VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SIMULAČNÍ NÁSTROJ PRO TESTOVÁNÍ VÝKONU SAFEQ CORE SERVER

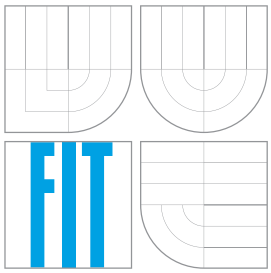
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETER CHOCHOLÁČEK

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SIMULAČNÍ NÁSTROJ PRO TESTOVÁNÍ VÝKONU SAFEQ CORE SERVER

SIMULATION TOOL FOR LOAD TESTS OF SAFEQ CORE SERVER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETER CHOCHOLÁČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. OTA JIRÁK

BRNO 2009

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2008/2009

Zadání bakalářské práce

Řešitel: **Chocholáček Peter**
Obor: Informační technologie
Téma: **Simulační nástroj pro testování výkonu SafeQ Core Server**
Kategorie: Modelování a simulace

Pokyny:

1. Analyzujte komunikaci mezi SafeQ Core Serverem (tiskové řešení) a MFD.
2. Navrhněte způsob zápisu scénářů testů.
3. Určete základní scénáře testů.
4. Implementujte simulátor MFD, který bude schopen přijímat tiskové úlohy ze strany SafeQ a reagovat na ně podle zadaných kritérií. Tento nástroj bude umožňovat logování veškerých událostí a konfigurovatelné reakce na tiskové úlohy.
5. Záležitosti související s položkami v bodech 1 až 4 konzultujte s p. Petrem Neugebauerem z Y Soft, s.r.o., Podnikatelská 4, Brno.
6. Zhodnoťe dosažené výsledky.

Literatura:

- A. Spillner, T. Linz, H. Schaefer: Software Testing Foundations: A Study Guide for the Certified Tester Exam, 2nd Edition, ISBN-13:978-1933952086

Při obhajobě semestrální části projektu je požadováno:

- Bez požadavků.

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

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

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

Vedoucí: **Jirák Ota, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2008
Datum odevzdání: 20. května 2009

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

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zabývá implementací jednoduché síťové aplikace, která navenek vypadá a chová se jako síťová tiskárna. Zkoumá možnosti jakými s ní lze komunikovat a způsoby jak na ní posílat tiskové úlohy. Na přijaté tiskové úlohy reaguje změnou svého interního stavu, který lze promítnout na straně klienta, který je k ní připojen.

Abstract

This bachelor's thesis pursues implementation of a simple network application, which from the outside looks and behaves as a network printer. Investigates ways to communicate and send print jobs to it. Change in internal state is caused by such a communication, which is possible to check on the client side.

Klíčová slova

tiskárna, testování, tisk v síti, tiskový jazyk, SNMP, LPD

Keywords

printer, testing, network printing, print language, SNMP, LPD

Citace

Peter Chocholáček: Simulační nástroj pro testování výkonu SafeQ Core Server, bakalářská práce, Brno, FIT VUT v Brně, 2009

Simulační nástroj pro testování výkonu SafeQ Core Server

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Oty Jiráka.

.....
Peter Chocholáček
19. května 2009

Poděkování

Tímto chci poďekovat panu Jirákovi za jeho rady při tvorbě této práce, jako i všem, kteří mě při psaní podpořili.

© Peter Chocholáček, 2009.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

Počítačová tlač je jedným z hlavných spôsobov využitia dnešných osobných počítačov. Tlačiarne sú neodmysliteľným prostriedkom na vytvorenie hmatateľnej kópie elektronických dát určených pre použitie vo firme, v škole alebo doma. Stali sa nepostrádateľným periférnym zariadením vo všetkých oblastiach pracujúcich s výpočtovou technikou. Dnes ich už môžme nájsť v mnohých kanceláriách a domácnostiach, ale trvalo takmer polstoročie, než si ľudia uvedomili ich dôležitosť a kým ich vývojári a dizajnéri spravili dostupnými pre každého.

Stálo to mnoho ľudského úsilia, než sa tlačiarne stali takými, ako ich poznáme dnes, sofistikovanými zariadeniami umožňujúcimi zjednodušiť nám základné kancelárske úkony ako je kopírovanie, tlač, skenovanie a ďalšie.

Myšlienka reprodukcie textu a obrazu pochádza ešte zo stredoveku, z čias, kedy boli všetky texty a knihy prepisované ručne. Vtedy bol najviac zaťažovaným článkom človek a na prepísanie rozsiahlej knihy mu hocikedy nestačil ani celý život. To sa z časti zmenilo s príchodom Guttenbergovej vylepšenej kníhtlače, ktorá umožňovala tlačiť obrovské množstvá celých kníh. Vynález kníhtlače znamenal pre ľudstvo obrovský skok vpred, kvalitatívne upravil obsah, formu a dizajn tlačených kníh. Avšak, čo sa týka výšky spotrebných nákladov, kníhtlač vyžadovala veľké množstvá vstupov a nebola dostupná pre radových občanov. Všeobecne, potreba znížiť náklady na tlač trvá dodnes a je dôležitou súčasťou plánovania úspor či zvýšenia efektivity využívania zdrojov.

Predstavme si stručný popis štruktúry našej práce.

V prvej kapitole popisujeme usporiadanie práce a jej jednotlivých častí.

Druhá kapitola pojednáva o možnostiach počítačovej tlače v konkrétnom prostredí a forme, akou sú dáta na tlač generované a odosielané.

Následne, v tretej kapitole sa zaoberáme možnosťami ako počítačovú tlač monitorovať a spravovať.

Štvrtá kapitola nám prináša náhľad významu softvérového testovania a určuje pozíciu implementovaného systému v konkrétnom testovacom prostredí.

Napokon v piatej kapitole detailnejšie popisujeme funkcionalitu a spôsob implementácie výsledného systému.

V závere úvádzame možnosti rozšírenia výslednej aplikácie.

Kapitola 2

Počítačová tlač

Počítačové tlačiarne prešli od svojich počiatkov dlhým vývojom. V tejto kapitole si priblížime základne pojmi súvisiace s počítačovou tlačou.

2.1 Lokálna tlačiareň

Ako už z názvu lokálnej tlačiarne vyplýva, dá sa používať iba na jednej lokálnej stanici. Môže na nej tlačiť len používateľ, ktorý ju má pripojenú k pracovnej stanici.

V závislosti na konkrétnych potrebách organizácie nie je vždy potrebné, aby bola ku každému počítaču pripojená tlačiareň. Takýto spôsob prevádzky tlače môže byť veľmi nákladný. Kým použitie lokálnych tlačiarní pripojených priamo k užívateľskému počítaču je pre užívateľa najjednoduchším riešením, tým častejšie je spojené s veľkou spotrebou energií.

2.2 Sieťová tlačiareň

Výrobcovia tlačiarní sa vysporiadali s potrebou zníženia správy a nákladov tlače vytvorením tlačového zariadenia, ktoré môže súčasne slúžiť nielen jednej kancelárii, ale aj celému oddeleniu vo firme. Sieťové tlačiarne sú často odolné, rýchle a majú dlhú životnosť. Obvykle sú pripojené k tlačovému serveru, ktorým môže byť aj obyčajný osobný počítač, ktorý riadi a smeruje tlačové úlohy po sieti k vybranej tlačiarni. Moderné multifunkčné zariadenia majú v sebe integrované sieťové rozhrania, ktoré umožňujú ich priame zapojenie do siete ako ktorúkoľvek inú pracovnú stanicu, bez nutnosti napojenia na tlačový server.

2.3 Tlačové jazyky

Kedysi dávno musel vývojár programu, ktorý mal podporovať tlač, vyvinúť svoje vlastné tlačové ovladače. To bolo dosť zložité, pretože rôzne programy mali odlišné formáty súborov. Dokonca ani programy z rovnakej oblasti, napríklad textové procesory, nepodporovali svoje formáty navzájom. Takže neexistovalo žiadne spoločné rozhranie pre tlačiarne a preto programy podporovali len niekoľko málo vybraných modelov. Ako sa objavovali nové modely tlačiarní, autori programov museli písať nové ovládače. Ani výrobcovia nemohli zaistiť, aby ich tlačiarne boli podporované každým programom, ktorý existoval (aj keď ich bolo menej ako dnes).

Starať sa o desať počítačov a dvanásť tlačiarní znamenalo pre správcu systému, že sa musí starať o stodvadsať ovládačov. Preto sa zvyšoval tlak na vývoj jednotného rozhrania medzi programami a tlačiarňami.

Vznik jazykov pre popis stránky - PDL (z angl. Page Description Language), ktoré štandardným spôsobom popisovali grafickú reprezentáciu čiar na papieri (alebo na inom zobrazovacom zariadení) bol udalosťou, ktorá pomohla riešiť tento problém.

Jazyky pre popis stránky pracujú na rovnakom princípe ako počítačové programovacie jazyky. Ak je dokument pripravený k tlači, pracovná stanica spracuje všetky typografické informácie a použije ich ako objekt reprezentujúci inštrukciu, ktorú má tlačiareň vykonať. Tlačiareň potom zparsuje objekty do rastrov a čiar, ktoré tvoria obrázok dokumentu (nazývaný Raster Image Processing - RIP) a vytlačí výstup na stánku ako jeden obrázok, kompletne so všetkými textami a kresbami. Tento tok prací (z angl. work-flow) robí z tlače dokumentov akejkoľvek zložitosti jednotný proces, zpravidla obdobne fungujúci na každej tlačiarni. PDL sú navrhované tak, aby boli prenosné na akýkoľvek formát tlače a škálovateľné na rôzne veľkosti stránky.

Jedným z týchto jazykov bol PostScript od firmy Adobe. Vďaka nemu sa mohol programátor aplikácie zamerať na svoj program a ako výstup vygenerovať popis stránky v jazyku PostScript. Následne sa mohli vývojári tlačiarní sústrediť na budovanie podpory pre zariadenia PostScript.

Samozrejme, objavili sa i ďalšie metódy popisu stránky. Najdôležitejšími protivníkmi PostScriptu sú PCL (z angl. Print Control Language) od firmy Hewlett-Packard, ESC/P (z angl. Epson Standard Code for Printers) od firmy Epson a GDI (z angl. Graphical Device Interface) od spoločnosti Microsoft.

Fakt, že sa tieto jazyky objavili, zjednodušil život a podporil ďalší rozvoj. Avšak skutočnosť, že stále ešte existujú rôzne nekompatibilné a konkurujúce si jazyky, vytrvalo komplikuje život užívateľom, administrátorom, vývojárom a výrobcom.

Charakterizujme si teraz, dátové typy spojované so sieťovou tlačou:

- RAW - dátový typ používaný v jazykoch pre popis stránky, ktorý je možné priamo odoslať na zariadenie, napríklad PCL alebo PostScript. Jedná sa o implicitný typ dát na počítačoch s Windows XP Professional alebo Windows 2000. Tieto dáta sú závislé na type zariadenia, pre ktoré boli určené, to znamená, že ich interpretácia môže byť na rôznych tlačový zariadeniach chybná a výsledný vytlačený dokument môže na každej vyzerať inak.

- EMF - proprietárny formát dát of firmy Microsoft. Enhaced metafile štruktúra špecifikuje metasúborový formát, ktorý dokáže uložiť obrázok vo formáte nezávislom na konkrétnom zariadení. Tento obrázok môže byť zparsovaný prečítaním metasúboru. EMF metasúbor je sériou záznamov nazývaných EMF záznamy, ktoré obsahujú grafické príkazy, definície objektov a vlastnosti. [30]

- TEXT - dátový typ určený pre odoslanie jednoduchej textovej tlačovej úlohy na tlačiareň. Pokiaľ napr. PostScriptová tlačiareň nedokáže interpretovať jednoduchý text, tlačový spooler (proces riadiaci prenos dát na tlačiareň) sa stará o to, aby sa text úlohy pridal k inštrukciám odvodeným z implicitného nastavenia vzhľadu dokumentu, ako je font, forma a orientácia. Kódovanie textových úloh je v ASCII, čo v praxi znamená, že lokalizované znaky sa na koncovom zariadení nemusia zobraziť správne.

2.3.1 Tlačové ovládače

Tlačový ovládač je softvér, ktorý vie, ako komunikovať s tlačiarňami a plotrami (zariadeniami používanými hlavne na kreslenie grafov, diagramov a inej vektorovej grafiky). Prekladá informácie vytvorené aplikáciou cez GDI do príkazov, ktorým tlačové zariadenie rozumie. Tieto príkazy slúžia na vytvorenie výslednej podoby textu a obrazu. Aby bolo možné podporovať rôzne druhy a typy zariadení, na tlačovom servery musia byť nainštalované rôzne tlačové ovládače, ktoré sú podporované pre daný hardvér a operačný systém.

Napríklad, ak administrátor spravuje Windows Server 2003 a zdieľa sieťovú tlačiareň s klientami, ktorí majú počítače s inou verziou Windows, je možné, že si bude musieť pridať ovládače týchto starších verzií tak, aby klienti neboli nútení si tieto ovládače inštalovať.

2.4 Tlačový server

Tlačový server (niekedy nazývaný tlačový spooler) je počítačový program alebo samostatné zariadenie, ktoré je spojené s jednou alebo viacerými tlačiarňami a s klientskými stanicami, od ktorých prijíma tlačové úlohy a preposiela ich na jednotlivé koncové zariadenia. Tento server má funkciu spoolera a prijaté úlohy radí do front, z ktorých sú zasielané na konkrétne tlačiarne. Spooling sa vzťahuje k prenosu dát ukladaných na dočasné miesto, odkiaľ môžu byť neskoršie vyzdvihnuté. Tieto dáta sú uložené do pamäťového zásobníka (angl. buffer), buď v operačnej pamäti alebo na pevnom disku, odkiaľ ich tlačiarne môžu čítať, keď sú pripravené k tlači. Zásobník v podobe súborov je prechodnou stanicou pre úlohy, než sa dostanú k spracovaniu tlačovým zariadením. To je v zásade oveľa pomalšie ako schopnosť servera prijímať, zaraďovať a preposielať úlohy. Takýto spooler umožňuje neskoršie spracovanie úloh, bez nutnosti čakať na výsledok spracovania každej jednej úlohy.

2.4.1 Line Printer Deamon

Line Printer Daemon/Line Printer Remote je tlačový protokol, používajúci sa na spojenie medzi počítačmi a tlačiarňami TCP/IP. Bol vyvinutý pre BSD UNIX a odvtedy je, de facto, štandardným multisystémovým protokolom.

LPD software je typicky spustený na tlačiarni alebo na tlačovom servery a LPR na klientskej stanici. LPR klient posiela požiadavok na tlač na IP adresu LPD servera, ktorý zaradí tlačovú úlohu do fronty a vytlačí ju, eventuálne pozdrží, kým je zariadenie pripravené k tlači.

Operačné systémy podobné systému Unix umožňujú tlač za asistencie týchto programov: lpr (pridanie do fronty), lpq (zobrazenie fronty), lprm (odstránenie úlohy z fronty), a lpc (konrola fronty). Ich možnosti a parametre závisia na konkrétnom použitom tlačovom prostredí. Program lpc typicky slúži na kontrolu a správu LPD démona, tlačových front ktoré démon riadi, nastavovanie poradia tlačových úloh vo frontách a zisťovanie informácií o tlačiarňach a ich frontách. V našom prípade je najdôležitejší príkaz lpr, ktorý zabezpečuje prenos dát na LPD server.

LPD démon počúva na svojom porte, typicky 515, a odpovedá na prichádzajúce príkazy. Port zo strany klienta musí ležať v rozmedzí od 721 až 731 vrátane. Všetky príkazy začínajú jedným znakom, jedným oktetom, ktorý vo forme čísla predstavuje požadovanú funkciu. Tento kód je nasledovaný ASCII menom tlačovej fronty, na ktorej má byť požiadavok vykonaný. Pokiaľ sa v príkaze nachádzajú ďalšie operandy, sú oddelené od mena tlačovej fronty

bielym znakom (ASCII mezerou, horizontálnym alebo vertikálnym tabulátorom, znakom nová stránka). Koniec príkazu je označený ASCII znakom nového riadku.

Konkrétne príkazy a detaily komunikácie môžme nájsť v [12].

2.4.2 Rozhranie typu AppSocket

Niektoré zariadenia podporujú príjem dát cez jednoduché TCP/IP spojenie, niekedy nazývané ako AppSocket protokol. Do tejto kategórie zapadajú hlavne zariadenia s rozhraním JetDirect od Hewlett Packard. Pre odoslanie úlohy na takéto rozhranie sa jednoducho otvorí TCP spojenie na konkrétny port, typicky 9100, a dáta sa odošlú v nezmenenej podobe, tak ako ich klient vygeneroval.

2.4.3 Ostatné protokoly

- Internet printing protocol - IPP, je aplikačný protokol používaný na lokálnu ako aj na internetovú tlač. Definuje komunikáciu medzi klientom a serverom. Klientovi umožňuje získať informácie o možnostiach tlačiarne a posielať na ňu dáta. Dovoľuje tiež kontrolu prístupu, autentifikáciu či šifrovanie. Na odosielanie tlačovej úlohy používa Hypertext Transfer Protocol (HTTP). Viac informácií o IPP môžme nájsť v [17].
- AppleTalk množina sieťových protokolov, na ktorých je postavená AppleTalk architektúra. Je určený na komunikáciu medzi sieťovými zariadenia firmy Apple.

2.5 Sieťová tlač

Sieťová tlač umožňuje užívateľom, ktorý sú geograficky na inom mieste ako je tlačové zariadenie (v inej kancelárii, na inom poschodí), tlačiť dokumenty. Možnosti, ako sieťovú tlač realizovať, sú závislé na danom prostredí a na platforme, pre ktorú je tlačová infraštruktúra budovaná.

2.5.1 Microsoft Windows

Komponenty využívané v sieťovej tlači na platforme Microsoft Windows sú server spooler, ktorý beží na počítači s operačným systémom podobnému Windows Server 2003, a klient spooler bežiaci na klientskom počítači. Podrobné informácie o týchto komponentoch je možné nájsť v dokumentácii Microsoft [28].

Pripojenie k sieťovému tlačovému zariadeniu je vyriešené pomocou portov [29]. Windows Server 2003 poskytuje zabudovanú podporu pre nasledujúce typy sieťových portov:

- Štandardný TCP/IP port monitor,
- LPR tlačový monitor,
- AppleTalk tlačový monitor,
- HTTP tlačový poskytovateľ.

Štandardný port monitor je nástupca LPR, ktorý je obecne prijatý ako, de facto, štandard v súvislosti so sieťovou tlačou. Na prenos dát využíva TCP/IP protokol. Taktiež používa SNMP na konfiguráciu a monitorovanie stavu tlačiarní. Spôsoby, akými posiela dáta na koncové zariadenie, sú buď RAW alebo cez LPR tlačový protokol.

Spôsob prenosu typu RAW je implicitným pre väčšinu tlačových zariadení. Na to, aby poslal úlohu vo formáte RAW, si tlačový server otvorí TCP stream na dané zariadenie. Najčastejšie je to port 9100. Po tom, ako je TCP/IP port vytvorený, Windows použije SNMP, aby sa opýtal zariadenia na niektoré z jeho identifikátorov objektov podľa [15].

Port monitor môže byť nakonfigurovaný tak, aby vyhovoval LPR štandardu, definovaného v [12]. Implicitne, štandardný port monitor je v rozpore s niektorými RFC 1179 [12] odporúčaniami:

- Neriadi sa odporúčaním, aby zdrojový port klienta ležal v rozmedzí portov 721 a 731. Používa nerezervované a voľné čísla portov od 1024 vyššie.
- LPR štandard uvádza, že tlačová úloha musí v kontrolnom súbore obsahovať veľkosť dát, ktorú klient odosiela na server. To znamená, že ak zvolíme automatické počítanie bajtov dát, úloha sa bude spoolovať dvakrát. Raz na klientovi, na zistenie veľkosti a raz pri odosielaní na tlačový server. Toto môže spôsobiť spomalenie rýchlosti prenosu. Bez tohto nastavenia sa ako veľkosť úlohy pošle implicitná, veľmi veľká hodnota nerešpektujúca skutočnú veľkosť dát, a po odoslaní celej úlohy sa stream jednoducho uzavrie.

I keď Windows server 2003 používa štandardný TCP/IP Port monitor ako prednastavený port pre spojenie so sieťovou tlačiarňou, stále podporuje LPR/LPD tlač kvôli interoperabilite s inými „legacy“ systémami a zariadeniami. „Legacy“ systém je zastaralý počítačový systém alebo program, ktorý je stále používaný, pretože uspokojuje používateľove potreby, i keď sa už na trhu vyskytujú omnoho modernejšie technológie.

2.5.2 Linux

Medzi vedúcimi implementáciami nasadzovanými v systéme Linux sú napr.:

- CUPS - Common UNIX Printing System, multiplatformné riešenie tlače pre prostredie typu UNIX. Je postavené na protokole IPP a poskytuje kompletné tlačové služby s podporou mnohých druhov tlačiarní. CUPS je šírené pod licenciou GNU GPL. Viac informácií je možné nájsť na domovskej stránke projektu [4].
- LPRng - Line Printer Remote Next Generation, je rozšírenou a vylepšenou implementáciou pôvodného BSD LPR spoolera. Poskytuje nielen požiadavky uvedené v pôvodnom [12], ale má implementovanú podporu napr.: pre dynamické presmerovanie tlačových front, podrobnú diagnostiku, vylepšenú kontrolu bezpečnosti a vylepšený autorizačný mechanizmus [9].

Kapitola 3

Monitorovanie sieťovej tlače

Počítačová tlač poskytuje efektívny a rýchly spôsob, ako preniesť elektronické dáta na papier, avšak tento druh efektivity, s ohľadom na všetky zdroje, ktoré používa, je veľmi nákladný. Potreba monitorovať a zisťovať náklady je znásobená hlavne na miestach, kde sa denne vytlačí veľký počet dokumentov, a náklady na ne môžu stúpať veľmi vysoko. Kontrola a monitorovanie zdrojov spojených s tlačou je pre takéto firmy spôsobom, ako optimalizovať náklady a zefektívniť využitie zdrojov, nehovoriac o šetrení životného prostredia a zamedzení plytvania.

Usporiadanie a návrh tlačovej infraštruktúry šitej na mieru konkrétnemu sieťovému prostrediu je veľmi komplexná procedúra. Zahŕňa analýzu aktívnych a pasívnych sieťových prvkov, počtu užívateľov a tlačiarní, stávajúci objem tlače a prenesených dát po sieti, priemernú nákladnosť na jeden vytlačený dokument, operačný systém na klientských a serverových staniciach a mnoho ďalších faktorov.

Zo strany používateľa, ktorý si je vedomý výdavkov na prevádzku tlačového zariadenia je dôležité, aby mal výdavky a teda aj konkrétne množtvo spotrebovaných prostriedkov pod kontrolou.

3.1 Tlačové riešenie SafeQ

SafeQ je komplexné riešenie pre riadenie nákladov na tvorbu dokumentov. Dokáže monitorovať náklady spojené s tlačou (spotreba papiera, tonera, energií a opotrebenia) a tieto náklady dokáže rozdeliť podľa jednotlivých užívateľov a skupín. Pokiaľ sa k zariadeniu pripojí aj terminál SafeQ, bude sprístupnená funkcia zabezpečenej (odloženej) tlače. Tá spočíva v tom, že sa úloha vytlačí užívateľovi až po jeho autorizácii na terminále (pinom, bezkontaktnou kartou).

Hlavné funkcie SafeQ je možné zhrnúť do niekoľkých základných bodov:

1. Účtovanie - presné sledovanie nákladov na reprografické výstupy (tlač, kópie, plotery) vrátane merania pokrytia, nákladov na toner, na farebnú a černobielu tlač, duplex apod.

2. Monitoring - hrubé sledovanie nákladov na malých lokálnych tlačiarňach pre umožnenie optimalizácie tlačového prostredia.

3. Vyhodnocovanie - priebežný prehľad nákladov spojených s reprografickými výstupmi podľa užívateľov, stredisiek a projektov.

4. Zabezpečenie - zabezpečenie prístupu k vytlačeným dokumentom (vytlačenie dokumentu až po autorizácii na SafeQ terminále u tlačiarne) a možnosť správy tlačovej fronty.

5. Riadenie prístupu - nastavenia kritérií pre zakázanie tlače úloh s určitým počtom strán na konkrétnu tlačiareň (s určitými obmedzeniami aj pre lokálne tlačiarne).

6. Užívateľský komfort - možnosť znovu tlačiť už vytlačené úlohy, funkcia follow-me (vytlačenie dokumentov podľa toho, na ktorej tlačiarni sa užívateľ autorizoval), náhľad na úlohu a iné.

SafeQ server zaisťuje správu tlačovných úloh v heterogénnych počítačových sieťach. Umožňuje spravovať tlač z týchto systémov: MS Windows NT (pomocou protokolu LPR), Novell (pomocou presmerovania NDPS fronty na LPD SafeQ), Linux (cez LPR), klony UNIX (cez LPR), MAC OS 9 a vyšší (pcez LPR), AS/400 a SAP (cez LPR).

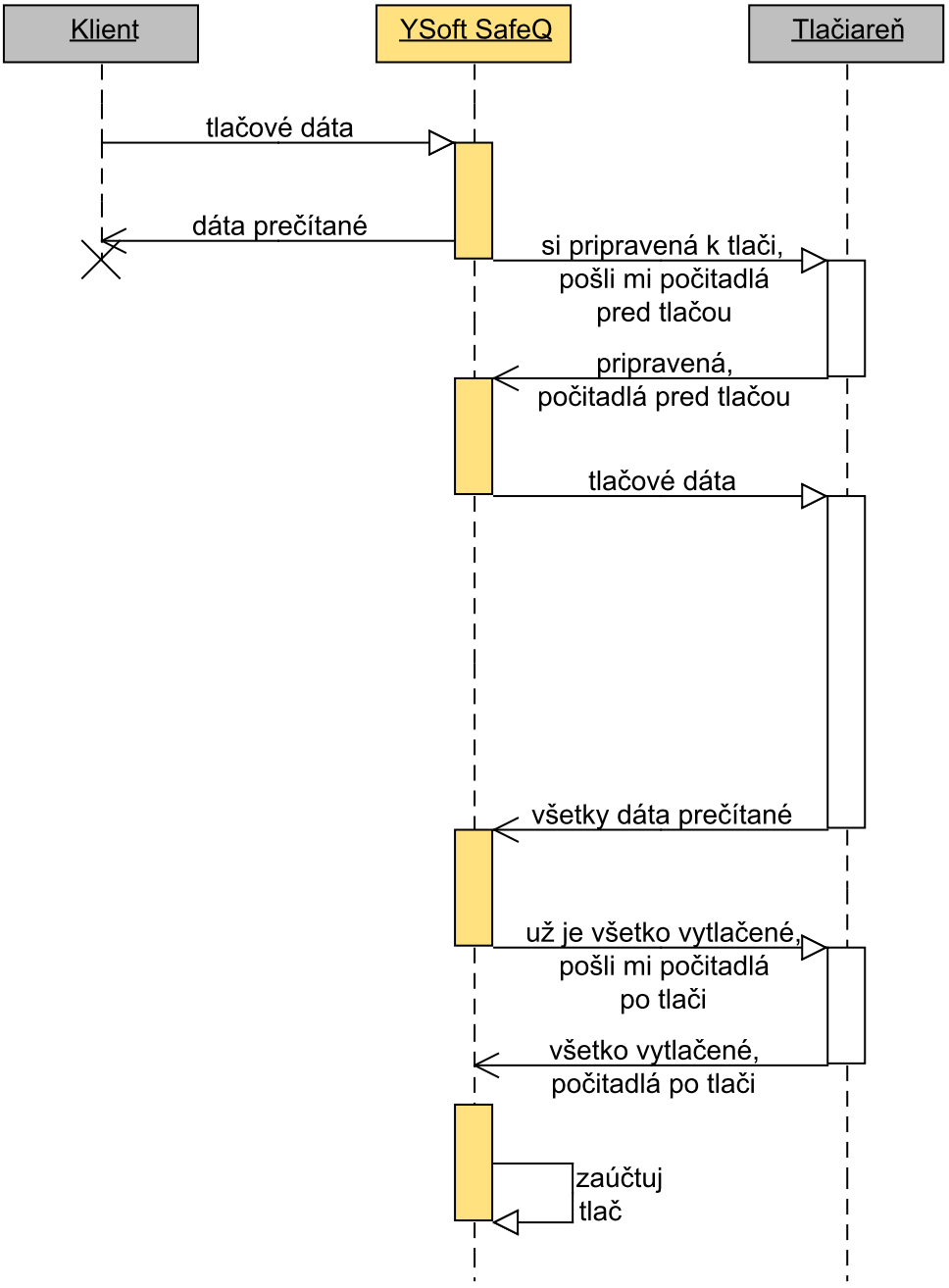
SafeQ dokáže presne identifikovať a monitorovať veľkosť, farebnosť, počet jednostranných a obojstranných (duplexných) výtlačkov. Umožňuje spoplatňovať jednotlivé výstupy tlače a to:

- cena za papier veľkosti A4,
- cena za papier veľkosti A3,
- cena za tlač jednej čiernobielej strany veľkosti A4,
- cena za tlač jednej farebnej strany veľkosti A4,
- cena za tlač jednej čiernobielej strany veľkosti A3,
- cena za tlač jednej farebnej strany veľkosti A3.

3.2 Spôsob komunikácie SafeQ a tlačiarne

SafeQ predstavuje centralizované riešenie riadenia tlače. V sieti vystupuje ako tlačový server, ktorý prijíma tlačové úlohy od klientských staníc, ktoré sú naň cez sieť pripojené. Pri príjme tlače pomocou LPR protokolu je užívateľ identifikovaný pomocou štandardných hlavičiek špecifikovaných protokolom LPR. Podľa toho či je užívateľ identifikovaný a nájdený v internej databáze užívateľov oprávnených k tlači cez konkrétnu frontu (jej meno sa nachádza v LPR hlavičke), je tlačová úloha poslaná na tlačiareň, ku ktorej daná fronta prislúcha. Spôsob akým dopraví tlačovú úlohu na koncové zariedenie je buď štardarným LPR alebo priamym TCP spojením typu AppSocket.

Účtovanie za tlač, ktoré patrí k jednej z hlavných možností využitia tohto systému, prebieha buď formou offline alebo online. Offline znamená, že tlačová úloha je spracovaná parserom (napríklad Ghostscript [6]) a hneď ako je odoslaná na tlačiareň, je aj zaúčtovaná. Parser prečíta tlačovú úlohu a pokúsi sa identifikovať konkrétne parametre úlohy ako je počet stránok, veľkosť strany a iné. Pri takomto prístupe zisťovania skutočných nákladov na vyprodukovanie tlačového dokumentu je hlavnou nevýhodou to, že ak napr. pri tlači 100 stranovej úlohy došlo k chybe tlačového zariadenia pri 50 stránke, užívateľovi je táto tlač, to znamená 100 stánok, zaúčtovaná ako celok i keď sa všetky strany nevytlačili. Naproti tomu, online účtovanie umožňuje presné účtovanie počtu vytlačených strán na danom zariadení. V tomto prípade je použitá SNMP (Simple Network Management Protocol) komunikácia



Obrázek 3.1: Komunikácia medzi YSoft SafeQ a tlačiarňou.

s daným zariadením. SafeQ tu vystupuje ako SNMP manažér a dotazuje sa SNMP agenta, tj. tlačového zariadenia, na konkrétne identifikátory určujúce jednotlivé počty a typy vytlačených stránok. Použitie online účtovania je závislé na bezproblémovom chode sieťového spojenia medzi SafeQ a tlačovými zariadeniami. Ďalej je závislé od prítomnosti SNMP agenta na danom zariadení a jeho schopnosti adekvátne a dostatočne rýchlo reflektovať aktuálny stav tlačového zariadenia v súvislosti s konkrétnym vyprodukovaným výstupom, ako je napríklad tlač dokumentu.

Ďalšie možnosti využitia a informácie o SafeQ je možné nájsť na stránkach produktu a firmy YSoft [7, 5].

3.3 Simple Network Management Protocol

Internet Engineering Task Force - IETF má na starosť definovanie štandardných protokolov použitých v sieťovej komunikácii, vrátane SNMP. IETF publikuje Request for Comments (ďalej len RFC), čo sú dokumenty obsahujúce špecifikácie sieťových protokolov využívaných v dnešných počítačových sieťach. Dokumenty predstavujúce konkrétny návrh protokolu sú najskôr publikované ako koncepčné návrhy. Až keď je finálna verzia návrhu odsúhlasená, potom je protokol označený ako štandardný.

Simple Network Management Protocol (ďalej len SNMP) poskytuje jednoduchú množinu operácií umožňujúcich monitorovanie a správu sieťových zariadení ako sú routre, switche, servery, tlačiarne a iné. Informácie, ktoré sa dajú monitorovať cez SNMP môžu byť rozličné, od štandardných, napríklad množstvo paketov prichádzajúcich na sieťové rozhranie, až po špeciálne ako je teplota vo vnútri routera.

Jadrom SNMP sú operácie dávajúce administrátorovi možnosť monitorovať a zmeniť stav zariadenia, v ktorom je implementovaný SNMP agent. Napríklad, SNMP môžme použiť na zakázanie príjmu dát na konkrétnom rozhraní routera alebo skontrolovať maximálnu rýchlosť sieťového pripojenia našej sieťovej karty. SNMP dokáže monitorovať teplotu vo switchi a upozorniť nás, pokiaľ je veľmi vysoká. SNMP je možné využiť pri správe mnohých typov zariadení. Kým predchodca SNMP Gateway Management Protocol SGMP, bol vyvíjaný na správu routerov, SNMP môže byť použité na správu Unixu, Windowsu, tlačiarní, zdrojov napájania a ďalších. Akékoľvek zariadenie, ktoré umožňuje príjem SNMP informácií môže byť monitorované, vrátane nielen fyzických zariadení, ale aj počítačových programov, ako napríklad webový server alebo databázový systém.

Viac informácií o SNMP môžme nájsť v [13], alebo v [26], z ktorej sú prebrané nasledujúce podsekcie.

3.3.1 Operácie

Formát dát prenášaný medzi manažérmi a agentami, sa nazýva Protocol Data Unit (PDU). Jednoduchý príklad komunikácie je ilustrovaný na obrázku 3.2. Každá z nasledujúcich SNMP operácií má štandardný PDU formát:

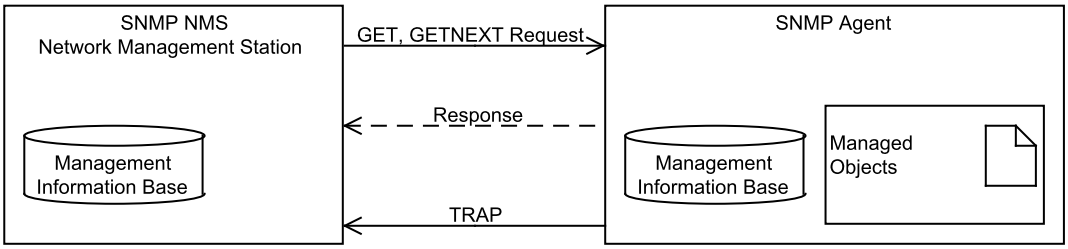
- GET - Požiadavok typu GET je odoslaný zo strany manažéra na agenta. Akonáhle agent požiadavok spracuje ihneď pošle odpoveď (typu RESPONSE). To, ktorú premennú má manažérovi vrátiť, zistí z požiadavku, ktorý obsahuje zoznam požadovaných premenných (angl. variable bindings).

- GETNEXT - Tento typ operácie je podobný typu GET, až na to, že manažér od agenta očakáva prvú nasledujúcu hodnotu objektu, akú požadoval.

- GETBULK (SNMPv2 a SNMPv3) - Umožňuje manažérovi získať maximámalnu možnú veľkosť odpovede na požiadavok.
- SET - dáva možnosť nastavovať agentove premenné, ak jeho objekty takúto operáciu dovoľujú.
- TRAP - agent asynchrónne posiela informácie o zmene hodnoty danej premennej, bez nutnosti manažéra sa na ňu dotazovať. Vo verzii SNMPv1 má iný formát PDU ako štandardné operácie GET alebo SET.
- NOTIFICATION (SNMPv2 a SNMPv3) - rozširuje PDU formát správy s operáciou TRAP na operáciu podobnej GET alebo SET.
- INFORM (SNMPv2 a SNMPv3) - pridáva možnosť spoľahlivého prenosu TRAPu na manažéra.
- REPORT (SNMPv2 a SNMPv3) - bola definovaná v prípravnej verzii štandardu SNMPv2, ale nebola nikdy implementovaná. Teraz je súčasťou SNMPv3 a má mať za účel definovať komunikáciu medzi SNMP entitami kvôli reportovaniu problémov v SNMP komunikácii.

3.3.2 UDP komunikácia

SNMP používa ako transportný protokol na prenos dát medzi manažérmi (z angl. Network Management Station, ďalej len NMS) a agentami User Datagram Protocol (ďalej len UDP). Tento typ protokolu bol vybraný kvôli jeho nespojovosti. Kvôli tomu je komunikácia medzi SNMP entitami, manažérmi a agentami označovaná ako nespoľahlivá. Na transportnej vrstve neprebieha potvrdzovanie neprijatých paketov (v prípade UPD, datagramov). Preposielanie datagramov je plne v réžii SNMP aplikácie. Typicky je to nastavením timeoutu a počtu opakovaní neúspešného dotazu. Ak manažér od agenta nedostane odpoveď v požadovanom čase, dotaz zopakuje až kým počet opakovaní nedosiahne nakonfigurované maximum.



Obrázek 3.2: Komunikácia medzi manažérom a agentom.

Pokiaľ nemá informácia, na ktorú sa manažér dotazuje kritickú dôležitosť, nespoľahlivosť protokolu UDP nepredstavuje až taký problém. Prinajhoršom NMS vytvorí požiadavok, na ktorý nedostane odpoveď. V prípade operácie typu trap je situácia iná. Ak agent pošle trap a ten sa u manažéra nikdy neobjaví, manažér už nikdy nemusí zistiť, či mu ho agent

ne/poslal. Agent totižto nevie, či manažér trap prijal alebo nie, pretože ten nie je povinný mu oznámenie o prijatí trapu zasielať.

Výhodou nespojovosti UDP je nízky nárok na vyťaženie siete. Pre ilustráciu, ak by SNMP pracoval nad TCP a sieťový tok by bol na pokraji ucpania, je lepším riešením použitie protokolu, ktorý sa po niekoľkonásobnom neúspešnom pokuse doručiť dáta vzdá spojenia, ako protokol, ktorý bude sieť zahlcovať pokusmi o retransmisiu paketov z dôvodu zabezpečenia spoľahlivosti.

3.3.3 MIB databáza

Databáza Management Information Base (MIB) je jednoduchý ASCII textový súbor, ktorý popisuje jednotlivé elementy SNMP ako strom dátových objektov. MIB môžme považovať za určitý typ slovníku SNMP jazyka, kde každé slovo, ktoré ma nejaký význam musí byť v tomto slovníku zapísané. SNMP MIB má stromovú štruktúru s rôznymi premennými, ktoré predstavujú jednotlivé listy podstromu. (Za list považujeme podstrom alebo vetvu, ktorá už neobsahuje ďalšie podstromy.)

Vnútorná definícia jednotlivých objektov, akou je spôsob pomenovania a dátové typy sú definované v štandardoch [14, 16]. Jedinečný identifikátor objektu (z angl. Object Identifier, ďalej len OID), je použitý na rozlíšenie jednotlivých MIB objektov v databáze. Definícia objektu pozostáva z:

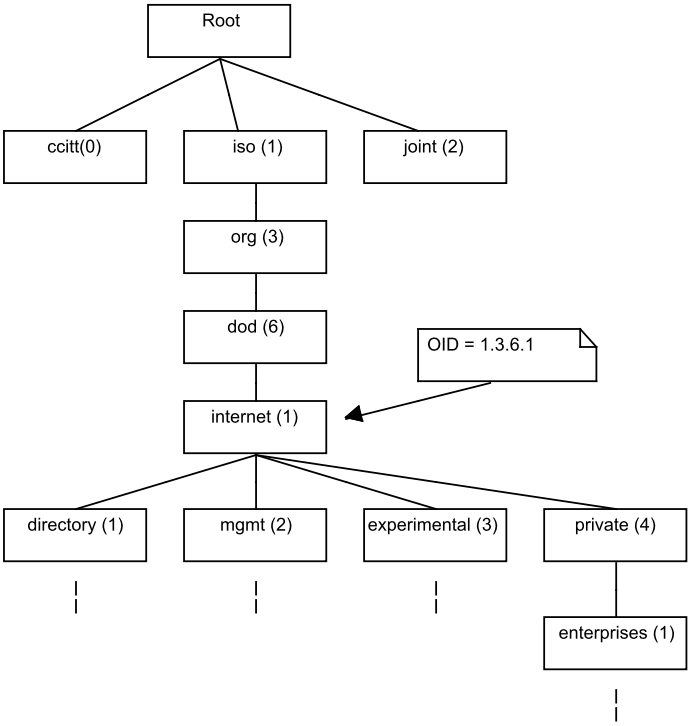
- Objektového identifikátora, OID - mena premennej, ktoré jednoznačne identifikuje daný objekt. Mená môžu mať dve formy, buď numerickú alebo slovnú. Objekty sú zoradené do stromovej štruktúry. OID je tvorené sériou čísiel oddelených bodkou, kde každé číslo predstavuje uzol v strome. Každý uzol má k sebe priradený aj reťazec, takže namiesto čísel môžme používať aj ľudsky lepšie zapamätateľné mená. Príklad MIB stromu ilustruje obrázok 3.3.

- Typu a syntaxe - dátový typ je definovaný podmnožinou notácie Abstract Syntax Notation One (ASN.1). Tá umožňuje špecifikovať, ako sú dáta reprezentované a následne prenášané medzi manažérmi a agentami. Viac o ASN.1 môžme nájsť na stránkach organizácie ITU-T a v dokumentácii [18].

- Kódovanie - inštancia objektu je zakodovaná do reťazca tvoreného bajtami použitím BER Basic Encoding Rules. BER definuje ako sú objekty zakódované a dekódované, aby bolo možné ich prenášať cez počítačovú sieť.

3.3.4 Verzie SNMP protokolu

SNMP verzia 1, označovaný tiež ako SNMPv1, je prvou verziou SNMP protokolu. Je definovaný v [13] a tiež označovaný ako historický IETF štandard. Bezpečnosť v SNMPv1 je založená na komunitách, heslách vo forme obyčajného textu, ktoré umožňujú získanie prístupu k MIB objektom daného agenta. V SNMPv1 sú typicky tri typy komunít: len na čítanie, na čítanie a zápis, a trap. Tieto tri komunity sú určené, ako už je poznať podľa ich názvu, na zjemnenie možnosti a nastavovanie prístupu k jednotlivým objektom. Použitie hesla určeného na čítanie umožňuje hodnoty objektov len čítať, na čítanie a zápis aj zapisovať, a hesla na nastavenie trapu na zasielanie trapov (asynchrónnych notifikácií) z agenta na NMS. I keď je SNMPv1 už historickou verziou štandardu, je stále primárnou implementáciu podporovanou mnohými výrobcami.



Obrázek 3.3: MIB strom objektov.

SNMP verzia 2 (SNMPv2) je definovaný v [20]. Rozširuje možnosti operácií o napr.: GETBULK, ktorá umožňuje manažérovi získať veľké množstvo dát z tabuľky naraz. Štandardná operácia GET umožňuje tiež získať viac objektov naraz, ale veľkosť odpovede zo strany agenta môže byť obmedzená a závislá na jeho možnostiach, takže pokiaľ by odpoveď mala prekročiť maximálnu možnú hranicu, agent vygeneruje chybovú odpoveď bez informácií, ktoré manažér požadoval. Naproti tomu odpoveď na GETBULK agentovi hovorí, aby do odpovede pridal maximálny počet premenných, ktoré pridať dokáže, aj keby musel niektoré vynechať.

SNMP verzia 3 je poslednou verziou SNMP. Jej hlavným zámerom je bezpečnosť SNMP komunikácie. Pridáva podporu overovania a bezpečnej komunikácie medzi SNMP entitami. Je definovaný v [19].

Kapitola 4

Testovanie softvéru

Na svete už takmer niet jediného odvetvia, ktoré by nebolo riadené počítačom. Počítače poskytujú nesporné výhody v urýchľovaní a zefektívňovaní vnútropodnikových procesov. Táto výhodnosť je ale obmedzená a silno závislá na bezchybnosti chodu využívaného počítačového systému, ktorého výpadok môže mať pre firmu fatálne následky. Dostatočná kvalita softvérového systému je jedným z dôležitých faktorov podieľajúcich sa na úspechu firmy na trhu.

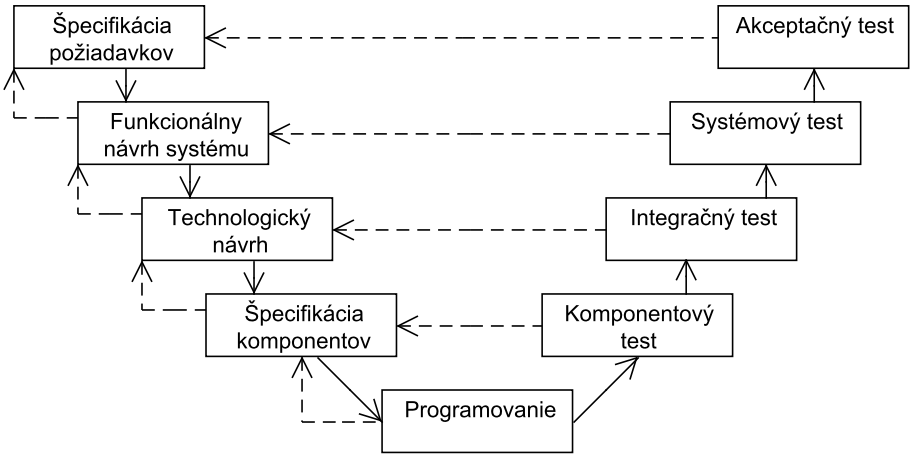
Mnohé firmy zaoberajúce sa vývojom softvéru si túto dôležitosť uvedomujú a snažia sa zlepšovať kvalitu po celú dobu vývoja ich produktov. Jedným zo spôsobov, ako toho dosiahnuť, je systematické vyhodnocovanie a testovanie čo i len čiastočnej verzie finálnej aplikácie.

Testovanie softvéru hraje veľmi dôležitú úlohu v procese vývoja softvéru. Prispieva k redukcii rizík súvisiacich s používaním počítačovej aplikácie. Na to, aby sme boli schopní opraviť chybu v programe, musíme ju najprv lokalizovať. Vtedy keď sa chyba objaví, je jediné čo o nej vieme, že sa stala. To čo ju v programe spôsobilo, avšak nevieme. Presné nachádzanie a oprava chýb majú na práci vývojári softvéru. Tento proces sa nazýva tiež ladenie (z angl. debugging). Ladenie a testovanie sú termíny, často považované za totožné. V skutočnosti tomu tak nie je. Zatiaľ čo ladenie má za úlohu lokalizovať a opraviť chybu, testovanie je systematický postup detekcie chýb. Celý proces systematického spúšťania testovanej aplikácie kvôli zisťovaniu korektnosti implementácie podľa vstupných požiadavkov, sa nazýva test. Okrem samotného spúšťania testov, je v testovacom procese dôležité, definovať a pripraviť testovacie dáta, testovací plán, implementáciu a analýzu testu.

V nasledujúcej sekcii si priblížime význam testovania v procese vývoja softvérového produktu. Pre túto sekciu sme z časti použili myšlienky z [24].

K vývoju každého softvérového projektu by sa malo pristupovať použitím nejakého konkrétneho modelu životného cyklu, ktorý bol vybraný ešte v predstihu pred samotným vývojom. Metodika výberu a použitia vhodného modelu je rozsiahle téma, je rozoberaná v mnohých publikáciách, napríklad v [27]. Všeobecný V-model [25] popisuje, akú dôležitú úlohu hrá testovanie. V ňom je testovanie rovnako dôležité ako vývoj a programovanie, a považuje sa za aktivitu prebiehajúcu počas celého vývoja. Toto symbolizujú dve vetvy písmena V ako je aj vidieť na obrázku 4.1. Ľavá vetva reprezentuje proces vývoja. Tu sa systém postupne dostáva cez fázu návrhu až k programovaniu. Pravá vetva reprezentuje integráciu a proces testovania, kde sú jednotlivé elementy programu spájané do väčších častí (subsystémov), a kde je testovaná ich funkčnosť. Integrácia a testovanie končí akceptačnými testami celého vyvíjaného systému.

Konštrukcia systému prebieha v niekoľkých fázach:



Obrázek 4.1: Všeobecný V-model v procese vývoja softvéru.

- Špecifikácia a analýza požiadavkov - zhromaždenie základných charakteristík a zákazníkových (alebo užívateľských) požiadaviek na vyvíjaný systém.
- Funkcionálny dizajn systému - požiadavky sú združené podľa funkcionality v novom systéme.
- Dizajn architektúry a použitej technológie - v tejto fáze je navrhovaná implementácia. To zahŕňa definíciu rozhraní v systéme a dekompozíciu systému do menších zrozumiteľných podsystémov. Každý podsystém potom môže byť vyvíjaný nezávisle na sebe.
- Špecifikácia komponentov (modulov) - chovanie, vnútorná štruktúra a rozhranie sú definované pre každý podsystém.
- Programovanie - Každý komponent (modul, trieda) je naprogramovaný v programovacom jazyku.

Činností odohrávajúcich sa v jednotlivých fázach je v skutočnosti oveľa viacej. Pokiaľ by sme v niektorej fáze urobili chybu, obyčajne najjednoduchším spôsobom ako ju nájsť, je hľadať ju na rovnakej úrovni abstrakcie, na ktorej bola vyprodukovaná. Pravá vetva definuje nasledujúce stupne testovacej abstrakcie:

- Jednotkový, komponentový test - overuje, či je každý komponent v súlade s jeho špecifikáciou
- Integračný test - kontroluje, či skupina komponentov spolupracuje tak, ako je to špecifikované v technickom návrhu systému.
- Systémový test - overuje, či systém ako celok splňuje špecifikované požiadavky
- Akceptačný test - kontroluje, či systém splňuje požiadavky, ako ich špecifikoval zákazník.

Na každom stupni musí prebehnúť kontrola, či výstupy vyprodukované vývojom splňujú požiadavky alebo sú dostatočne relevantné vzhľadom na daný stupeň abstrakcie.

Podľa V-modelu by sa mohlo zdať, že testovanie začína relatívne neskoro, až po implementácii. Avšak príprava testov (plánovanie a kontrola testov, návrh a analýza testov) začína ešte skôr a je vykonávaná paralelne, už počas vývoja na ľavej strane.

4.1 Testovanie výkonu SafeQ

SafeQ je veľmi komplexný systém s veľkým množstvom funkcií. Otestovanie všetkých je zložitý proces zaberajúci veľké kvantum času a ľudskej sily. Snaha jednotlivé úkony zautomatizovať hraje dôležitú úlohu v procese zefektívnenia a zrýchlenia testovacieho procesu.

Pri testovaní správnej funkcionality účtovania tlače, je dôležitá prítomnosť tlačového zariadenia, na ktorom sa má daná úloha vytlačiť. Takže je nutné, aby sieťová tlačiareň bola schopná komunikovať v sieti a pripravená k tlači.

4.1.1 Tlačové úlohy

Jednou z funkcionalít SafeQ, ktorá je z pohľadu zákazníka veľmi atraktívna, je presné účtovanie tlače. Pre organizáciu, ktorá toto tlačové riešenie u sebanasadzuje, môže znamenať chyba vo funkčnosti veľký problém, pretože aj malá nepresnosť v zaúčtovaní tlače, môže pri veľkom počte výtlačkov spôsobiť veľkú finančnú stratu.

Na to, aby sme dokázali správnu funkčnosť online účtovania, musíme najskôr izolovať vstupné dáta, ktorými budú jednotlivé tlačové úlohy s rôznymi parametrami.

Implementácia našej virtuálnej tlačiarne neumožňuje zisťovanie presných detailov o úlohe, ktorú prijme. Z časti by to bolo možné nasadením parsera tlačových úloh [6], ale to by znamenalo nárast potreby na systémové zdroje, ako je napríklad pamäť RAM. Použitie mennej konvencie, ktorou zaručíme správne, resp. požadované identifikovanie parametrov úlohy (počet strán apod.), je ideálnym riešením.

Takýmto spôsobom vytvoríme pseudo tlačové úlohy (obyčajné textové súbory) nesúce informácie o veľkosti a počte strán. Príkladom názvu takého súboru môže byť:

```
_size=a4_bw=0_col=1_duplex.printJob
```

kde `size=a4` predstavuje parameter veľkosti papiera, `bw=0` počet čiernobielych strán, `col=1` počet farebných strán, a `duplex` indikujúci, že úloha by na skutočnej tlačiarni bola vytlačená obojstranne.

4.1.2 Scenár testov

Konkrétny scenár testu je zapísaný vo forme skriptu, ktorý posiela množinu izolovaných úloh obsahujúcich rôzne kombinácie množstva farebných a černobielych strán a veľkosti papiera, postupne podľa nakonfigurovaného intervalu, na tlačový server SafeQ. Ten ich má za úlohu posielať na virtuálnu sieťovú tlačiareň a následne reagovať na zmenu jej počítadiel a jednotlivé simulované výtlačky účtovať ako keby to boli skutočné dokumenty. Spôsob, akým prenesieme tlačovú úlohu na vzdialený tlačový server SafeQ, je použitím štandardného príkazu `lpr`, ktorý podľa špecifikácie LPR protokolu automaticky dosadí meno užívateľa do kontrólneho súboru a umožní tak SafeQ identifikovať užívateľa a presmerovať dokument na tlačiareň.

Kapitola 5

Simulácia chovania MFD

Našou úlohou je nájsť riešenie, ktoré zjednoduší testovanie a to tým, že eliminuje nutnosť fyzickej prítomnosti, spravidla veľkého a hlavne drahého sieťového tlačového zariadenia.

Sieťová tlačiareň, tiež nazývaná multifunkčné zariadenie, dokáže často vykonávať mnohé činnosti, ako je kopírovanie, tlač, skenovanie a mnohé iné. Zameriame sa hlavne na simuláciu počítačovej tlače.

Implementovaný program, simulátor virtuálnej tlačiarne, ako už jeho meno napovedá, bude mať za úlohu nahradiť prítomnosť a simulovať činnosť skutočnej sieťovej tlačiarne, ako aj jej odozvu na prijaté úlohy. Zásadnou funkcionalitou, ktorú musíme riešiť je, ako z prijatých úloh zistiť, ako sa má náš program navonok chovať (aký typ úlohy prijala, koľko strán je čiernobielych alebo farebných a pod.). Spôsob, ktorý sme na toto zvolili, je jednoduchý. Úlohy, ktoré budeme na virtuálnu tlačiareň posielať, budú mať v sebe uložené hodnoty, na základe ktorých bude meniť svoj stav. NMS potom môže pomocou SNMP dotazov monitorovať jednotlivé stavy pred a po tlači, ako je to zobrazené na obrázku 3.1. K implementácii sme si vybrali objektovo orientovaný programovací jazyk Java [8]. Ten umožňuje jednoduché použitie cudzích knižníc, ktoré nám pomohli pri implementácii požiadavkov na funkcianalitu a v ďalších častiach si ich popíšeme.

5.1 Konfigurácia

Keďže simulátor tlačiarne je program komunikujúci pomocou počítačovej siete, je nutné, aby na platforme, na ktorej je spustený, bolo možné komunikovať so sieťovým rozhraním, počúvať na ňom príchodzie dotazy a odosielať na ne odpoveď. Dôležité je mať možnosť meniť porty na ktorých môže počúvať, z dôvodu možnosti súčasného spustenia viacerých virtuálnych tlačiarní naraz.

Ďalej je nutné konfigurovať spôsoby, pomocou ktorých identifikuje parametre vstupných úloh, ako sú počty strán ap. To je vyriešené možnosťou hľadania danej vlastnosti aplikáciou regulárneho výrazu, buďto na meno úlohy alebo na obsah tlačového súboru.

5.2 Príjem úloh

Náš implementovaný LPD démon odpovedá odporúčaniam uvedeních v [12] hlavne s ohľadom na príjem úlohy. Ostatné príkazy, ako je správa front, riadenie tlačových úloh sú pre jednoduchosť implementácie vypustené.

Komunikácia medzi klientom (SafeQ, alebo iným klientským programom, schopným komunikovať LPR protokolom) a serverom (virtuálnou tlačiarňou) je nasledovná:

1. LPR klient odošle príkaz na príjem úlohy.
2. Virtuálna tlačiareň potvrdzuje príkaz.
3. Po potvrdení príkazu na príjem úlohy, odosiela klient príkaz na príjem kontrólneho súboru obsahujúceho podpríkazy.
4. Po jeho potvrdení odošle kontrolný súbor, v ktorom, mimo iné, musí byť uvedená veľkosť dátového súboru.
5. Po potvrdení prijatia kontrolného súboru serverom, klient odošle dátový súbor.
6. Server jeho príjem klientovy potvrdí a spojenie sa uzavrie.

Odpoveď, ktorou server potvrdzuje klientovi prijatie príkazov, je bajt s nulovou hodnotou. Pokiaľ chce server indikovať chybu pri príjme úlohy, odošle bajt s akoukoľvek inou hodnotou.

Druhou možnosťou, ako prijať tlačové dáta, je cez rozhranie AppSocket, teda jednoduchým prečítaním všetkých dát, ktoré klient poslal v prúde (z angl. stream). Akonáhle sú zo strany klienta všetky dáta odoslané na server, klient ukončí spojenie a na žiadne potvrdzovanie nečaká. V tomto prípade neprebieha žiadna komunikácia pomocou kontrólneho a dátového súboru, ako je tomu v prípade LPR, ale tlačový server musí zistiť detaily o úlohe z jej obsahu.

Veľkosť zásobníka, v ktorom môžu byť tieto dáta uložené je konfigurovateľná, a po prečítaní dát tejto veľkosti sú ďalšie bajty úlohy jednoducho zahodené.

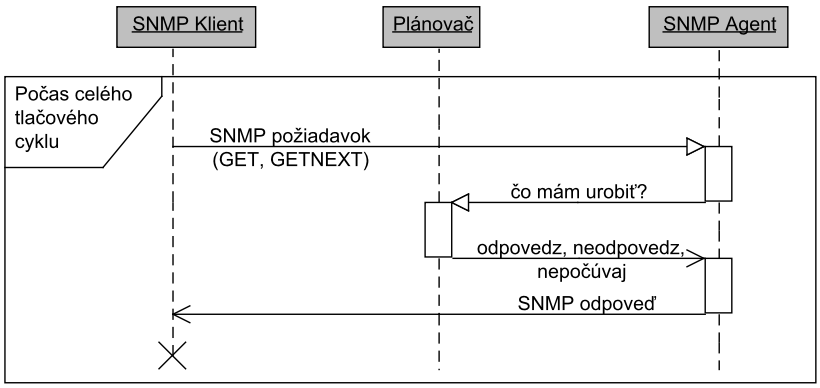
5.3 SNMP agent

SNMP agent má za úlohu reflektovať aktuálny stav virtuálnej tlačiarne. Pre implementáciu sme použili knižnicu SNMP4J [10]. Tá poskytuje aplikačné rozhranie (z angl. Application Programming Interface - API) pre vývoj SNMP manažérov a agentov. Je šírená pod Apache Public Licence [1] licenciou. Príklady použitia možno nájsť v jej vygenerovanej dokumentácii (angl. javadoc), ale hlavne v mail archívoch [11].

Náš agent umožňuje odpovedať na dotazy typu GET, GETNEXT, ktoré sú základnými operáciami SNMPv1. Ostatné príkazy a verzie SNMP protokolu sme pre jednoduchosť implementácie vynechali.

Stavy, do ktorých sa môže sieťová tlačiareň dostať v jednotlivých fázach tlače, pred, počas a po tlači, sú definované v [15]. Z hľadiska použiteľnosti zariadenia sú pre užívateľa významné dva stavy. Stav, v ktorom je zariadenie schopné prevádzky a je možné na ňom tlačiť dokumenty, a stav kedy neprijíma tlačové úlohy, napríklad z dôvody hardvérovej závady, alebo z dôvodu spracovávania predchádzajúcej úlohy. Nazvyme si tieto stavy pre jednoduchosť pripravená a zaneprázdnená. Typické činnosti, ktoré naša tlačiareň v tlačovom cykle vykonáva:

1. Pred príjmom úlohy je tlačiareň pripravená, odpovedá na SNMP dotazy a na prichádzajúce spojenie zo strany klienta na príjem dát.
2. Klient pomocou SNMP zistí či je pripravená, ak áno odošle tlačovú úlohu.



Obrázek 5.1: Diagram chovania systému pri SNMP komunikácii.

3. Akonáhle tlačiareň začne prijímať úlohu, prepne sa do stavu zaneprázdnená. V tomto stave zotrváva až kým danú úlohu nezpracuje.

4. Po spracovaní úlohy sa opäť prepne do stavu pripravená a zmení hodnoty svojich objektov podľa toho, aké výstupy vyprodukovala.

SNMP agent musí po celú dobu tlačového cyklu byť schopný odpovedať na SNMP dotazy, i keď práve virtuálna tlačiareň vykonáva nejakú činnosť, to je ilustrované na obrázku 3.2.

Pokiaľ SNMP manažér vie, na ktoré objektové identifikátory sa má dotazovať, môže identifikovať, koľko a aký typ stránok sa v daný moment vytlačilo. Typicky sú to listy z privátnej vetvy pomyselného MIB stromu. Ďalej sme implementovali možnosť zapamätania si posledných hodnôt objektových identifikátorov, aby bolo možné nájsť zariadenie v rovnakom stave v akom bolo pred reštartovaním.

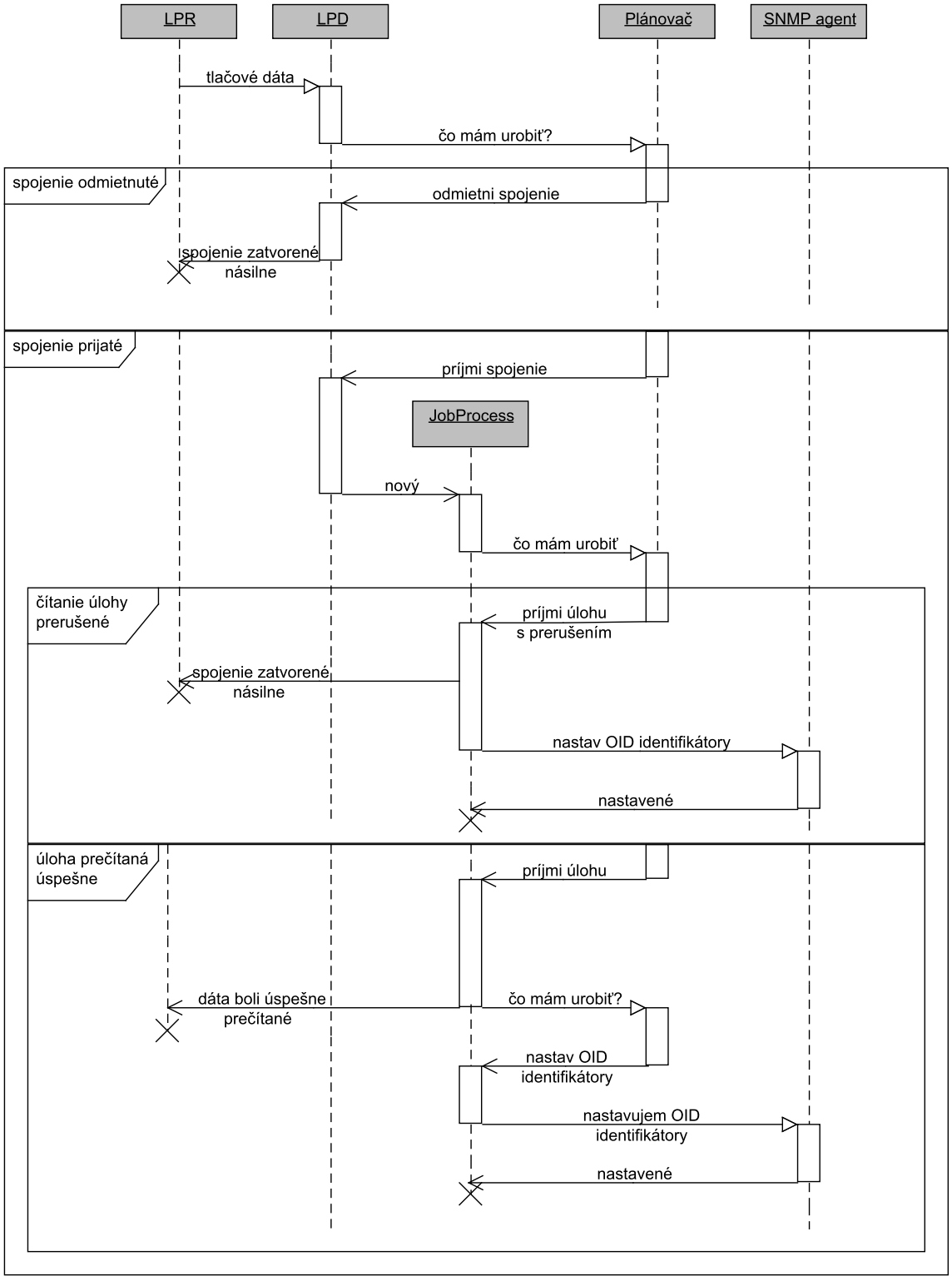
5.4 Nastavovanie chovania

Medzi hlavné funkcie našej tlačiarne patrí môžnosť zmeny hodnoty objektového identifikátora. Jej typ a hodnota sú konfigurovateľné. To nám umožňuje definovať si vlastnú množinu OID, ktoré sú nastavované podľa stavu v ktorom sa nachádza.

Chovanie našej tlačiarne v tlačovom cykle sme si rozdelili na jednotlivé fázi (ilustrované na obrázkoch 5.2 a 5.3):

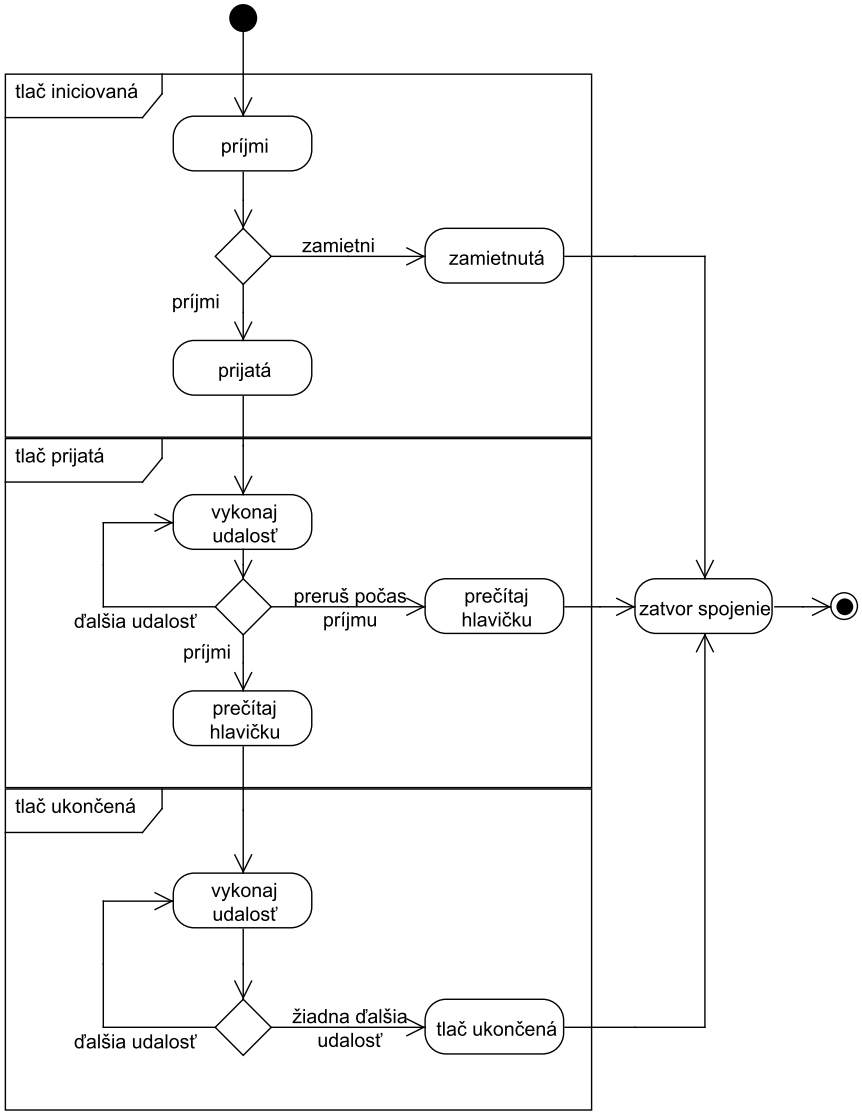
1. Tlač iniciovaná - to sa deje v okamihu, kedy klient naviaže spojenie s tlačovým serverom a chce sním komunikovať resp. posielať tlačové dáta. Server sa pozrie do súboru s chovaním a zistí, ako sa má zachovať - prijať alebo odmietnuť spojenie. Po prijatí spojenia sa dostáva do fázy tlač prijatá.

2. Tlač prijatá - v tejto fázy, vykoná ďalšie úlohy uvedené v súbore s chovaním až kým príde po úlohu typu prijať dáta alebo prijať dáta s prerušením. Pri prijatí dát s prerušením prečíta zo streamu len časť úlohy a následne spojenie násilne zatvorí. Takto je možné nasimulovať násilné prerušenie spojenia medzi klientom a serverom. Ak má prijať dáta bez prerušenia, prečíta dáta až do konca. Najskôr prečíta hlavičku tlačovej úlohy a potom ostatné dáta zahodí.



Obrázek 5.2: Diagram chovania systému pri príjme dát.

3. Tlač ukončená - sem sa dostane po prijatí tlačovej úlohy, a vykoná ostávajúce úlohy zo súbora s chovaním.



Obrázek 5.3: Diagram toku dát a aktivity systému.

Definícia úloh, ktoré sa majú v danej fáze odohrávať, je zapisaná v XML (z angl. Extensible Markup Language) súbore, kde sú jednotlivé množiny udalostí zoradené podľa času, kedy sú platné na ich vykonanie. Izolovali sme niekoľko konkrétnych udalostí napr.: prijmi, odmietni spojenie, prestaň počúvať prichádzajúce spojenia a zopár ďalších, ktorých význam je pevne daný, a vplýva na komunikáciu tlačiarne s klientom. Takýmto spôsobom je teda možné meniť chovanie virtuálnej tlačiarne, simulovať pomalé sieťové spojenie, prerušiť ho, zakázať komunikáciu alebo pridať stratovosť odpovedí na SNMP dotazy ap.

```
19.05.2009 12:58:29.355 | INFO  | SnmpAgent | SnmpAgent       | SnmpAgent sleeping before listening again: 0ms (SnmpAgent.java:61)
19.05.2009 12:58:29.355 | INFO  | SnmpAgent | SnmpV1Listener  | Starting Snmp Listener @192.168.1.11/161 (SnmpV1Listener.java:92)
19.05.2009 12:58:29.386 | INFO  | LPD   | LPD             | Waiting 0ms before LPD server socket start (LPD.java:59)
19.05.2009 12:58:29.402 | DEBUG | LPD   | LPD             | LPD listening, accepting connection [192.168.1.11/515] (LPD.java:119)
19.05.2009 12:58:29.402 | INFO  | main  | VPStartUp       | VP StartuUp Thread Dump: 6 active threads (VPStartUp.java:208)
19.05.2009 12:58:29.418 | INFO  | main  | VPStartUp       | ... Thread[main,5,main] (VPStartUp.java:211)
19.05.2009 12:58:29.418 | INFO  | main  | VPStartUp       | ... Thread[SnmpAgent,5,main] (VPStartUp.java:211)
19.05.2009 12:58:29.418 | INFO  | main  | VPStartUp       | ... Thread[DispatcherPool.0,5,main] (VPStartUp.java:211)
19.05.2009 12:58:29.418 | INFO  | main  | VPStartUp       | ... Thread[DispatcherPool.1,5,main] (VPStartUp.java:211)
19.05.2009 12:58:29.418 | INFO  | main  | VPStartUp       | ... Thread[LPD,5,main] (VPStartUp.java:211)
19.05.2009 12:58:29.418 | INFO  | main  | VPStartUp       | ... Thread[JD,5,main] (VPStartUp.java:211)
19.05.2009 12:58:29.418 | INFO  | JD    | JD              | Waiting 0ms before JD server socket start (JD.java:60)
19.05.2009 12:58:29.433 | DEBUG | JD    | JD              | JD accepting connection[192.168.1.11/9101] (JD.java:97)
```

Obrázek 5.4: Formát logovacieho súboru s informáciami o stave systému.

5.5 Logovanie

Na zisťovanie funkčnosti behu programu, ako aj informácií o priebehu príjmu a spracovania úloh, prípadne pre nachádzanie chýb, nám slúži logovací súbor. Podľa neho musí byť jasné, v akom stave sa v určitú dobu jednotlivé časti systému nachádzali.

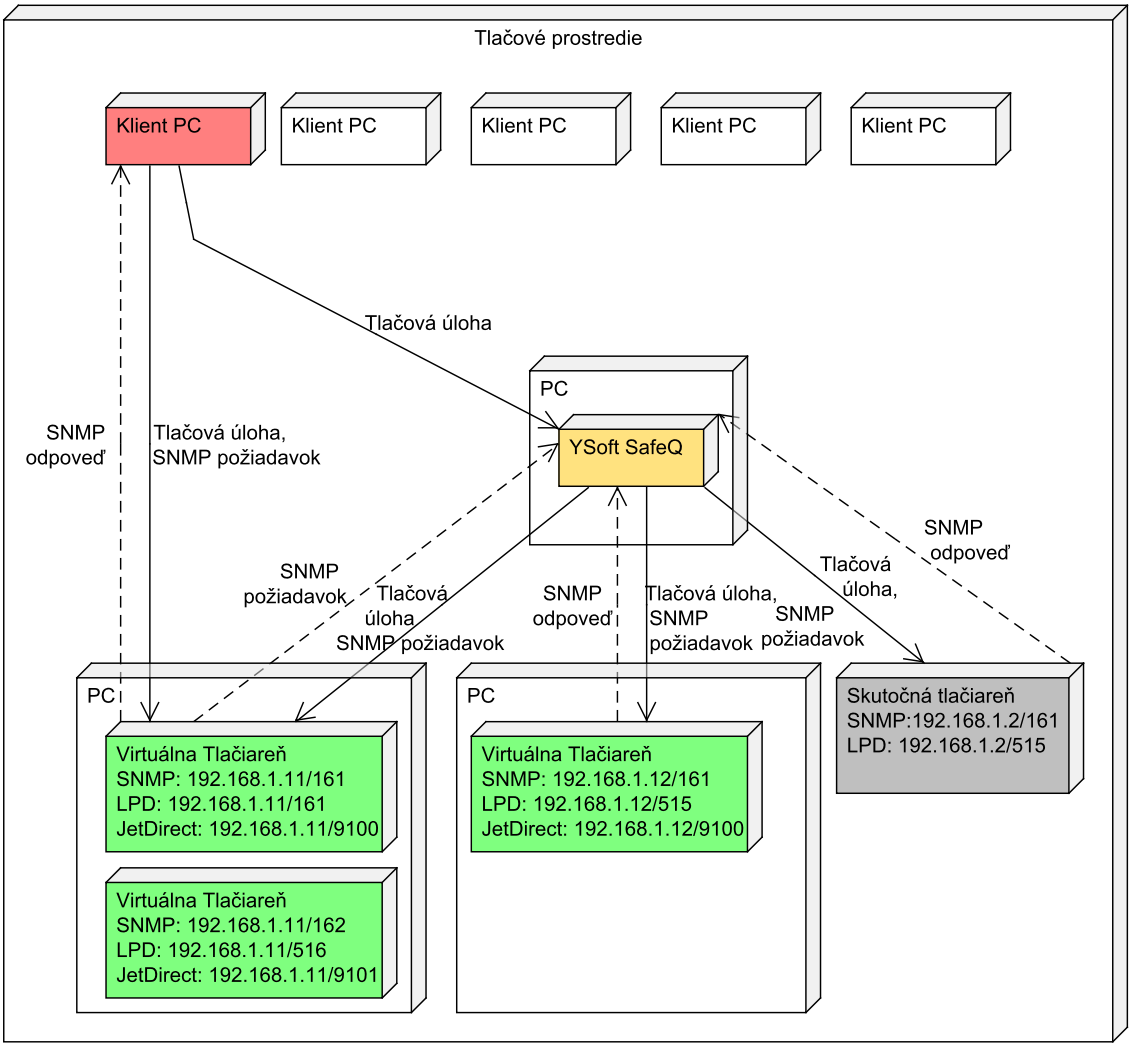
Využili sme logovacieho frameworku log4j [2]. Medzi jeho hlavné výhody patrí široká škála možností, ako meniť výsledný vzhľad logovacieho súboru, bez nutnosti zásahu do kódu. Takýto súbor by malo byť možné pohodlne čítať, a orientovať sa v ňom. Príklad formátu log súboru je ilustrovaný v 5.4.

Ďalej sme využili balík Apache Commons Logging [3], ktorý predstavuje tenký most medzi aplikáciou a použitým logovacím frameworkom. Program, ktorý využíva logovacie rozhranie Commons Logging, môže meniť východziu implementáciu logovacieho systému za inú, bez toho, aby musel meniť zdrojové súbory.

Obidva tieto frameworky sú šírené pod Apache Public Licence [1].

5.6 SNMP komunikácia

Nástrojov, ktorými je možné pomocou SNMP komunikovať je veľké množstvo. Pre demonstračné a testovacie účely sme použili program Net-SNMP [22], šírený pod Net-SNMP licenciou [21], ktorý dokáže komunikovať so SNMP agentom a zisťovať tak aktuálne hodnoty z jeho MIB databáze.



Obrázek 5.5: Použitie virtuálnej tlačiarne v testovacom prostredí.

Kapitola 6

Záver

V našej bakalárskej práci sme si priblížili možnosti tlače v počítačovej sieti. Zaoberali sme sa spôsobmi, ako tlačové úlohy dostať na tlačové zariadenia, ktoré sú na nich pretvárané do hmatateľnej podoby.

Predstavili sme si spôsob monitorovania sieťových zariadení, a jeho využitie v skutočnom živote.

Následne sme si ukázali, ako je dôležité, zvyšovať kvalitu softvérových produktov, ako aj spôsoby, ktorými sa toho dá dosiahnuť.

Čo sa týka splnenia našich cieľov, navrhovaný systém sa nám podarilo implementovať. Simulátor virtuálnej tlačiarne je program spúšťaný z príkazového riadku a konfigurovaný pomocou konfiguračných súborov. Je schopný prijímať tlačové úlohy a meniť svoj stav podľa nastavenia. Tým sa dokáže priblížiť správaniu skutočnej tlačiarne v sieti.

Pokračujúc v rozvýjaní témy, popísali sme si základné myšlienky ako má tento program fungovať, aby dokázal plniť funkiu nástroja, ktorý je možné použiť pri simulácii skutočného tlačového prostredia.

6.1 Možnosti rozšírenia funkcionality

Možností, ako výsledný program vylepšiť a rozšíriť je veľa. Uveďme si tu aspoň niektoré.

Keďže konfiguráciu je možné meniť editáciou textových súborov, je dôležité, aby ich užívateľ vyplnil správne. V tom by nám mohlo pomôcť grafické užívateľské rozhranie (z angl. Graphical User Interface - GUI), ktoré by užívateľa v prípade nesprávnej alebo nezmyselnej konfigurácie upozornilo na chybu.

Toto GUI by mohlo byť rozšírené o možnosť vzdialenej správy jednotlivých virtuálnych tlačiarní.

Náš simulátor je možné nakonfigurať tak, aby počúval na rôznych portoch, a tým umožniť, aby bol spustený viacnásobne naraz. Skutočná sieťová tlačiareň, zapojená v sieti, má vlastné sieťové rozhranie na ktorom komunikuje. To znamená, že ak by sme chceli, aby naša tlačiareň používala vlasté rozhranie, a platforma, na ktorej je spustená, má len jednu sieťovú kartu, museli by sme použiť nejaký druh virtualizačného softvéru (napr.: WMware [23]).

Mohli by sme implementovať rozšírenie SNMP agenta o podporu ďalších operácií ako sa ho dotazovať na odpoveď.

Keďže naša aplikácia obsahuje len obmezenú implementáciu persistentnej vrstvy, bolo by vhodné rozšíriť ju o nejaký typ dotabázy, kde by sme zaznamenávali príjem dát a celkové

výstupy systému.

6.2 Prínosy práce

Pri implementácií nášho simulátora, sme sa museli potýkať s rôznimy problémami spojenými s vývojom softvéru. Snažili sme sa dodržať dobré praktiky písania kódu, ktorý by mal byť čitateľný a rozšíriteľný. V jeho vylepšovaní budeme pracovať aj naďalej v spolupráci s českou firmou YSoft [7].

Mimo to, že sme získali skúsenosti z oblasti vývoja jednoduchého softvérového produktu, sme prišli k poznaniu, že testovanie softvéru nie je len automatické klikanie myšou, ale že je aj o systematickom prístupe k testovaniu a o vývoji nástrojov, ktoré ho zefektívňujú.

Literatura

[1] Apache License, Version 2.0 - The Apache Software Foundation. [online], [cit. 2009-05-18].
URL <http://www.apache.org/licenses/LICENSE-2.0.html>

[2] Apache log4j 1.2 - log4j 1.2. [online], [cit. 2009-05-19].
URL <http://logging.apache.org/log4j/1.2/index.html>

[3] Commons Logging - Overview. [online], [cit. 2009-05-19].
URL <http://commons.apache.org/logging/>

[4] CUPS. [online], [cit. 2009-05-15].
URL <http://www.cups.org>

[5] Finance (and) Insurance - SafeQ. [online], [cit. 2009-05-16].
URL <http://www.safeq.eu>

[6] Ghostscript, Ghostview and GSview. [online], [cit. 2009-05-17].
URL <http://pages.cs.wisc.edu/~ghost>

[7] Home - YSoft, Ltd. [online], [cit. 2009-05-16].
URL <http://www.ysoft.com/home/index.html>

[8] Java SE at a Glance. [online], [cit. 2009-05-19].
URL <http://java.sun.com/javase>

[9] LPRng Web Page. [online], [cit. 2009-05-15].
URL <http://www.lprng.com>

[10] SNMP4J - Free Open Source SNMP API for Java. [online], [cit. 2009-05-19].
URL <http://www.snmp4j.com>

[11] The SNMP4J Archives. [online], [cit. 2009-05-19].
URL <http://lists.agentpp.org/pipermail/snmp4j>

[12] Line Printer Daemon Protocol. [online], August 1990, [cit. 2009-05-15].
URL <http://www.ietf.org/rfc/rfc1179.txt>

[13] Simple Network Management Protocol (SNMP). [online], Máj 1990, [cit. 2009-05-15].
URL <http://www.faqs.org/rfcs/rfc1157.html>

[14] Structure and identification of management information for TCP/IP-based internets. [online], Máj 1990, [cit. 2009-05-19].
URL <http://www.faqs.org/rfcs/rfc1155.html>

[15] Printer MIB. [online], Marec 1995, [cit. 2009-05-15].
URL <http://www.faqs.org/rfcs/rfc1759.html>

[16] Structure of Management Information Version 2 (SMIv2). [online], April 1999, [cit. 2009-05-19].
URL <http://www.faqs.org/rfcs/rfc2578.html>

[17] Internet Printing Protocol/1.1: Model and Semantics. [online], September 2000, [cit. 2009-05-15].
URL <http://www.ietf.org/rfc/rfc2911.txt>

[18] Abstract Syntax Notation One (ASN.1): Specification of basic notation. [online], Júl 2002, [cit. 2009-05-18].
URL
<http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf>

[19] Introduction and Applicability Statements for Internet-Standard Management Framework. [online], December 2002, [cit. 2009-05-15].
URL <http://www.faqs.org/rfcs/rfc3410.html>

[20] Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP). [online], December 2002, [cit. 2009-05-15].
URL <http://www.faqs.org/rfcs/rfc3416.html>

[21] License. [online], Marec 2007, [cit. 2009-05-19].
URL <http://net-snmp.sourceforge.net/about/license.html>

[22] Net-SNMP. [online], Marec 2007, [cit. 2009-05-19].
URL <http://net-snmp.sourceforge.net/>

[23] VMware Virtualization - Optimize IT Resources with Virtual Technology. [online], 2009, [cit. 2009-05-19].
URL <http://www.vmware.com/technology/virtualization.html>

[24] Andreas Spillner, H. S., Tilo Linz: *Software Testing Foundations*. dpunkt.verlag, 2006, ISBN 3-89864-363-8, 266 s., [cit. 2009-05-18].

[25] Boehm, B. W.: Guidelines for Verifying and Validation Software Requirement and Design Specifications. Euro IFIP, 1979, s. 711–719.

[26] Douglas Mauro, K. S.: *Essential SNMP, 2nd Edition*. O'Reilly, September 2005, ISBN 0-596-00840-6, 460 s.

[27] IEEE/EIA: Information Technology - Software life cycle processes. Technická Zpráva 12207.0, IEEE/EIA, 1996.

[28] Microsoft: How Network Printing Works. [online], Marec 2003, [cit. 2009-05-14].
URL <http://technet.microsoft.com/en-us/library/cc783789.aspx>

[29] Microsoft: Network Printer Ports. [online], Marec 2003, [cit. 2009-05-14].
URL <http://technet.microsoft.com/en-us/library/cc728404.aspx>

[30] Microsoft: Enhanced Metafile Format Specification. [online], 2009, [cit. 2009-05-16].
URL <http://msdn.microsoft.com/en-us/library/cc204166.aspx>