaniach k inštalácii L4Re uvedené že balík bol testovaný na distribúciách Debian 6.0, Ubuntu 10.10 a novších. Na preklad sa dá taktiež použiť ľubovoľný BSD systém. Ako najvhodnejšia z rady linuxových distribúcií vyšla verzia Ubuntu 11.10. Jednoduchá inštalácia pomocou Live USB a užívateľsky prijateľná obsluha systému umožnila rýchlejšie napredovanie aj bez hlbších znalostí prostredia. Po inštalácii bolo z internetu stiahnuté behové prostredie L4 Runtime Enviroment (L4Re), ktoré mimo samotného mikrojadra Fiasco.OC obsahuje nástroje. Prostredie pre vývoj aplikácií nad týmto jadrom a inštalačných sprievodcov. Pre ďalšiu prácu je potrebné mať užívateľské práva na úrovni užívateľa root.

L4Re je možné získať z internetu zadaním nasledujúceho príkazu, kde `somedir` je ľubovoľný adresár:

```
somedir$ svn cat http://svn.tudos.org/repos/oc/tudos/trunk/repomgr | perl - init http://svn.tudos.org/repos/oc/tudos fiasco l4re
```

alebo prostredníctvom prehliadača zadaním adresy:

http://os.inf.tu-dresden.de/download/snapshots-oc/l4re-snapshot-2011081207.tar.bz2

Zadaním tohto príkazu bude stiahnutý skomprimovaný súbor, ktorý po rozbalení obsahuje nasledujúce položky:

- Fiasco.OC – mikrojadro L4
- L4Re – sada knižníc a rozšírení pre mikrojadro Fiasco.OC
- L4Linux – Linux upravený pre beh nad Fiasco.OC ako samostatný proces
- Ďalší softvér ako napríklad Fiasco-UX, jadro upravené pre beh nad Linuxom.

4.3.1 Nastavenie a preklad balíku L4Re

Existuje niekoľko spôsobov ako sa dá dosiahnuť nastavenie, preklad a zostavenie kódu určeného pre spúšťanie L4Re spolu s Fiasco.OC. Je možné si napríklad zvoliť z rady oficiálne podporovaných procesorových architektúr. Najuniverzálnejšia z nich je architektúra x86, preto bude pre túto architektúru celý balík preložený a následne nad ním implementovaný AC portál.

K samotnému prekladu balíka potrebujeme v Linuxe nasledujúce nástroje a programy: make, gcc, g++, ld, gawk, perl, pkg-config. Po inštalácii Ubuntu v systéme chýbali g++ a gawk. Tie sa doinštalujú zadaním príkazu do príkazového riadka:

```
somedir$ sudo apt-get install gawk g++
```

Po úspešnej inštalácii týchto prostriedkov sa musí nakonfigurovať preklad balíka. Nasledujúci príkaz musí byť spustený z najvrchnejšieho adresára rozbaleného balíka. V tomto adresári sa nachádzajú podadresáre bin, doc, files, src.

```
L4redir$ make setup
```

Po zadaní príkazu sa v prípade nainštalovaného dialogového programu zobrazí prostredie pre konfiguráciu prekladu. V našom prípade necháme všetko nastavené a potvrdíme konfiguráciu. Tá bude trvať približne minútu. Ak pri prebiehajúcej konfigurácii nastanú chyby, je potrebné urobiť nasledujúcu zmenu a následne spustiť príkaz znova. Zmena sa bude týkať súboru Makefile s nasledujúcim umiestnením:

```
L4redir/src/l4/tool/gendep/Makefile
```

V tomto súbore je potrebné riadok s obsahom:

```
LIBDL-linux := -ldl
```

Nahradiť riadkom s obsahom:

```
LIBDL-linux := -Wl,--no-as-needed -ldl
```

Následne súbor uložte, vráťte sa do koreňového adresára balíku `L4redir` a spuste príkaz `make setup` znova. Po jej úspešnom ukončení nastavení môžeme spustiť samotný preklad:

```
L4redir$ make
```

Preklad trvá zhruba 20 minút, v závislosti na použitom počítači. Preložené binárne súbory sa nachádzajú v adresári s menom obj.
Následne bude vytvorený adresárový strom určený pre implementáciu AC portálu a ďalších dokumentov s ňou súvisiacich na základe vzorovej stromovej štruktúry L4Re. Vytvorený adresárový strom umiestnime do koreňového adresára L4redir.

Samotný preklad AC portálu sa vykoná zavolaním nasledujúceho príkazu z koreňového adresára AC portálu.

```
ACPpkg$ make O=../obj/l4/x86
```

Po bezchybnom preložení sa vytvorí binárny preložený súbor s menom acportal. Jeho spustenie je popísané v kapitole venovanej testovaniu.

4.4 Implementácia AC portálu

Implementácia AC portálu pozostáva z jeho popisu a samotnej realizácie. Bude vytvorená minimálna funkčná implementácia s možnosťou aplikácie v koncových prenosných zariadeniach slúžiacich na identifikáciu, ako napríklad prístupové čipové karty. Táto implementácia nebude obsahovať nástroje na tranzitovanie transakcií, keďže sa predpokladá že čipové karty nebudú transakcie postupovať ďalším systémom. Vylúčením tranzitovania z implementácie sa systém podstatne zjednotuší. Na druhú stranu bude podporovať možnosť pripojenia cez viacero komunikačných rozhraní, čo by mohlo byť využité u čipových kariet so súčasným elektrickým aj rádiovým rozhraním. Ďalej bude implementované všeobecné rozhranie na súčasné zapracovávanie viacerých transakcií.

Tento systém bude implementovať iba služby na strane poskytovateľa služieb, takže nikdy nebude iniciovať žiadnu transakciu. Preto pre otestovanie tohto systému bude potrebné realizovať samostatnú aplikáciu vytvárajúcu stranu žiadateľa.

Keďže sa jedná o server, portál nebude obsahovať časti potrebné na jeho ukončenie, keďže sa predpokladá jeho činnosť počas celého behu systému. Všetky vlákna budú v sebe obsahovať nekonečné slučky, preto budú existovať po celú dobu behu AC portálu.

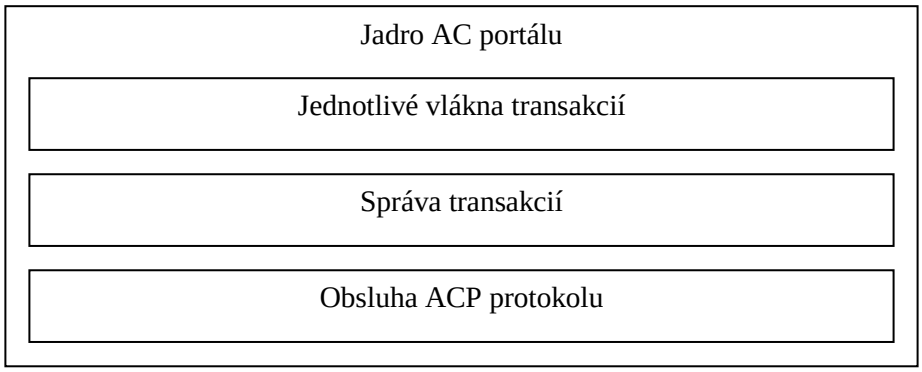
Samotná implementácia bude realizovaná v programovacom jazyku C. Program napísaný v tomto jazyku sa rozdeľuje do modulov, ktoré sú reprezentované jednotlivými súbormi. Implementácia AC portálu tak bude pozostávať z nasledujúcich programových modulov:

- Acportal.c
- Transaction.c
- CommModule.c
- PolicyModule.c
- AuthenModule.c

Ďalšie vytvorené programové moduly budú potrebné pre samotné spustenie a otestovanie funkčnosti implementácie:

- main.c
- TestCommDriver.c

Vzhľadom k tomu, že L4Re neobsahuje podporu TCP/IP protokolu a tým pádom pripojenie do siete, bolo navrhnuté simulované pripojenie, ktoré bude bežať priamo vo vytvorenom serveri. Týmto spôsobom bude možné jednoducho otestovať funkčnosť predovšetkým AC portálu bez ohľadu na použité komunikačné rozhranie. Zároveň bude v komunikačnom serveri (commdrive.c) implementovaná simulovaná strana žiadateľa. Keďže všetky servery nad jadrom medzi sebou komunikujú prostredníctvom IPC a nie sú ničím iným navzájom previazané, bude sa dané simulované chovanie odpovedať reálnemu chovaniu funkčného komunikačného rozhrania.



Obrázok 4.3: Časti jadra AC portálu

4.4.1 Implementácia jadra

Jadro bude implementované pomocou samostatného vlákna nad novým adresným priestorom ktorý bude zdieľať s vláknami komunikačných správnych a autentizačných modulov.

Jadro bude v základe rozdelené na dve časti. Prvá z nich bude implementovaná v súbore acportal.c, tá sa bude starať o správny formát ACP správ. Bude to tzv. obsluha ACP protokolu. Ďalej sa táto časť bude starať o pripojovanie modulov a o vzájomnú komunikáciu jadra AC portálu s komunikačnými modulmi. Druhá časť jadra nazvaná správa transakcií bude implementovaná v súbore transaction.c a tá bude mať na starosti udržovanie kontextu jednotlivých transakcií. Bude sa starať o ich správny priebeh a komunikáciu so správnymi a autentizačnými modulmi.

4.4.1.1 Inicializácia jadra

Bude spustená ako prvá z celého AC portálu. V nej sa zinicializujú všetky štruktúry jadra, vytvoria sa všetky vlákna pre obsluhu transakcií a tiež sa zinicializujú. Pripravia sa tiež štruktúry na pripojovanie modulov. Tie sa budú môcť následne po inicializácii pripájať.

4.4.1.2 Obsluha ACP protokolu

Táto časť jadra bude v slučke vykonávať nasledujúce operácie. Najskôr sa pokúsi vyčítať dáta z komunikačného modulu. Ak prišla nová správa tak ju zavolaním funkcie ParsAcpHeader skontroluje. Ak ide o korektnú ACP správu, funkcia naplní štruktúru typu TAcpHeader nesúcu spracované dáta z hlavičky ACP správy. Výhodou vyňatia informácii do samostatnej štruktúry uľahčia prácu s týmito informáciami. Zvyšok programu už bude nezávislý od formátu hlavičky ACP protokolu. Ak by sa jej formát zmenil, bude stačiť upraviť iba tento modul a zvyšok programu zostane nezmenený.

Ak je prijatá nová ACP správa so správnou hlavičkou bude podľa typu, buď predaná správe transakcií, alebo v prípade podpory tranzitovania bude správa postúpená inému AC portálu. Všeobecne sa prijaté správy typu Offer, Request a Finish vždy tranzitujú a nespracovávajú správou transakcií. Program takto prechádza jednotlivé pripojené komunikačné moduly a predáva správne prijaté ACP správy na ďalšie spracovanie.

Ďalšia časť tohto modulu sa stará o odosielanie dát na komunikačné moduly. Obsluha ACP protokolu sa pravidelne dotazuje modulu správy transakcií, či má pripravené nejaké dáta k odoslaniu. Ak áno, prijme tieto dáta, vygeneruje pomocou funkcie CreateAcpHeader hlavičku pre túto správu a predá ju spolu s dátami patričnému komunikačnému modulu k odoslaniu spolu s adresou.

Samostatnou funciou jadra spadajúcu pod správu ACP protokolu je registrácia komunikačných modulov. To prebieha prostredníctvom funkcie RegisterCommModule a tá umožňuje dynamicky pripájať komunikačné moduly. Túto funkciu musia komunikačné moduly zavolať, vtedy keď sú pripravené na prácu. Súčasťou registrácie je overenie autentickosti modulu. Implementácia bude umožňovať pripojenie len obmedzeného počtu komunikačných modulov, ich počet je definovaný hodnotou konštanty CommModuleCnt, ktorá definuje ich maximálny počet.

4.4.1.3 Správa transakcií

Tento modul je druhou zásadnou časťou jadra AC portálu. Jeho hlavnou úlohou je starať sa o uchovávanie kontextu transakcií a komunikáciu so správnymi a autentizačnými modulmi.

V prípade prijatia novej správy od obsluhy ACP protokolu, správa transakcií skontroluje na základe ID danej správy, existenciu aktívnej transakcie s týmto ID. Ak neexistuje zistí, či je k dispozícii neaktívne vlákno pre správu transakcie a obsadí ho touto transakciou. Ak daná transakcia existuje, predá jej prijaté dáta s kódom danej správy prostredníctvom IPC.

Ak je transakčné vlákno aktívne a čaká na odpoveď, v tom prípade čaká na IPC s odpoveďou v časovo obmedzenom zablokovaní wait. Ak čas uplynie a odpoveď nepríde, vlákno ukončí transakciu a vymaže všetky informácie z jej priebehu. Využitie časovo obmedzeného čakania na IPC systému L4 nám zastúpi funkciu samostatného časovača a ten tým pádom v implementácii nebude potrebný.

V prípade prijatia správy vláknom transakcie, predá jeho obsah správnemu modulu. Ak správny modul zistí, že prišla nesprávna odpoveď, tak vráti obsluhe vlákna informáciu o chybe. Obsluha pozastaví všetok príjem pre danú transakciu a po uplynutí ochrannej doby sa transakcia ukončí zmazaním všetkých jej dát. Ak správny modul vyhodnotí prijaté dáta ako správne, predá s kladnou návratovou hodnotou taktiež AVP správy, ktoré majú byť odoslané strane žiadateľa. Vlákno predá tieto informácie správe transakcií a tá sa postará o ich predanie obsluhe ACP protokolu. Špeciálnym prípadom je ak správny modul dokončil obsluhu transakcie a vráti návratovú hodnotu Finish. Vtedy vlákno po predaní dát správe transakcií stále beží a čaká na odoslanie dát. Po ich odoslaní sa táto transakcia ukončí.

4.4.2 Implementácia komunikačného modulu

Jednotlivé komunikačné moduly budú samostatné vlákna. Vlákna budú zdieľať adresný priestor spolu s jadrom AC portálu. Týmto spôsobom sa zjednoduší predávanie dát jadru AC portálu. Zamedzí sa tak zbytočnému a časovo náročnému kopírovaniu dát medzi dvoma adresnými priestormi. Nevýhodou tohto prístupu je nutnosť uchovávať dáta komunikačným modulom až do ich kompletného spracovania a tým pádom väčšie pamäťové nároky.

Vytvorený komunikačný modul bude pomocou blokujúcich IPC komunikovať s radičom komunikačného rozhrania. S jadrom bude komunikovať pomocou volaní funkcií komunikačného modulu. Komunikačný modul bude mať samostatné vlákna pre príjem a vysielanie. Jeho implementácia sa nachádza v súbore CommModule.c

4.4.2.1 Príjem dát

Pre príjem dát bude v komunikačnom module vyhradená vyrovnávacia pamäť o pevnej veľkosti 4kB. Ak by prijímaný balík mal mať väčšiu veľkosť, bude zahodený. Veľkosť je volená vzhľadom k tomu, aký maximálne veľký balík môže obsahovať ešte korektnú správu. Týmto sa systém chráni proti útoku na pretečenie pamäte. Prijímacie vlákno volaním komunikačného serveru pomocou IPC čaká na príjem dát. Ak dáta nie sú k dispozícii, vlákno je zablokované až do doby, keď dáta k dispozícii sú. Dáta vyčíta do vyrovnávacej pamäte a po prijatí celého balíka ho poskytne na spracovanie jadru AC portálu. Vtedy sa prijímacie vlákno zablokuje, až do spracovania dát. Jadro následným zavolaním funkcie ReadCommModuleData prijme dáta a odblokuje tým vlákno prijímacej časti komunikačného modulu.

4.4.2.2 Odosielanie dát

Pri odosielaní sa dáta budú kopírovať do vyrovnávacej pamäte. Následne pomocou IPC budú prenášané do serveru komunikačného rozhrania. Pokiaľ dáta boli vyslané, vlákno sa zablokuje a čaká na nové dáta z jadra AC portálu.
Súčasťou rozhrania komunikačných modulov je adresa. Tá v tejto implementácii nebude využitá. Predpokladá sa, že na druhej strane sa bude priamo komunikovať iba s jedným zariadením a to nebude tým pádom potrebné adresovať.

4.4.3 Implementácia správneho modulu

V AC portáli sa musí nachádzať aspoň jeden správny modul. Táto implementácia bude podporovať práve jeden autentizačný modul, ktorý bude riadiť priebeh transakcií. Súčasťou správneho modulu bude aj databáza užívateľov a systém záznamov prístupov. Implementácia správneho modulu sa nachádza v súbore PolicyModule.c.

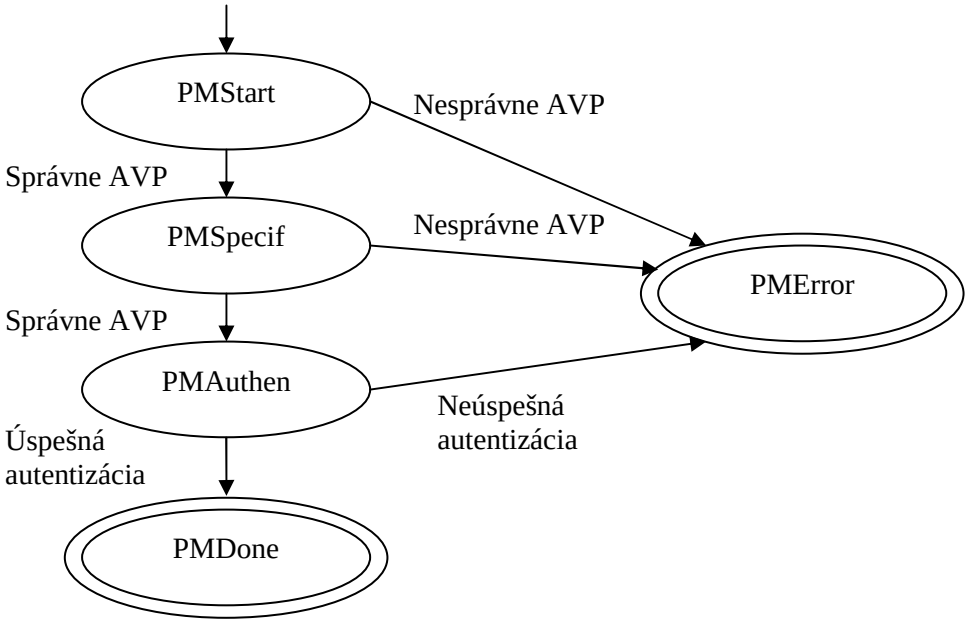
Správny modul bude volaný vláknom transakcie po prijatí nových dát. Bude zdieľať pamäťový priestor s vláknom, ktoré ho bude volať, čo zjednoduší predávanie dát, ale zvyšuje nároky na bezpečnosť týchto modulov. Jednotlivé správy tak budú obsahovať iba odkazy na jednotlivé dátové štruktúry v zdieľanej pamäti. Správny modul bude obsahovať stavový automat, ktorý na základe aktuálneho stavu a prijatých dát zmení svoj stav a zašle odpoveď druhej strane. Dáta, kód správy spolu s deskriptorom transakcice budú do správneho modulu predávané prostredníctvom volania jeho funkcie ProcessPolicy. Mimo tejto funkcie musí správny modul implementovať funkciu AuthorizePolicyModule. Prostredníctvom tohto si jadro portálu overuje autentickosť daného modulu, kedy mu na výzvu musí vrátiť očakávanú odpoveď.

Ak je deskriptor vynulovaný, správny modul je v počiatočnom stave. To znamená, že transakcia ešte nebola začatá a má byť prijatá správa s kódom Start. Implementácia bude

prijímať správu Start iba v prípade, že nebude obsahovať žiadne AVP. Správny modul bude pracovať v režime ACP-VSA kedy si bude jednotlivé AVP od žiadateľa požadovať prostredníctvom výzev. Výzva bude pozostávať z prázneho AVP, ktoré modul požaduje a z kódu správy. Po každej výzve bude očakávať to jedno konkrétne AVP so správnymi dátami. V opačnom prípade bude transakcia vždy ukončená.

V prípade chyby funkcia vráti hodnotu konštanty Error a prejde do stavu PMError. Jadro AC portálu sa následne postará o korektné ukončenie transakcie po chybe. V prípade, že transakcia prebieha správne vráti návratovú hodnotu AllOk. Ak je transakcia na svojom konci a žiadateľovi bude odoslaná odpoveď, správny modul vráti špeciálnu návratovú hodnotu Finish. Modul sa nastaví do stavu PMDone a neprijíma viac ďalšie správy. Po tejto návratovej hodnote jadro odošle odpoveď a po tom transakciu ukončí.

Samotné riadenie priebehu transakcie bude implementované pomocou stavového automatu pozri obr. 4.4. Ak sa po prijatí správy správny modul nedostane do koncového stavu automatu poznačí si nový stav do deskriptoru. Na základe aktuálneho stavu vygeneruje výzvu a predá ju na odoslanie jadru.

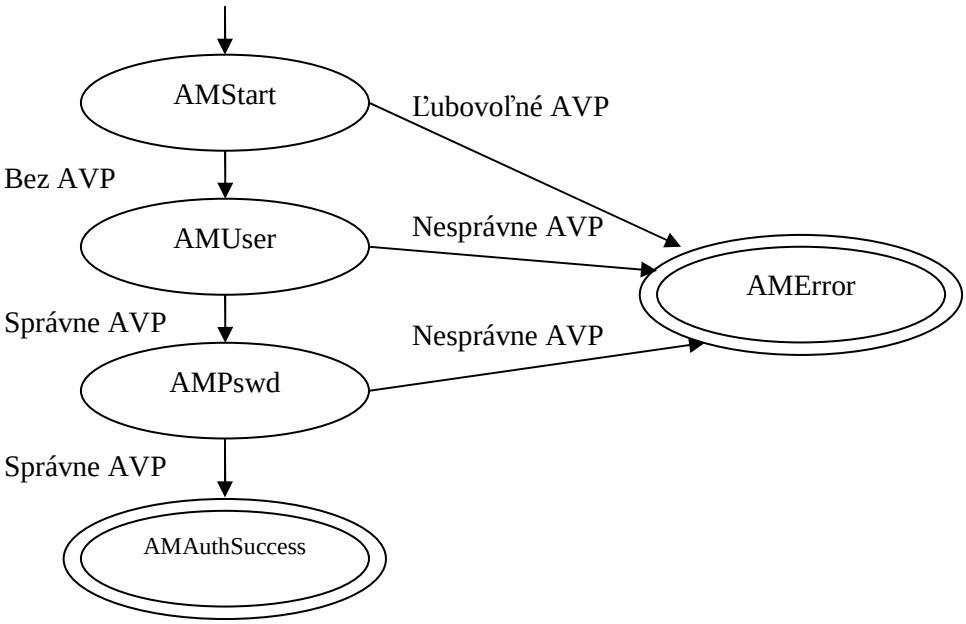


Obrázok 4.4: Stavový automat správneho modulu

4.4.4 Implementácia autentizačného modulu

Autentizačné moduly budú implementované podobne ako správne moduly. Budú obsahovať tentokrát rozhranie smerom k správnemu modulu. Toto rozhranie by šlo použiť aj v jadre, ale táto implementácia bude vždy komunikovať s autentizačným modulom prostredníctvom správneho. Ten bude pristupovať k službám autentizačného modulu prostredníctvom volania funkcie ProcessAuthentication. Autentizačný modul bude prijímať AVP prijaté od žiadateľa a bude preň generovať výzvy v podobe prázdnych AVP. Rozdielov v porovnaní so správnym

modulom bude, že autentizačný modul nebude rozlišovať kódy správ. Tie bude generovať a kontrolovať správny modul. Ďalším rozdielom bude začatie procesu autentizácie. Ten začne správny modul zavolaním autentizačného bez predania AVP. Ak sa autentizačný modul nachádza v základom stave a je zavolaný bez predania AVP, chápe to ako začiatok autentizačného procesu, zmení svoj stav na stav AMUser a vytroví AVP s dotazom na žiadateľove meno. Po jeho prijatí, modul vyhľadá užívateľa s týmto menom v databázi užívateľov a zašle mu výzvu na zaslanie prístupového hesla. Ak užívateľ neexistuje, autentizačný modul prejde do koncového chybového stavu AMError a ukončí proces autentizácie. To isté sa stane v prípade prijatého nesprávneho hesla. Tieto stavy následne rieši buď správny modul alebo jadro portálu. Autentizačný modul bude jedným z najdôležitejších prvkov systému. Bude zabezpečovať overovanie užívateľov pristupujúcich k službám, preto musí byť dobre navrhnutý a overený. Jediná chyba v ňom môže spôsobiť zlyhanie celého systému riadenia prístupu.



Obrázok 4.5: Stavový automat autentizačného modulu

4.5 Testovanie

Táto kapitola je venovaná spôsobu spustenia a samotnému testovaniu implementácie AC portálu. Samotné testovanie je založené na simulácii komunikácie žiadateľa generovanej špeciálnym modulom. Ten je k AC portálu pripojený na mieste radiča komunikačného rozhrania, ktorý nahradzuje. Testovanie bude pozostávať z nastavenia parametrov testovacieho modulu, spustením a následným sledovaním chovania AC portálu prostredníctvom jeho výstupu do terminálu.

4.5.1 Zavedenie a spustenie AC portálu

Po kompilácii AC portálu získame binárny súbor obsahujúci kód aplikácie spustiteľného nad jadrom Fiasco.OC. Tento súbor je možné pomocou Lua skriptu načítať prostredníctvom procesu Ned do pamäti a spustiť. L4Re umožňuje z jednotlivých binárnych súborov vytvoriť jediný tzv. obraz (image). Ten je následne jednoduchým spôsobom pomocou bootloaderu zavedený a spustený. V tomto prípade bude k spusteniu potrebné zaviesť následujúce súbory:

- fiasco,
- moe,
- l4re,
- ned,
- acportal.cfg,
- acportal.

Na vytvorenie tohto obrazu je nutné vytvoriť súbor obsahujúci zoznam modulov uložený v súbore modules.list, ktorý má nasledujúci obsah:

```
modaddr 0x01100000

entry acportal
      kernel fiasco -serial_esc
      roottask moe rom/acportal.cfg
      module l4re
      module ned
      module acportal.cfg
      module acportal
```

Súbor acportal.cfg obsahuje informácie o spôsobe spustenia obrazu. Jeho obsah je následujúci:

```
require("L4");
L4.default_loader:start({}, "rom/acportal");
```

Tieto dva súbory je vhodné umiestniť do koreňového adresára AC portálu. Po ich vytvorení je možné nasledujúcim príkazom vytvoriť obraz jednoduchý na zavedenie a spustenie. Príkaz musí byť zavolaný z adresára pre preklad platformy L4Re. Jeho umiestnenie je:

```
L4redir/src/l4
```

Samotný príkaz je potrebné upraviť podľa umiesnenia súborov modules.list a koreňového adresára AC portálu. Príkaz tak môže vyzerať nasledujúcim spôsobom.

```
makemake O=../../obj/l4/x86 grub2iso E=acportal
MODULES_LIST=~/DP/ACPpkg/modules.list MODULE_SEARCH_PATH=~/DP/ACPpkg
```

Vykonaním príkazu bude vytvorený súbor acportal.iso. Jeho umiestnenie je nasledujúce:

```
L4redir/obj/l4/x86/images
```

Spustením aplikácie vytvárajúcej virtuálne systémové prostredie je možné tento súbor spustiť. Jedným z takýchto programov pre Linux je Qemu. Spustenie AC portálu tak bude vykonané nasledujúcim príkazom.

```
qemu -cdrom acportal.iso -serial stdio
```

Príkaz spustí samostatné okno s konzolou, v ktorom sa spustí AC portál. Ten po spustení vypíše na obrazovku nasledujúci text:

```
Implementacia AC portalu vytvorena v ramci diplomovej prace.
  Autor prace: Bc. Tomas Kolarik
  Veduci prace: doc. Ing. Karel Burda, CSc.

Popis programu:
  Program je serverom mikrojadra L4, ktoremu tak umoznuje komunikovat
  pomocou ACP protokolu. Ide o funkcny prototyp demonstrujuci funkciu
  AC portalu. Program po spusteni bezi po celu dobu behu systemu.

Co program robi:
  Testovaci modul cyklicky generuje ACP spravy pre AC portal.
  Ten spravy prijima, zpracovava a posiela na ne odpoved.
  Na obrazovku su vypisovane hlasenia jednotlivych casti programu
  o spracovavani na jednotlivych urovniach.

  Hlasky jednotlivych casti su odlisene dvojpismenovou predponou:
    TE - testovaci modul,
    SA - obsluha ACP protokolu v jadre AC portalu,
    ST - sprava transakcii v jadre AC portalu,
    KM - komunikacny modul,
    SM - spravny modul,
    AM - autentizacny modul.

Vysup programu:
```

Za ním potom nasledujú výpisy zo samotného behu programu.

4.5.2 Možnosti testovania

Keďže vstupné rozhranie nie je štandardnou súčasťou koreňovej úlohy Moe, bol vytrovený testovací systém, ktorého parametre sa definujú v zdrojovom kóde a tak nevyžaduje vstup počas behu testu.

Základným nástrojom pre sledovanie priebehu testu je textový výstup z programu. Ten je zobrazovaný v termináli. Jednotlivé časti AC portálu majú svoje výpisy na začiatku označené dvoma písmenami.

- TE - testovací modul,
- SA - obsluha ACP protokolu v jadre AC portálu,
- ST - správa transakcií v jadre AC portálu,
- KM - komunikačný modul,
- SM - správny modul,
- AM - autentizačný modul.

Pre riadenie výpisov je v zdrojových kódoch zavedená direktíva DebugOut. Tá umožňuje zakazovať vypisovať hlásenia pre ľubovoľný modul či jadro. To umožňuje sprehľadnenie testovania.

Jednotlivé moduly výpisom reagujú na príjem a odosielanie dát, takýmto spôsobom sa dá sledovať cesta dát systémom. Ak sa jedná pre prechod dát modulom, modul výpis neukončuje prechodom na nový riadok. Tak je možné na jednotlivých riadkoch pozorovať cestu jednotlivých správ systémom.

```
acportal| SA ACP OK| ST Predam vlaknu| SM kod SPECIF.
```

V tomto prípade je povolený výstup iba jadru a správnemu modulu. Z tohto záznamu môžme vyčítať, že správa ACP protokolu overila správnosť formátu správy výpisom ACP OK, ďalej ho predala správe transakcií, ta našla vlákno pre danú transakciu a predala mu dáta. Nakoniec správny modul zaznamenal príchod očakávanej správy s kódom Specificaion.

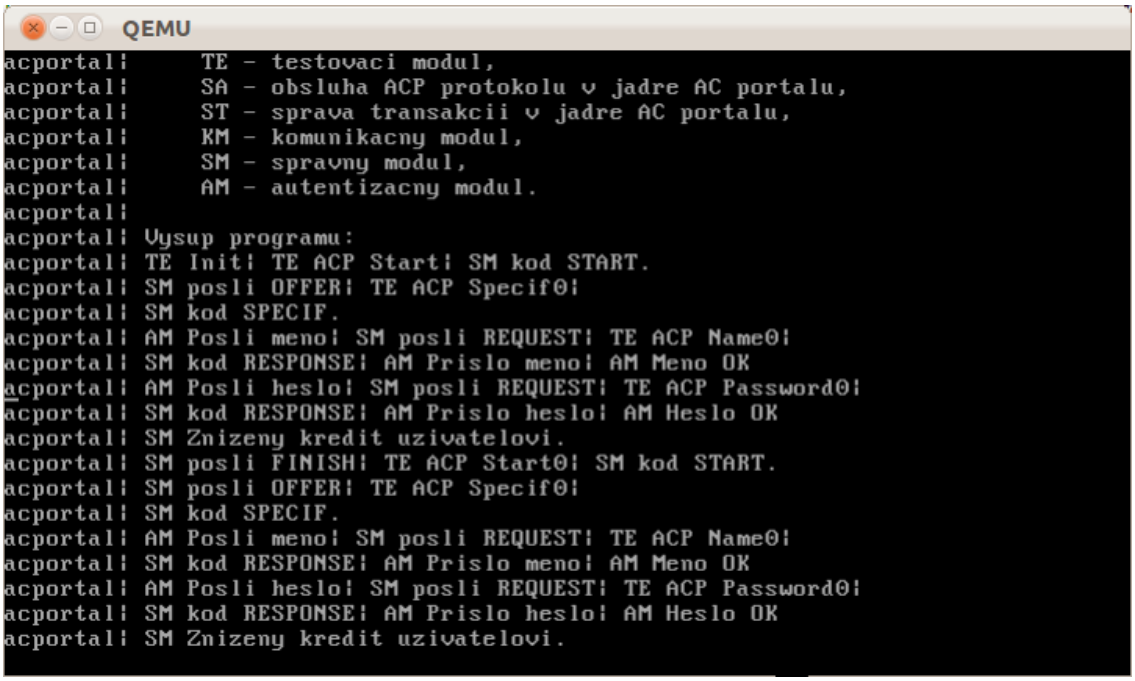
Priebeh testu je možné ovplyvňovať prostredníctvom testovacieho modulu. Ten obsahuje jednotlivé posielané správy, tie je možné ľubovoľným spôsobom editovať a testovať tak reakcie portálu na chyby na jednotlivých úrovniach. Je možné meniť aj ich poradie. Ďalšou funkciou testovacieho modulu je možnosť generovať paralelne niekoľko transakcií, čím je možné otestovať schopnosť systému spracovať ich. Jednotlivé paralelné transakcie sú vo výstupe označované číslom na konci textu. Poslednou možnosťou testovania je vkladanie oneskorenia medzi dáta, tým je možné overiť reakcie na oneskorený príchod správy ACP.

4.5.3 Výstupy testov

Jednotlivé testy sa zameriavajú na niektorú vlastnosť AC portálu. Na základe ich výstupu je potom sledované chovanie AC portálu. Je dôležité vždy povoliť výpis len tých častí, ktoré sú pre daný test potrebné. Povolením všetkých výpisov sa výstup stáva neprehľadným a informácie sa z neho len ťažko vyčítavajú. Samotná komunikácia medzi testovacím

modulom a komunikačným modulom je dostatočne pomalá na to aby sa dal beh programu sledovať bez nutnosti nástroja na zastavovanie.

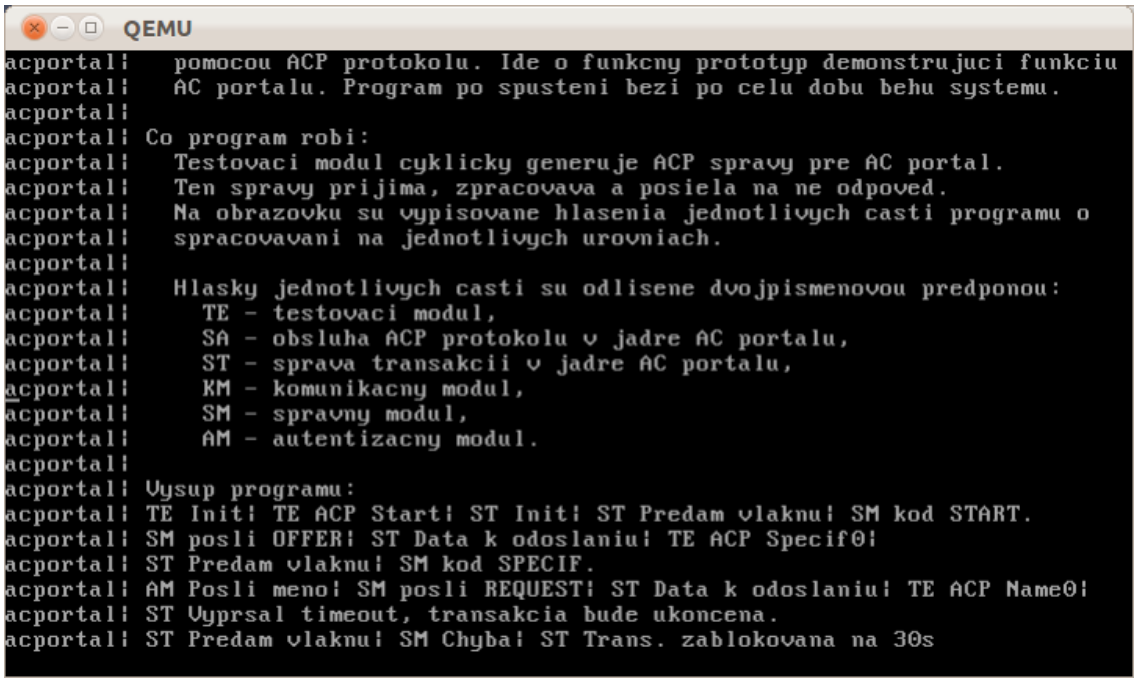
Na nasledujúcom obrázku je možné vidieť test správneho opakovaného behu jednej transakcie.



Obrázok 4.6: Výpis testu behu korektnej transakcie

Je tu možné pozorovať, že po úspešnom autorizovaní žiadateľa mu je znížený kredit. Čo ukazuje funkčnosť účtovacieho systému AC portálu.

Ďalšou ukážkou testu je prípad kedy správa nedorazí včas a AC portál zastaví beh transakcie a znemožní príchod ďalších jej správ.



Obrázok 4.7: Výpis testu behu transakcie s vypršaním časovača

Záver

Hlavným cieľom diplomovej práce bolo vytvoriť implementáciu ACP protokolu do operačného systém L4.

Práca sa v úvode venuje problematike riadenia prístupu pomocou AAA systémov. Tie k svojej komunikácii využívajú AAA protokoly, medzi ktoré patrí aj ACP. V ďalšej kapitole je podrobný popis a formát jednotlivých správ ACP protokolu a stručný prehľad používaných AVP štruktúr.

Ďalšia časť práce vytvára všeobecný prehľad o problematike operačných systémov. Na začiatku je popis ich architektúry a následne je pozornosť venovaná vlastnostiam vzťahujúcim sa k mikrojadrám. Tým bol vytvorený dobrý úvod k pochopeniu ďalšej časti, venovanej systému L4. Táto časť oboznamuje čitateľa s myšlienkou L4 systémov a o mikrojadrách druhej generácie. Približuje ich históriu a následne naznačuje vývoj mikrojadier tretej generácie. Potom je podrobnejšie popísané mikrojadro Fiasco.OC s rozširujúcim prostredím L4Re. Tento systém bol zvolený za platformu, pre ktorú bol AC portál implementovaný. V práci je podrobne popísaný spôsob prekladu a spustenia vytvoreného kódu. Súčasťou je popis prípravy implementačného prostredia pozostávajúci z nastavenia a prekladu L4Re a Fiasco.OC. Tu je tiež zadokumentovaná odchýlka v chovaní prekladu L4Re na novších systémoch Linux, ktorá spôsobuje zlyhanie prekladu. S pomedzi dostupných systémov L4Re ponúka široké možnosti pri tvorbe aplikácií a serverov. Táto práca tak ponúka návod na prácu s týmto systémom a uľahčí tak vývoj ďalších súčastí.

Súčasťou práce je všeobecný úplný návrh implementácie ACP protokolu nezávislý na platforme. Ten je vytvorený na základe analýzy popisu protokolu ACP. Návrh rozdeľuje systému na jednotlivé časti. Popisuje ich funkcionalitu a vzájomné rozhrania, ktorými sú prepojené. Súčasťou je tiež návrh dátových štruktúr a popis inicializácie a spúšťania AC portálu. Tento návrh zohľadňuje maximalistickú variantu AC portálu, ktorý môže byť základom pre implementáciu AC portálov pre rôzne platformy a operačné systémy.

Tento všeobecný návrh bol základom pre vytvorenie implementácie do operačného systému L4. V úvode bola urobená analýza možností implementácie v rámci systému, tu bola zvolená varianta implementácie ako serveru operačného systému bežiaceho v užívateľskom režime procesoru. Vytvorený AC portál implementuje základnú sadu vlastností potrebných pre podporu ACP protokolu. Implementácia nepodporuje tranzitovanie transakcií, keďže táto funkcia nie je nevyhnutná. Vytvorený AC portál tak bude možné nasadiť iba na koncových zariadeniach, ako sú napríklad prístupové čipové karty. Mimo to umožňuje paralelné spracovanie niekoľkých transakcií, čo umožnilo rozdelenie ich spracovania na samostatné vlákna. Tento prístup a neprítomnosť tranzitovania umožnili zjednodušiť celý návrh. Vytvorený AC portál tiež obsahuje jednoduché zásuvné moduly pre obsluhu komunikácie, správu transakcií a autentizáciu.

Táto práca vytvorila prvý funkčný prototyp AC portálu pre tento operačný systém. Spolu s ním bol vytvorený samostatný testovací modul. Ten umožňuje testovať chovanie AC portálu v rôznych situáciách aj bez prítomnosti komunikačného rozhrania. Vytvorením ďalších serverov pre operačný systém L4 ako napríklad TCP/IP serveru, by bolo možné tento

prototyp naplno otestovať aj v reálnych podmienkach počítačových sietí. Ďalším vývojom zásuvných modulov a rozširovaním podpory autentizačných metód tak môže vzniknúť portál naladiteľný do reálnych aplikácií.

Literatúra

[1] BOVET, P.D., CESATI M. *Understanding The Linux Kernel*. 3rd ed. Sebastopol: O'Reilly Media, 2005. 944 s. ISBN 0596005652.

[2] BURDA, Karel. *AAA systémy a protokoly*. Elektrorevue. 2009, 46, s. 1-7. Dostupný také z WWW: <http://www.elektrorevue.cz/cz/clanky/informacni-techologie/0/aaa-systemy-a-protokoly/>.

[3] BURDA, Karel. *Univerzální rámec pro řízení přístupu v počítačových sítích*. Elektrorevue [online]. 2011, 9, [cit. 2011-12-10]. Dostupný z WWW: <http://elektrorevue.cz/cz/download/univerzalni-ramec-pro-rizeni-pristupu-v-pocitacovych-sitich/ >.

[4] BURDA, K., STRASIL, I., PELKA T., STANCIL, P. *Access Control Protocol (ACP)*. Brno: Vysoké učení technické v Brně, 2011. 25 s. Dostupné z WWW: <http://tools.ietf.org/html/draft-kaaps-acp-01>.

[5] GARRIDO, J., SCHLESINGER R., HOGANSON K.E. *Principles of Modern Operating Systems*. 2nd ed. Sudbury: Jones and Bartlett, 2011. 564 s. ISBN 1449626343.

[6] KUZ, Ihor. *L4 User Manual : API Version X.2* [online]. Sydney: UNSW, School of Computer Science and Engineering, 2004 [cit. 2011-12-10]. Dostupné z WWW: <http://www.l4hq.org/docs/manuals/l4uman-x2.pdf>.

[7] L4HQ - THE L4 HEADQUARTERS. *Kernel Implementations* [online]. 2012 [cit. 2012-5-11]. Dostupné z WWW: <http://www.l4hq.org/projects/kernel>.

[8] L4KA TEAM. *eXperimental Kernel. Reference Manual* [online]. Revision 7. Karlsruhe : Universität Karlsruhe, 2011 – [cit. 2011-12-14]. Dostupné z WWW: <http://www.l4ka.org/l4ka/l4-x2-r7.pdf>.

[9] Labrosse, J., et. al. *Embedded software*. 1st ed. Amsterdam: Newnes, 2008. xxi, 770 s. ISBN 978-0-7506-8583-2.

[10] LIEDTKE, Jochen. *Improving IPC by kernel design*. Asheville, NC, USA: 14th ACM Symposium on Operating System Principles, 1993. s. 175 –188. Dostupné z WWW: <http://os.ibds.kit.edu/downloads/publ_1993_liedtke_improving-ipc.pdf>.

[11] NICTA. *About Microkernels* [online]. 2011 – [cit. 2011-12-12]. Dostupné z WWW: <http://ertos.nicta.com.au/research/l4/microkernels.pml>.

[12] NICTA. *Secure Microkernel Project (seL4)* [online]. 2011 – [cit. 2011-12-13]. Dostupné z WWW: <http://ertos.nicta.com.au/research/sel4/>.

[13] OPEN KERNEL LABS. *The Next Billion Devices* [online]. 2012 – [cit. 2012-3-30]. Dostupné z WWW: <http://www.ok-labs.com/about/about-ok-labs>.

[14] OPERATING SYSTEMS GROUP. *L4Re - L4 Runtime Environment, Fiasco.OC & L4 Runtime Environment (L4Re)*[online]. Dresden: TUD, 2011 – [cit. 2012-5-3]. Dostupné z WWW: <http://os.inf.tu-dresden.de/L4Re/doc>.

[15] PENUMUCHU, Chowdary Venkateswara. *Simple Real-time Operating System: A Kernel Inside View for a Beginner*. Victoria: Trafford Publishing, 2007. 322 s. ISBN 1425117821.

[16] SILBERSCHATZ, Abraham. *Operating system concepts with Java.* 6th ed. New York: John Wiley, 2004. xxiii, 952 s. ISBN 0-471-45249-1.

[17] SIVARAMA, P. Dandamudi. *Guide To Risc Processors: For Programmers And Engineers*. New York: Springer, 2005. 387 s. ISBN 0387210172.

[18] SHARMA, Vivek. *Design and implementation of operating system.* 1th ed. New Delhi: Laxmi Publications Pvt Limited, 2010. 483 s. ISBN 9380386416. Dostupné z WWW: <http://books.google.cz/books?id=3iE3UWrE4L0C>.

Zoznam príloh

Príloha 1.

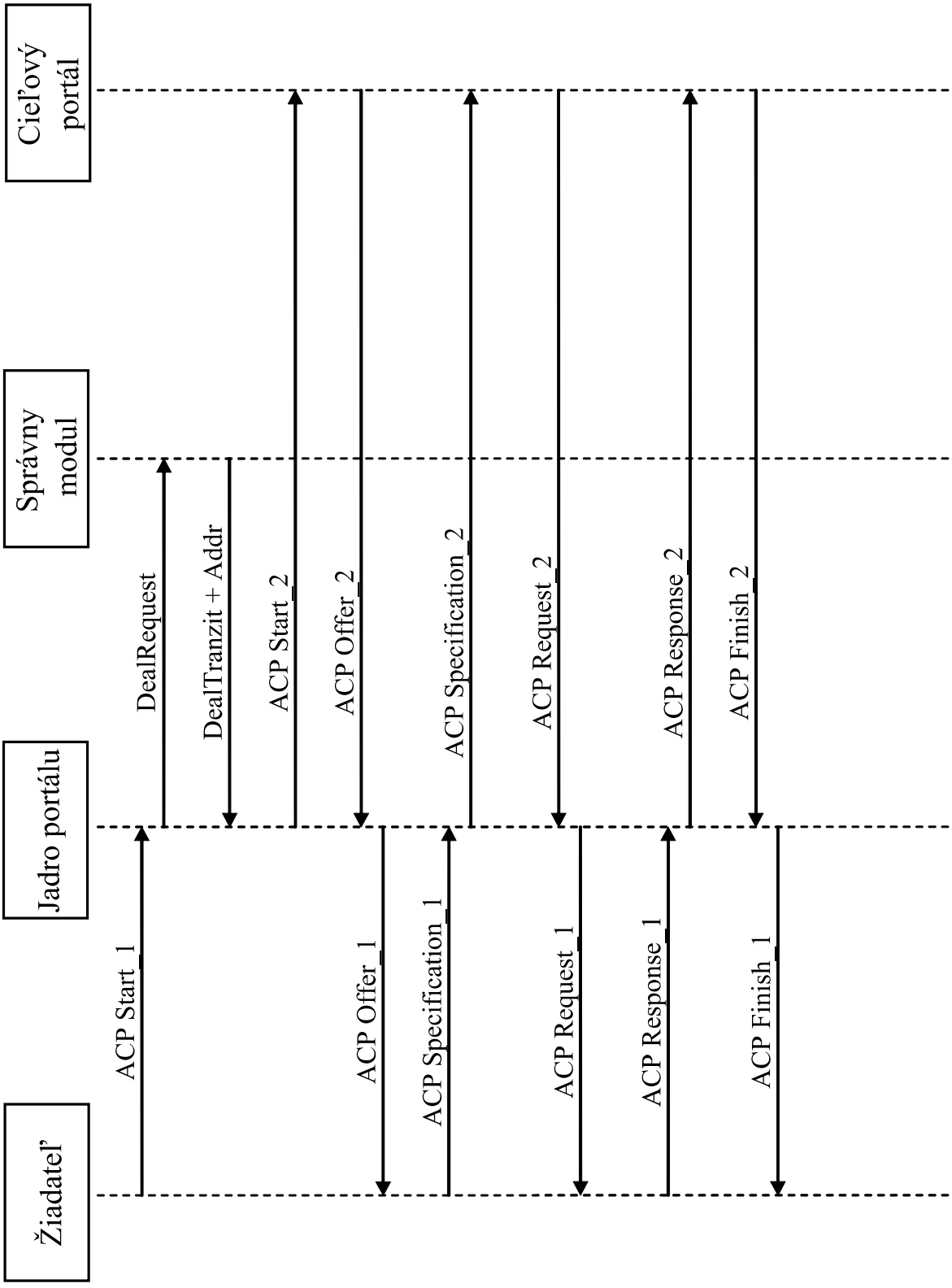
Zoznam použitých skratiek

AAA	Authentication Authorization Accounting
AC	Access Control
ACP	Access Control Protocol
ACP-VSA	Access Control Protocol - Variant of Single Answers
ACPI	Advanced Configuration and Power Interface
API	Application Program Interface
ARM	Advanced RISC Machine
AVP	Attribute Value Pair
BR	Buffer Register
BSD	Berkeley Software Distribution
C	Názov programovacieho jazyka
C++	Názov programovacieho jazyka
CAVP	Container Attribute Value Pair
DDE	Device Driver Environment
DROPS	Dresden Real-Time Operating System Project
EAP	Extensible Authentication Protocol
GNU	GNU Not Unix
GPLv3	General Public License v 3
GUI	Graphical User Interface
ID	Identificator
IPC	Inter-Process Communication
IP	Internet Protocol
IPv4	Internet Protocol v 4
IPv6	Internet Protocol v 6
I/O	Input Output
L4	Mikrojadro zamerané na výkon
L4Re	L4 Runtime Enviroment
LAM	Local Authentication Method
LAVP	Long Attribute Value Pair
MAC	Medium Access Control
MMU	Memory Management Unit
MR	Message Register
OKL	Open Kernel Labs
OS	Operating System
PCI	Peripheral Component Interconnect
RADIUS	Remote Authentication Dial In User Service
RAM	Random Access Memory
RFC	Request for Comments
ROM	Read Only Memory
SAVP	Short Attribute Value Pair

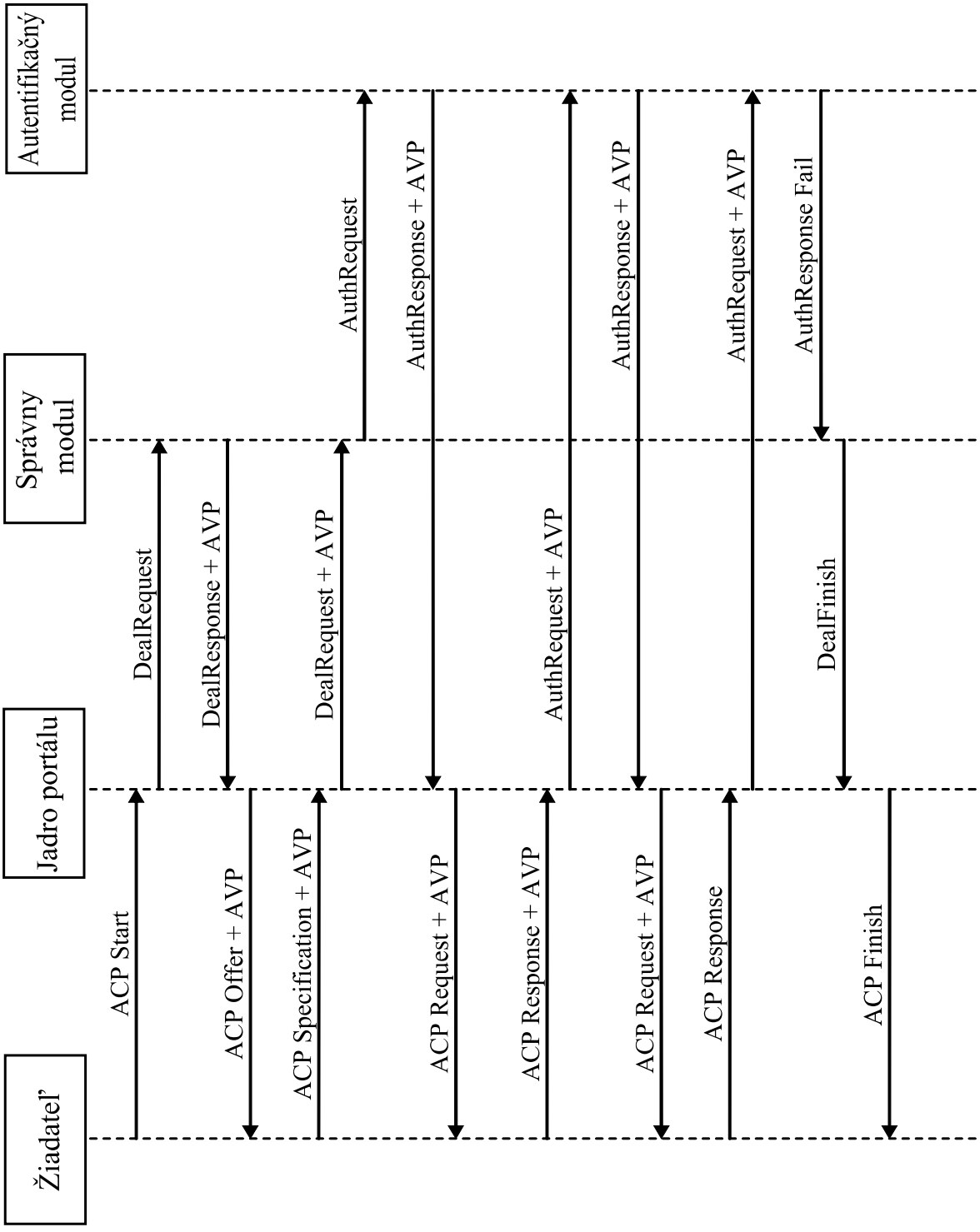
TACACS+	Terminal Access Controller Access-Control System
TCP	Transmission Control Protocol
TCR	Thread Control Register
TLS	Transport Layer Security
UDP	User Datagram Protocol
UID	Unique identifier
USB	Universal Serial Bus

Príloha 2.

Sekvenčné diagramy



Obrázok prílohy 2.1: Diagram komunikácie s vložením transakcie



Obrázok prílohy 2.2: Diagram komunikácie so zlyhaním autentizácie