Portál pre odborné publikovanie ISSN 1338-0087

Synchronizácia dát

Javorský Stanislav · Elektrotechnika, Informačné technológie, Študentské práce

12.05.2008

V odbornom príspevku sa budem zaoberať univerzálnym návrhom synchronizácie údajov v rámci lokálnej siete a v rámci internetu pre rôzne štruktúry dát a rôzne dátové zdroje. Funkčnosť navrhnutej synchronizácie bude overená aplikáciou, ktorá bude zostrojená pre synchronizáciu údajov v oblasti dochádzkových systémov.

1. Úvod

Synchronizáciou a konverziu údajov sa zaoberá každý systém, ktorý čerpá dáta aj z iných zdrojov. Synchronizácia prebieha často medzi databázovými systémami, ale stretávame sa s ňou aj v iných situáciách. V bežnom živote sa stretávame so synchronizáciou údajom napríklad medzi mobilným telefónom a počítačom, v technickej praxi je to napríklad sťahovanie údajov z meteorologických staníc, v oblasti informačných technológií je to najme B2B (business-to-business) kde sa rieši otázka synchronizácie údajov medzi rôznymi informačnými systémami (banky, podniky, štátne inštitúcie).

Dáta, ktoré sa synchronizujú existujú v rozmanitých formách (databázy, zásobníky, registre, súbory – xml, pdf, txt, doc, xls) a aj prístup k týmto dátam je rôzny (sériová linka, ethernet, IrDA, Wifi), a práve kvôli rozmanitosti dát existuje celá rada rozmanitých synchronizačných nástrojov. Spoločnosti neustále investujú nové prostriedky do vývoja nových synchronizačných nástrojov a programátori riešia otázku synchronizácie znova a znova, pričom vďaka rôznorodosti procesov synchronizácie navrhujú pre rôzne druhy synchronizácie rôzne architektúry programov. V konečnom dôsledku vznikajú stále nové a nové aplikácie, ktoré so sebou prinášajú nový vzhľad, nový spôsob ovládania a nové problémy pri ich používaní.

Cieľom mojej práce bolo navrhnúť univerzálny spôsob synchronizácie, ktorý zabezpečí bezpečnosť prenosu dát (ochrana pred stratou dát, alebo pred ich neželaným zneužitím), a ktorý bude dostatočne rýchli a flexibilný. Zameral som sa predovšetkým na rozšíriteľnosť a modulárnosť systému, aby sa pre nový druh synchronizácie nemusela tvoriť celá logika od začiatku, ale aby existovala možnosť pridať k existujúcemu systému len nový blok, ktorý zadefinuje štruktúru nových dát a spôsob prístupu k týmto dátam a okamžite sa stane súčasťou jedného komplexného synchronizačného systému.

Problémy, ktorá so sebou synchronizácia prináša sú najme:

1. rozmanitosť štruktúry dát a dátových zdrojov
2. strata dát pri výpadku siete
3. bezpečnosť dát (ochrana pred odcudzením)
4. veľké objemy dát a s časová náročnosť prenosu
5. požiadavka na včasné doručenie dát

2. Charakteristika synchronizácie

Synchronizáciou rozumieme výmenu dát medzi dvoma úložiskami. Pri tejto výmene identifikujeme dve strany. Jednou stranou, je dátové úložisko, ktoré dáta vlastní a potrebujeme tieto dáta preniesť. Túto stranu nazývame zdrojom dát (source). Druhá strana potrebuje prebrať dáta zo zdroja dát a túto stranu nazývame cieľ dát (destination). Synchronizačná úloha nám teda zabezpečuje prenos dát zo zdroja do cieľa.

Podľa možností, ktoré zariadenie ponúka, rozdeľujeme synchronizáciu na plnú, čiastočnú a základnú.

2.1. Druhy synchronizácie

Pri plnej synchronizácii je možné na zariadení spustiť program a dáta sú prístupné na čítanie aj na zápis. Príkladom môže byť napríklad server s SQL databázou, z ktorého potrebujeme synchronizovať údaje do centrálneho systému. Na serveri je možné spustiť program, ktorý bude dáta synchronizovať a k dátam je prístup na čítanie aj zápis.

Pri čiastočnej synchronizácii nie je možné na zariadení spustiť program, ale dáta môžeme čítať aj zapisovať (mazať). Napríklad spojenie fotoaparát – PC. Do fotoaparátu vlastný program nenahráme, ale dáta v ňom môžeme čítať aj zapisovať. Dáta musíme synchronizovať z vonku. Preto je tento prístup nie plný, ale čiastočný.

Pri základnej synchronizácii je prístup k dátovému zdroju možný len na čítanie. Tento stav nastáva veľmi často na internete, kde máte prístup k množstvu dát, ale tieto dáta nemôžete nijako ovplyvniť. Veľmi dobrý príklad je kanál RSS. Váš počítač môže z tohto kanála len čítať – nemôže doň zapisovať ani z neho nič vymazať. Príkladom by mohli byť aj kurzy mien, vydávané Národnou Bankou Slovenska. Tieto kurzy môžeme len čítať a nemôžeme ich meniť.

3. Návrh synchronizácie

3.1. Import a export

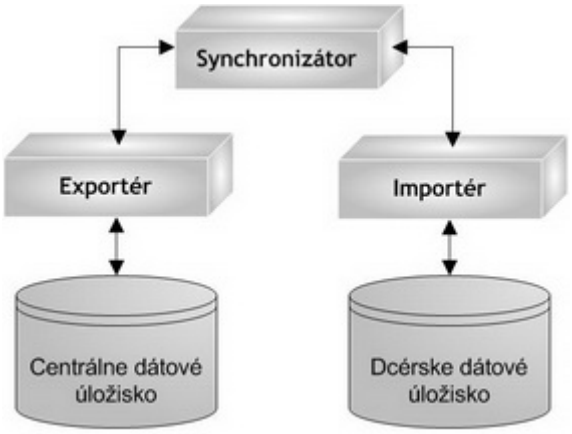
Synchronizácia pozostáva z prenosu údajov zo zdroja do cieľa a tento proces je rovnaký bez ohľadu na štruktúru a formát dát a tiež bez ohľadu na typ spojenia so zdrojom dát. Ak zoberiem do úvahy odlišnosti procesu synchronizácie pre jednotlivé typy synchronizácie, ktoré som opísal vyššie, rozdiel je len v tom, že v procese synchronizácie sa neuskutočnia niektoré kroky.

Posledné zjednodušenie je v definovaní tzv. importéra a exportéra. Pri synchronizácii totiž dochádza k tomu, že program, ktorý dáta synchronizuje, musí tieto dáta najprv z nejakého zdroja načítať teda importovať a následne tieto údaje do iného dátového úložiska uložiť teda exportovať.

3.2. Importér a exportér

Celé riešenie synchronizácie je v jednoduchom návrhu importéra, ktorý sa stará o zdroj dát, a exportéra, ktorý sa stará o cieľ dát.

Jednoduché by bolo, ak by importér a exportér mohli pracovať jednotlivo najprv import a potom export údajov a nemuseli by o sebe vôbec vedieť. To však nie je možné. Musí totiž existovať spätá väzba, ktorá zabezpečí, aby sa mohli dáta zo zdroja po úspešnom exporte vymazať, prípadne, aby sa mohla spustiť iná úloha (dodatočné spracovanie dát, notifikácia u vykonanej synchronizácii a podobne). Z toho jednoznačne vyplýva, že importér a exportér musia byť navzájom riadený. Riadenie týchto dvoch blokov, zabezpečuje synchronizátor. Bloková štruktúra synchronizačného procesu je na obrázku Obr.1.

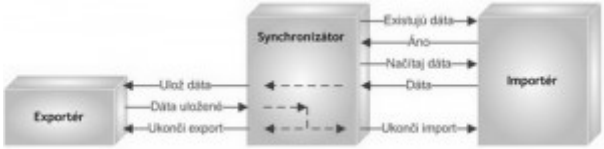


Obr. 1. Synchronizácia (bloková schéma)

3.3. Synchronizátor

Synchronizátor je hlavný riadiaci blok, ktorý kontroluje a riadi celý proces synchronizácie od začiatku – vytvorenie importéra a exportéra, cez riadenie ich vzájomnej komunikácie a nakoniec zabezpečuje ukončenie synchronizačného procesu a oznamuje stav, v akom synchronizačný proces skončil.

Schématické riadenie synchronizačného procesu je na obrázku Obr.2.



Obr. 2. Princíp synchronizácie

Synchronizátor najprv vytvorí importéra a pomocou neho zisťuje, či existujú dáta, ktoré je potrebné synchronizovať. Ak takéto dáta existujú, tak vytvorí exportéra, načíta dáta z importéra a predá ich exportérovi, aby ich uložil. Exportér odpovedá na

uloženie dát, čo dáva synchronizátoru signál, že môže ukončiť import aj export. Ukončenie importu a exportu dáva jednotlivým blokom možnosť, aby vykonali úlohy, ktoré má každý z nich nezávisle definované. Importér môže napríklad prenesené údaje vymazať a poslať správu o vymazaní kompetentnej autorite a niečo podobné môže vykonať aj exportér.

3.4. Synchronizačný kľúč

Pre bezpečnosť a spoľahlivosť synchronizácie je dôležité, že importér posiela spolu s dátami aj kľúč, podľa ktorého tieto dáta jednoznačne identifikuje. Importér si svoje dáta týmto kľúčom podpíše a synchronizátor sa postará, aby tento kľúč dostal aj exportér. Následne, keď exportér oznámi, že ukončil uloženie dát (spolu s oznámením posiela synchronizátoru aj kľúč), synchronizátor tento kľúč využije a pošle ho importérovi aj exportérovi spolu so správou o úspešnom ukončení prenosu. Importér dokáže na základe tohto kľúča identifikovať údaje, ktoré boli prenášané a môže ich napríklad zmazať. Podobne aj exportér môže podľa tohto kľúča tieto záznamy nejakým spôsobom ďalej spracovať. Týmto kľúčom môže byť napríklad názov a plná cesta k súboru, ktorý bol prenášaný, môžu to byť identifikačné čísla záznamov v databáze alebo to môže byť nejaký druh hash-u (záleží na konkrétnej aplikácii).

3.5. Výpadok spojenia

Mnohé dnešné synchronizačné techniky trpia najmä dvomi chorobami. V prvom rade je to strata dát pri výpadku spojenia, v druhom rade je to neužitočnosť dát v prípade, ak prichádzajú neskoro. Čo sa týka druhej choroby, túto rieši importér, pretože je to on, kto je schopný zistiť, ktoré dáta sú a ktoré dáta nie sú zaujímavé pre synchronizáciu. V konečnom dôsledku je správanie sa importéra vecou konkrétnej aplikácie.

Čo sa týka straty dát pri výpadku spojenia, pri návrhu synchronizácie som kládol dôraz v prvom rade na túto požiadavku. Návrh som preto spravil tak, aby mohlo byť spojenie prerušené v ktorejkoľvek okamihu synchronizačného procesu. Je len vecou konkrétnej implementácie, ako sa proces pri tomto stave zachová, dôležité však je, že vďaka potvrdzovaniu a vďaka prenášaniu identifikačného kľúča, nemôže prísť k straty dôležitých dát.

Všetky metódy majú svoje výhody aj nevýhody a samozrejme aj mnou navrhnutá metóda musí platiť daň za bezpečnosť synchronizácie. V prípade synchronizácie, ktorej ako dátový zdroj slúži napríklad zásobník (môžete čítať len jednu hodnotu a kým ju nevymažete, nemôžete čítať ďalšiu) dochádza k situácii, kedy je nutné spúšťať synchronizačný proces toľko razy, koľko je záznamov v zásobníku. Len tak sa docieli bezpečné prenesenie všetkých dát. Cenou za bezpečnosť synchronizácie je v tomto prípade čas, pretože rýchlejšie by bolo naraz vyčítať zo zásobníka všetky dáta a následne ich poslať. Tu však môže nastať výpadok spojenia a strata týchto dát. Samozrejme, ak je pri synchronizácii prednejšia rýchlosť pred bezpečným prenosom, je možné importéra upraviť.

3.6. Popis importéra

Existujú dáta –metóda zisťuje, či existujú dáta, ktoré je potrebné preniesť.

Načítaj dáta – importér na toto volanie vráti dáta, ktoré potrebuje synchronizovať. Formát týchto dát nie je pre synchronizátor dôležitý, pretože s týmito dátami nič nerobí. Dáta sú určené pre exportéra. Spolu s týmito dátami je tiež prenášaný kľúč, ktorý prenášané dáta jednoznačne identifikuje.

Ukonči prenos – ukončenie prenosu je volané len v prípade úspešného skončenia prenosu. Importér môže na základe identifikačného kľúča vymazať prenášané záznamy, alebo môže vykonať inú operáciu.

3.7. Popis exportéra

Ulož dáta – exportér dostane dáta, ktoré musí uložiť. Pri ukladaní dát musí kontrolovať, či ukladané dáta už raz nedostal. Tento stav môže nastať, ak v predchádzajúcom synchronizačnom cykle nastal výpadok spojenia po uložení a importér sa nedozvedel o úspešnom uložení. Samozrejme je vecou konkrétnej aplikácie, ako sa exportér zachová.

Ukonči prenos – po tom, ako boli dáta úspešne uložené, môže exportér poslať správu o uskutočnenom prenose, alebo môže prenesené dáta ešte dodatočne spracovať.

4. Aplikácia

4.1. O aplikácii

Aplikácia je vytvorená nad platformou .NET v jazyku Visual C# v prostredí Microsoft Visual Studio. Názov aplikácie je DataReader.

Na účely aplikovania navrhnutej metódy synchronizácie som vymyslel príklad pre využitie v dochádzkových systémoch, kde je potrebné synchronizovať dáta (snímania) z rôznych dochádzkových terminálov do databázy a opačne, je potrebné nahrať zamestnancov a ich čipové karty do terminálov.

Podporované „dátové zdroje“:

Dochádzkový terminál CipherLab 5100

1. pamäť typu flash, prístup cez firmware
2. pripojenie cez RS232 a Ethernet

Dochádzkový terminál Logic 310

1. pamäť zásobníka typu LIFO
2. terminál je možné pripojiť cez Ethernet

Dátový súbor CPT

1. Textový súbor s koncovkou *.dat s presne definovaným formátom

SQL databáza dochádzkového systému

Všetky snímania, ktoré boli na dochádzkových termináloch uskutočnené sú

synchronizované do databázy dochádzkového systému. Medzi databázou a dochádzkovými terminálmi sú navyše synchronizované mená zamestnancov a ich čipové karty.

4.2. Požiadavky

Teória synchronizácie je jednoduchá, pretože je v značnej miere intuitívna. Problém vzniká až v okamihu jej použitia, pretože mnohé spoločnosti riešia problém synchronizácie v momente, keď je to potrebné a vždy stavajú celý proces synchronizácie opäť od základu. Neuvedomujú si totiž, že je potrebné budovať systém, ktorý bude zastrešovať celý proces a nový druh synchronizácie sa tak stane len doplnením jedného modulu, ktorého úlohou je v konečnom dôsledku len konverzia údajov. O všetko ostatné sa totiž stará už zostrojený synchronizačný systém. Hlavné požiadavka na aplikáciu je smerovaná práve týmto smerom.

Požiadavky na aplikáciu:

1. modulárnosť
2. stabilita
3. jednoduchosť a komplexnosť
4. automatické synchronizovanie
5. jednoduché pridanie nového zdroja dát
6. signalizácia poruchových stavov

4.3. Návrh aplikácie

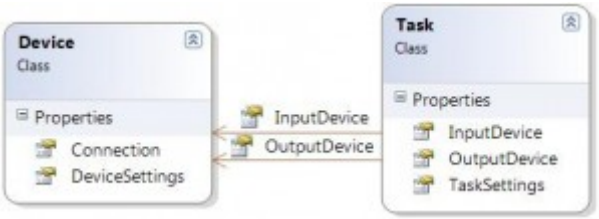
Aplikácia musí v prvom rade definovať odkiaľ a kam má dáta synchronizovať. V prípade dochádzkového systému sú to dochádzkové terminály, rôzne súbory a databázy. Pre iné systémy to môže byť niečo iné. Nazvime si tieto dátové zdroje ako „Zariadenia“ a „Databázy“.

V druhom rade je potrebné, aby aplikácia definovala, kedy, čo a akým smerom sa bude synchronizovať. Takto sa definujú takzvané „Úlohy“, ktoré synchronizácia vykonáva, pričom každá úloha má definované vstupné a výstupné zariadenie. Napríklad pri synchronizácii snímaní z dochádzkového terminálu je vstupom dochádzkový terminál a výstupom je databáza dochádzkového systému.

Základné objekty aplikácie teda sú:

1. zariadenie/databáza (Device)
2. úloha (Task)

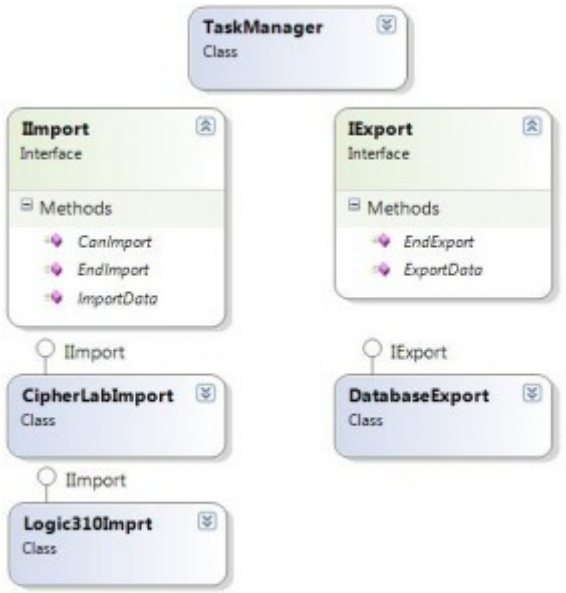
Schematické znázornenie vzťahu úlohy a zariadenia je na obrázku Obr.3.



Obr. 3. Vzťah zariadenia a úlohy

V rámci konfigurácie aplikácie, sa nastavuje kolekcia zariadení, databáz a tiež úloh, ktoré sa pri synchronizácii majú vykonať.

Základným výkonným prvkom aplikácie je synchronizátor. Keďže ide o vykonávanie úloh, synchronizátor sa nazýva „TaskManager“. Postup práce synchronizátora som opísal vyššie v teoretickej časti. Schematické zobrazenie vnútornej štruktúry (synchronizátor a zariadenia, s ktorými pracuje) je na obrázku Obr.4.



Obr. 4. Vnútorná štruktúra aplikácie

„TaskManager“ vykonáva jednotlivé úlohy, pričom na ich vykonanie používa objekty, ktoré implementujú interface pre import a export. Pridanie nového druhu importu alebo exportu preto v praxi znamená len pridanie jedného bloku. Nie je nutné zasahovať do iných častí aplikácie.

4.4. Použitie aplikácie

Navrhnutý spôsob synchronizácie je možný použiť pre rôzne druhy aplikácií. Či už by to bola synchronizácia dát s mobilnými telefónmi alebo prenos dát z meteorologických staníc do databázy. Aplikácia DataReader je navrhnutá na konkrétne použitie pre dochádzkové systémy na synchronizáciu dát z dochádzkových terminálov do databázy.

Automatické spúšťanie aplikácie zabezpečuje tzv. Task Scheduler, ktorý umožňuje variabilné nastavovanie spúšťania aplikácie (denne, mesačne, každú párnu hodinu a iné). Aplikácia môže byť spustená s rôznymi parametrami, ktoré definujú aké konkrétne úlohy sa majú spustiť (aby sa nespúšťali všetky definované úlohy, keďže ich môže byť niekoľko desiatok a viac), ďalej je možné parametrami definovať, ako sa má aplikácia chovať (či má zobrazovať priebeh synchronizácie, či nemá zobrazovať nič iné okrem porúch, alebo či má poruchy posielať napríklad na e-mail).

4.5. Možnosti komunikácie so zariadeniami

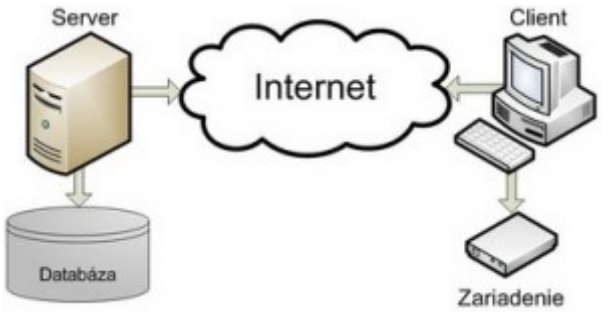
Zariadenia, ktoré vstupujú do procesu synchronizácie používajú rôzne spôsoby komunikácie (sériové rozhranie RS232, ethernet, paralelné rozhranie LPT, USB a iné). V práci som implementoval konkrétne zariadenia, ktoré môžu byť pripojené cez

sériové rozhranie alebo cez ethernet. Spôsob komunikácie so zariadením je záležitosťou len importéra a exportéra a synchronizátor o spôsobe komunikácie so zariadením nič nevie. Túto vlastnosť som neskôr využil na spojenie dvoch aplikácií DataReader medzi dvomi počítačmi v rámci internetu.

5. Synchronizácia v internete

5.1. Opis situácie

Doteraz som rozprával najmä o synchronizácii v rámci jedného počítača, prípadne v rámci lokálnej LAN siete. Predstavme si však situáciu, keď potrebujeme synchronizovať dáta medzi zariadením a databázou, avšak nie na lokálnom intranet, ale v internete. Znázornenie tejto situácie je na obrázku Obr.5.



Obr. 5. Bloková schéma Client-Server

V princípe je synchronizáciu možné riešiť tak, že sa na strane servera otvorí port na prístup k databáze a klient tak môže synchronizovať dáta rovnako, ako to robí na LAN sieti. Otvorenie portu k databáze na strane servera však nie je vždy vhodné, keďže týmto krokom umožníme prístup k citlivým dátam pre hackerov. Z toho dôvodu je oveľa lepšie, ak je server zabezpečený a port je otvorený na strane klienta. Klienta môžeme skôr obetovať pri prípadnom útoku. A nakoniec, ak sme nútený otvárať komunikačný port na serveri, tento port nikdy neotvárame priamo na databázu.

5.2. Návrh synchronizácie

Problémom pre internetovú synchronizáciu je, že na jednej strane (otázka, či na strane servera alebo klienta je len otázka bezpečnosti a možností) musí byť neustále spustená aplikácia, ktorá počúva volania z internetu.

Pri návrhu synchronizácie cez internet, som chcel zachovať existujúci model synchronizácie. Dôležité teda bolo, aby sa nemenil komunikačný interface pre importéra a exportéra.

Návrh si v konečnom dôsledku vyžadoval vytvorenie troch nových komunikačných blokov:

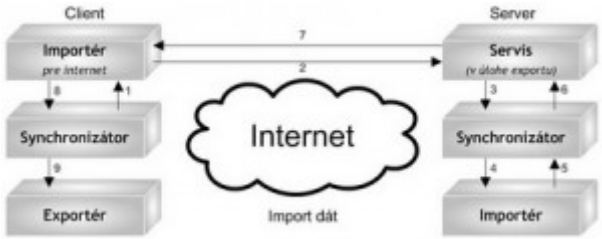
1. Exportér pre internet – blok exportéra, ktorý exportuje záznamy cez internet
2. Importér pre internet – blok importéra, ktorý importuje záznamy z internetu
3. Servis – blok, ktorý umožňuje čakať na požiadavku o export/import dát z internetu

Princíp komunikácie pre export dát je na obrázku Obr.6. a a princíp komunikácie pre

import dát je na obrázku Obr.7.

Obr. 6. Export dát

Export dát – synchronizátor na strane klienta pomocou importéra získa dáta, ktoré následne predá na export. Exportér nadviaže komunikáciu so serverom a predá mu dáta. Na strane servera sa inicializoval importér (v skutočnosti servis), ktorý prevzal od klienta dáta a odovzdá ich synchronizátorovi. Úlohou synchronizátora je oživiť exportéra a predať mu dáta na export.



Obr. 7. Import dát

Import dát – synchronizátor na strane klienta inicializuje importéra. Importér nadviaže komunikáciu so serverom, pričom mu odovzdá informácie o požadovaných dátach. Servis na strane servera pomocou synchronizátora oživí exportéra, pomocou ktorého získa dáta a následne ich pomocou servisu odovzdá klientovi. Na strane klienta dáta príjme importér a odovzdá ich synchronizátoru, ktorý ich exportuje.

Ako môžete vidieť na uvedených obrázkoch, na strane klienta aj na strane servera nebola porušená základná koncepcia Exportér – Synchronizátor – Importér. Len na strane servera nastala jedna zmena, a to, že proces synchronizácie nezačína synchronizátor, ale servis. Servis je totiž na strane servera ten, kto čaká na požiadavky od klienta a na základe týchto požiadaviek riadi synchronizačný proces, pričom pre synchronizátor vystupuje servis raz ako exportér a inokedy ako importér.

5.3. Komunikačná technológia

V momente, keď som si pre realizáciu projektu vybral technológiu od spoločnosti Microsoft, zúžil som výber možných technológií, ktoré môžem na komunikáciu použiť. Aplikácia je postavená na platforme .NET, ktorá je jednou z najprogresívnejších technológií na trhu a spoločne s jazykom C# konkuruje technológiám ako Java, Delphi či FoxPro, ktoré sú na trhu oveľa dlhšie. Pre platformu .NET sa ponúka hneď niekoľko možností, ktoré je možné použiť na komunikáciu v internete. Sú to (v zátvorke sú uvedená najvýznamnejšie výhody konkrétnych technológií):

1. ASP.NET Web Services (interoperabilita)
2. Web Services Enhancements (WSE)
3. .NET Remoting (výkonnosť)

4. Enterprise Services (distribuované transakcie)
5. MS Message Queuing (MSMQ) (spoľahlivosť)

Z týchto technológií má každá svoje východy aj nevýhody a v prípade ich použitia sa črtá otázka, či nepoužiť kombináciu niektorých dvoch, čo je často výhodné.

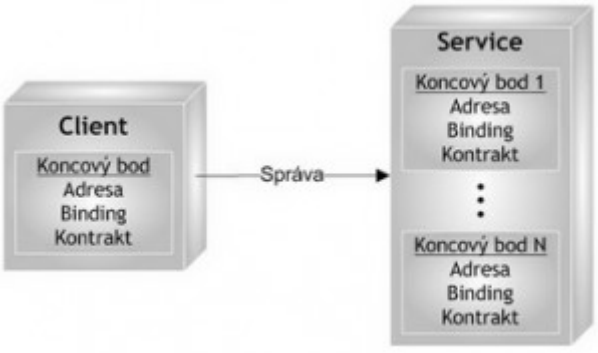
Na trh však prišla najnovšia technológia, ktorá zjednocuje všetky vyššie vymenované technológie do jedného celku. Touto technológiou je Windows Communication Foundation (WCF).

WCF je kompaktný framework na vytváranie servisne orientovaných aplikácií (SOA). Výhodou WCF je, že priamo v sebe obsahuje viacero mechanizmov na ochranu prenášaných dát a správu prístupu.

5.4. Windows Communication Foundation

WCF je založené na komunikácií pomocou správ. Správa (message) je skupina dát obsahujúca záhlavie a telo správy. Príkladom správy je napríklad HTTP požiadavka alebo MSMQ správa. Model WCF rozlišuje medzi klientmi a službami. Klient je aplikácia ktorá iniciuje komunikáciu, služba je aplikácia ktorá čaká na požiadavky klientov. Keďže WCF je implementáciou servisne orientovanej architektúry, základným prvkom je služba.

Službu chápeme ako systém s ktorým komunikujeme pomocou koncových bodov. WCF je teda pre okolitý svet chápaná, ako množina koncových bodov. Koncový bod slúži na prijímanie správ a odosielanie odpovedí. Služba môže mať jeden alebo viac koncových bodov. Schematické znázornenie komunikácie klient-služba je na obrázku Obr.8.



Obr. 8. Princíp komunikácie WCF

Koncový bod definujú tri informácie, a to: adresa, binding a kontrakt.

Adresa koncového bodu špecifikuje, kam sú dáta v internete posielané. Príklad adresy môže byť http://www.mojserver.sk.

Binding definuje mechanizmus, ktorý popisuje, ako majú byť správy posielané. Binding má viacero častí, no medzi základné patrí spôsob prenosu (HTTP, HTTPS, TCP, Named Pipes) a kódovanie (text, binárne). Dôležitou súčasťou je aj bezpečnosť, kde je možné nastaviť kryptovanie správ. Binding tiež zahŕňa session a podporu transakcií.

Kontrakt definuje, aký typy správ môžu byť poslané a prijaté. V princípe ide o definíciu

štruktúry dát, ktoré sú medzi klientom a službou posielané.

WCF ponúka široké možnosti využitia, je ľahko implementovateľný, je plne podporovaný a robustný, preto som ho použil.

5.5. Implementácia WCF

Z vyššie uvedenej teórie vyplýva, že servis, o ktorom som rozprával v návrhu synchronizácie, nie je nič iné ako WCF služba, ktorá na niekoľkých koncových bodoch čaká na volania z internetu. V celej aplikácii je teda len servis niečím novým, čo pre lokálnu sieť nie je potrebné. Importér a exportér, ktoré vedia s týmto servisom komunikovať sú vo svojej podstate len ďalšími modulmi s rovnakou štruktúrou ako ostatný importéri a exportéri. Vďaka tomu nemusí synchronizátor vôbec vedieť, že dáta, ktoré pre synchronizáciu dostáva nepochádzajú z lokálnej siete, ale z internetu.

6. Zhodnotenie

V práci som sa venoval návrhu synchronizačného procesu, ktorého hlavnou myšlienkou bola univerzálnosť a základnými požiadavkami bol bezpečný prenos dát bez strát.

Synchronizačný proces som navrhol tak, aby bol použiteľný na všetky procesy synchronizácie bez ohľadu na štruktúru dát a bez ohľadu na spôsob pripojenia sa k týmto dátam. V teoretickej časti som identifikoval možné typy dátových zdrojov z hľadiska prístupu k dátam a možností synchronizácie. Synchronizačný proces som navrhol tak, aby bola operácia pridania nového zariadenia čo najjednoduchšia.

Proces synchronizácie je založený na vzájomnej komunikácii blokov Importér a Exportér, ktoré sú riadiacim procesom vhodne riadené. Importované dáta sú podpísané identifikačným kľúčom, ktorý sa použije na identifikáciu dát na strane importéra aj na strane exportéra. Bezpečnosť dát z hľadiska straty je zabezpečená samotným postupom synchronizácie. O bezpečnosť dát z hľadiska odcudzenia treťou stranou sa stará bezpečný prenos údajov využitím komunikačnej technológie Windows Communication Foundation.

Pre simuláciu navrhnutého procesu synchronizácie som vytvoril aplikáciu, ktorá synchronizuje údaje pre dochádzkový systém. Rýchlosť synchronizačného procesu je následne najviac závislá od rýchlosti komunikačného kanála.

7. Literatúra

1. John Sharp, “Visual C# 2005 Step by Step”, 2005 Microsoft Corporation, http://msdn2.microsoft.com/en-us/library
2. http://msdn2.microsoft.com/en-us/library
3. Microsoft Corporation, “Introduction to the Microsoft Sync Framework Runtime” November 2007, http://msdn2.microsoft.com/en-us/sync/bb821992.aspx
4. Microsoft Corporation, “Windows Comunication Foundation”, http://msdn2.microsoft.com/en-us/netframework/aa663324.aspx

Projekt bol vyvíjaný v spolupráci so spoločnosťou Abiset s.r.o.

Abiset s.r.o.
Nám. SNP 2/1209
901 01 Malacky
Slovakia

tel.: +421 34 773 4163
fax: +421 34 773 4164
abiset@abiset.com, http://www.abiset.com