1 Současný stav řešené problematiky

Zájem o síťové aplikace a jejich programování v poslední době vzrůstá spolu s exponenciálním nárůstem uživatelů Internetu. Internet se osvědčil jako platforma pro šíření informací, elektronický obchod a zábavu. Růst popularity Javy jako programovacího jazyka, který podporuje přenos kódu po síti, je další hnací silou tohoto vývoje. Použití internetových technologií v privátních sítích přináší vyšší požadavky na síťové aplikace. Jako odezva na tento trend se neustále vyvíjí nové techniky, jazyky a přístupy usnadňující tvorbu aplikací. Asi nejslibnější mezi novými formami je použití *mobilních agentů*.

Jedná se o moderní trend realizace distribuovaných aplikací, který je prozatím nejobecnějším modelem komunikace mezi uzly distribuovaného systému. Z historického hlediska jsou mobilní agenti následníkem systémů zasílání zpráv, vzdáleného volání procedur [Tay1990], vzdáleného vyhodnocení [Sta1990], vzdáleného výpočtu a aktivních zpráv [Vit1981, Ban1986]. Mobilní agenti přinášejí nové možnosti při návrhu aplikací pro elektronický obchod [Ibm1997], dolování dat [The1999], monitorování počítačové sítě [Bau1997a] nebo automatické instalaci softwaru. Možnosti využití mobilních agentů pro konkrétní aplikace jsou v přehledových studiích [Har1995, Lan1998].

Pojem *agent* je těžké přesně definovat, neboť se používá jako zastřešující pojem pro různé druhy výzkumu. Nejobecnější definice [Fra1996] popisuje softwarového agenta pomocí jeho charakteristik, jakými jsou samostatnost [Sho1994, Woo1995], sociální chování, reaktivita [Bro1986, Agr1987], vlastní aktivita [Cha1994], komunikativnost [Mae1994], mobilita a schopnost učit se.

Několik akademických a komerčních skupin v současné době zkoumá a vytváří systémy mobilních agentů. Tyto systémy se dají rozdělit podle množství podporovaných funkcí a podle použitého programovacího jazyka.

Hlavním znakem mobilních agentů je jejich schopnost migrovat z uzlu na uzel. Proto je také podpora mobility agentů základním požadavkem na infrastrukturu systému. Při migraci agent specifikuje buď *absolutní* odkaz na cíl (tj. agentem je specifikováno celé jméno uzlu) nebo *relativní* odkaz (agentem specifikované jméno musí být doplněno na kompletní). Většina systémů podporuje absolutní odkazy. Systémy *Telescript*, *Tacoma* a *Ajanta* podporují relativní odkazy.

Systémy musí také poskytovat mechanismus pro nalezení současného umístění všech entit. Tento proces se nazývá *rozpoznání jména*. Jméno entity může být závislé na jejím umístění, což dovoluje jednoduchou implementaci mechanismu rozpoznávání. Systémy *Agent Tcl*, *Aglets* a *Tacoma* používají jména založená na jméně uzlu a čísle portu. Pro rozpoznání jména uzlu používají systém DNS [Moc1987]. Druhou možností je vytvoření jmen entit, která jsou nezávislá na jejich umístění. Toho lze docílit buď poskytováním lokálních *zástupců* pro vzdálené entity (*Voyager*), nebo zavést globální pojmenování entit nezávislé na jejich umístění (*Ajanta*).

Mobilní agenti, kteří mohou migrovat počítačovou sítí, způsobují bezpečnostní problémy. Systémy mobilních agentů musí poskytovat bezpečnostní mechanismy pro detekci a odstranění napadení. Ty zahrnují *přístupový mechanismus* (pro ochranu soukromého kódu a dat), *ověřovací mechanismus* (ověření identit komunikujících stran) a *autorizační mechanismus* (poskytuje agentům kontrolovaný přístup ke zdrojům na uzlu).

Protože se agenti mohou pohybovat v heterogenním prostředí, přenositelnost jejich kódu je hlavním požadavkem. Z toho důvodu je většina systémů mobilních agentů založena na interpretovaných jazycích [Tho1997], které poskytují přenositelné *virtuální stroje*. Dalším důležitým kritériem je bezpečnost programového kódu. Jazyky podporující typovou kontrolu, zapouzdření a výhradní přístup do paměti jsou vhodné pro implementaci bezpečného prostředí.

Některé systémy používají skriptovací jazyky jako *Tcl*, *Python* nebo *Perl* pro implementaci agentů. Jiné systémy používají objektové jazyky jako *Java*, *Telescript* nebo *Obliq* [Tho1997]. Objevily se také pokusy o standardizaci rozhraní pro mobilní agenty. Konsorcium OMG (*Object Management Group*) vydalo specifikaci *MASIF* (*Mobile Agent System Interoperability Facility*), která se však zabývá správou a ovládáním agentů, přenosem agentů mezi uzly a pojmenováním agentů. Tato specifikace je omezena pouze na prostředí *CORBA*.

Historický vývoj: *Telescript*, vyvinutý firmou General Magic, byl prvním systémem určeným primárně pro programování mobilních agentů. Třebaže byl komerčně neúspěšný a v dnešní době již není k dispozici, drží si historické prvenství mezi jazyky s podporou mobilních agentů. Za ním následovaly systémy *Tacoma* a *Agent Tcl*, ve kterých bylo tělo agenta zapisováno pomocí skriptu. Vývoj programovacího jazyka *Java* a programovacího prostředí s podporou mobilního kódu [Gos1996, Lin1996] vedl ke zrychlení výzkumu v této oblasti. *Aglets*, *Odyssey*, *Voyager* a *Concordia* jsou příkladem systémů mobilních agentů založených na jazyce Java.

Shrnutí současného stavu vede k následujícím závěrům:

1. Existuje celá řada různých systémů mobilních agentů, ale většina z nich implementuje pouze některé vlastnosti mobilních agentů.

2. Existující systémy nebyly navrženy jako bezpečné pro prostředí Internetu.

3. Neexistuje standard pro mobilní agenty.

Literatura: *Ajanta* [Tri1998, Kar1998a], *Agent Tcl* [Gra1996, Kot1997], *Aglets* [Ibm1998, Kar1997], *Concordia* [Mit1997], *Odyssey* [Gen1997], *Tacoma* [Joh1995], *Telescript* [Whi1995], *Voyager* [Obj1997]

2 Cíle práce

Tato práce byla motivována použitím mobilních agentů pro sběr dat ze sítě. Primárním cílem práce bylo vytvoření systému, který bude schopen shromažďovat a třídit data. Původním záměrem byl sběr dat v prostředí Jednotného identifikačního systému (JIS) na ZČU. Na základě zhodnocení současného stavu řešené problematiky byly formulovány následující požadavky na systém:

Bezpečné výpočetní prostředí: Uzel systému i agent potřebují bezpečné výpočetní prostředí. Z hlediska agenta je potřeba zajistit jeho integritu během migrace mezi uzly. Stejně tak musí systém ověřovat příchozí požadavky pomocí metod ověřování identity.

Spolupráce agentů: Agenti během svého života potřebují komunikovat se svým okolím ať už se jedná o agenty nebo jiné entity v systému. Pro podporu spolupráce je potřeba zajistit vhodné pojmenování a vyhledávání entit v systému (*jmenné služby*) a vytvořit komunikační infrastrukturu.

Podpora migrace agentů: Migrace bude založena na metodě přenosu neaktivního agenta. Agent se sám označí, že chce migrovat, a po přechodu do neaktivního stavu je migrace provedena.

Podpora multithreadingu: Agent může být prováděn ve více vláknech.

Využívání dostupných standardů: Pro dosažení přenositelnosti agentů, bezproblémové spolupráce mezi uzly systému a možnosti širokého výběru implementačních možností musí být systém založený na dostupném standardu.

Snadná konfigurace: Je stále více obtížné spravovat existující softwarové vybavení a přizpůsobovat ho potřebám konkrétní aplikace. V případě systémů mobilních agentů se tento problém týká nejen agentů, ale také systému samotného.

Rozšiřitelnost: Jedním z nejdůležitějších požadavků na systém agentů je jeho rozšiřitelnost. Návrh systému musí počítat s nároky uživatelů na jeho budoucí rozšiřování.

V neposlední řadě je potřeba umožnit odložení nevyužitých agentů z paměti a jejich opětnou aktivaci po příchodu požadavku na ně. Tím se odstraní blokace prostředků na uzlu.

Jednoduché použití: Uživatelé nebudou využívat rozsáhlé rozhraní systému (API) v případě, že budou chtít vytvořit agenty. Navíc nesmí být omezováni jednoduchým a přímočarým modelem programování, protože pak by nemohli využívat všech možností daného jazyka.

3 Popis řešení

`SMAS` (*Simple Mobile Agent System*) je systém mobilních agentů, jehož realizace je zapsána v jazyce Java. Jedná se o vícevrstvý systém, což umožňuje jasně definovat bezpečnostní politiku.

3.1 Složení systému

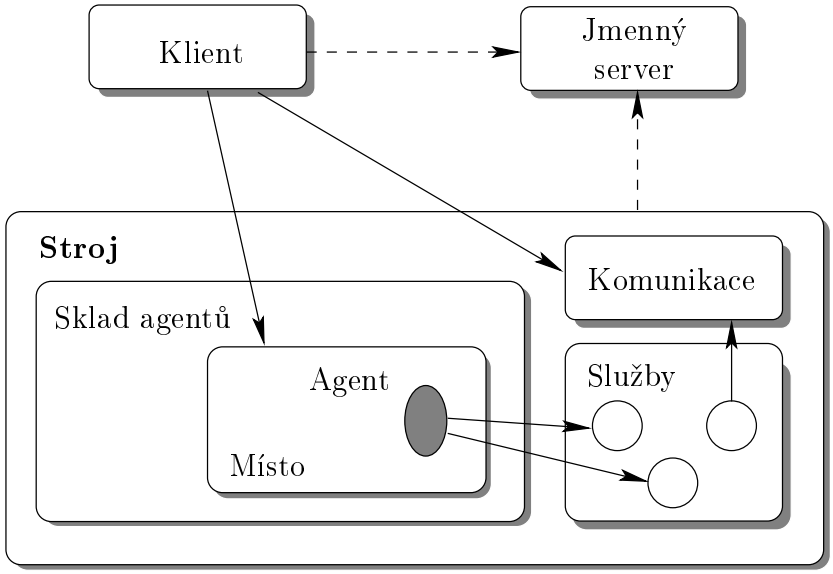
Systém obsahuje tři druhy základních součástí (klient, jmenný server a stroj), které jsou typicky (není to zcela nutné) realizovány oddělenými programy a mohou být umístěny na různých uzlech.

- **Klient** je program, který vytvoří agenta a vyšle jej. Během činnosti agenta s ním může komunikovat a po ukončení se agent opět navrátí na klienta.
- **Jmenný server** udržuje databázi strojů a agentů, u každého z nich pak momentální lokaci ve formě *JNDI URL*.
- **Stroj** (*engine*) je základní součástí systému, který realizuje funkci hostitele. Stroj realizuje služby pro agenty a také komunikační subsystém, ke kterému mají agenti přístup (prostřednictvím služby).

  Stroj obsahuje jeden nebo více **skladů** (*agent repository*). V těchto skladech se nacházejí **místa** (*place*), v každém místě jeden agent (nebo vyjádřeno obráceně: každý agent má své místo).
- **Sklad** (*agent repository*) je kontejner, do něhož se vkládají agenti (ne přímo, ale se svým místem). Stroj může obsahovat jeden nebo více skladů. Jeden sklad na hostiteli má výsadní postavení: je **implicitní** (*default*), tj. všichni příchozí agenti se implicitně umísťují do něj. Sklad zajišťuje vkládání a rušení agentů, vytváření a rušení míst, migraci, pasivaci a aktivaci agentů. Udržuje také množinu nabízených služeb.

  Jednotlivé sklady se liší množinou nabízených služeb a bezpečnostní politikou. Smyslem výše uvedené architektury je umožnit oddělení různých aplikací. Implicitní sklad slouží pro všechny agenty bez rozdílu účelu a nabízí jen omezenou množinu služeb a základní bezpečnostní pravidla.
- **Místo** (*place*) zprostředkovává styk mezi strojem a agentem. Směrem k agentovi reprezentuje stroj (umožňuje přidělování služeb), směrem „ven“ ke stroji reprezentuje agenta (uchovává informace o něm a umožňuje komunikaci). Jednotlivé sklady se liší tím, jakým způsobem (podle jaké politiky) přidělují práva agentům.

Agent sám je uzavřen v tzv. **kontextu**. Tento kontext je to jediné, s čím může agent komunikovat přímo (tj. voláním metod). Veškerý styk s okolím je prováděn pomocí zástupců. Kontext sám poskytuje pouze přístup k službám.



Obrázek 1: Přehled částí systému

3.2 Identifikace a adresování

V systému `SMAS` jsou výrazně rozlišena jména a adresy. Obojí má jiný účel a jiné vlastnosti. Převod jména na adresu je prováděn pomocí *jmenného serveru*. Pro opačný převod je nutné navázat s entitou komunikaci a ta sdělí své jméno. Tímto způsobem se zároveň ověří, zda adresy označují stejnou entitu.

- **Jméno** se používá pro jednoznačnou identifikaci entit a jeho vazba tedy musí být 1:1. Všechna jména používaná v `SMAS`u mají pro člověka srozumitelnou formu, tedy řetězec (String).
- **Adresa** umožňuje nalézt entitu a navázat s ní komunikaci. Vazba je většinou 1:N, protože entitu lze adresovat několika způsoby (několika protokoly, různé formulace apod.). Jednoduchá adresa je URL, tedy řetězec. Entita však často má těchto adres víc a sdružují se do tzv. **kompletní adresy**.

3.3 Agent

Při tvorbě agenta je nutné vzít v úvahu především dvě okolnosti:

- Agent bude prováděn ve svém *kontextu* a musí tedy být schopen s ním správně komunikovat. Pro tento účel je použit standardní balík *Java beans*, zvláště pak koncepce *kontextu* (do kterého se *bean* vkládá). Celý agent se navenek jeví jako jeden *bean*.

- Při různých příležitostech je nutné zjistit kompletní stav agenta. K tomu slouží **serializace**. Agent musí umožnit svoji serializaci.

Vnitřní struktura agenta není nijak specifikována, záleží jen na rozhodnutí programátora. Obecně platí, že se skládá z libovolného množství objektů (v krajním případě jen jednoho). Je sice vhodné agenta také složit z *beanů*, ale není to nijak nutné. Pro své okolí je agent reprezentován jediným objektem nazývaným **hlavní objekt** (*main object*).

3.3.1 Části uchované odděleně

O agentovi je nutné znát a uchovávat některé standardní údaje, které ho charakterizují (jako je například jméno nebo identita vlastníka). Tyto údaje samozřejmě mohou být interní součástí agenta, který je pak na požádání sděluje. Tento přístup je však málo bezpečný, neboť umožňuje agentovi měnit svoji identitu a své vlastnosti. Z těchto důvodů jsou ve `SMASu` tyto informace uchovávány odděleně.

- **Metadata** vytváří klient a obsahují základní údaje o agentovi: vlastníka, tvůrce (klienta) a umístění kódu. Všechny tyto údaje jsou pak klientem podepsány a pomocí tohoto podpisu pak může být ověřena jejich pravost.

- **Jméno a adresa** jsou vygenerovány strojem při vyslání agenta a nemohou být tedy součástí metadat.

- **Programový kód** může být uložen ve formě jednotlivých class souborů nebo ve formě jar archívu a adresuje se pomocí *URL* některým ze standardních protokolů (*HTTP, FTP*). Informace o veškerém programovém kódu je v metadatech a nazývá se *codebase*.

  Každý agent má svůj vlastní *classloader*. V Javě to znamená, že má také vlastní instanci tohoto kódu. Nemůže tak dojít ke konfliktu verzí ani k úmyslnému napadení pomocí upraveného kódu.

3.3.2 Identita vlastníka a tvůrce

Pro autorizaci akcí agenta (především jeho přístupu ke službám) se jen výjimečně použije identita jeho samého. Téměř výhradně se používají identity **vlastníka** (*owner*) a **tvůrce** (*creator*, většinou klient) uložené v *metadatech*. Pro uložení těchto identit se používá třída Subject obsahující dva typy údajů: *principals* a *credentials*.

- **Principals** je objekt, který obsahuje jméno entity. Protože entita může vystupovat v mnoha rolích, mívá několik jmen v různých jmenných prostorech. Je definováno několik typů, které musí stroj standardně rozeznávat: jednoduché jméno, přihlašovací jméno (login), reálné jméno a X.509 certifikát.

- **Credentials** obsahuje informaci o entitě, kterou se tato entita může prokazovat a používat ji k autorizaci. *Credentials* může být jakýkoliv objekt, jehož kód je k dispozici na jakémkoliv stroji. `SMAS` definuje jediný používaný typ *credentials*, který má však zásadní význam. Jsou to **certifikáty**. Jediný implementovaný a podporovaný druh certifikátů jsou certifikáty podle normy X.509. Tyto certifikáty obsahují identitu vlastníka a jsou podepsané certifikační autoritou, takže lze ověřit jejich pravost.

3.3.3 Autorizace agenta

Jazyk Java obsahuje sám o sobě poměrně propracovanou bezpečnostní architekturu založenou na **právech** (*permissions*). Ta je však navržena pouze pro případ mobilního kódu (tzv. appletů) a proto autorizace probíhá jen na úrovni kódu. Doplňkový balík *JAAS* (*Java Authentification and Authorization Service*) sice tuto architekturu rozšiřuje o autorizaci na základě identity, ovšem jen na úrovni vláken. Pro autorizaci přístupu ke službám je celá bezpečnostní architektura Javy obtížně použitelná a proto `SMAS` zavádí tzv. **omezení** (*constraints*). Obě tyto architektury se navzájem doplňují.

- **Práva** jsou v konfiguraci přiřazena úplně všem, tj. nejsou omezena na umístění ani na určitý podpis. Jedná se o statické přiřazení.

- **Omezení** přiděluje sklad při vkládání agenta. Mohou být statická (implementace pak využívá standardní třídy) nebo i dynamická (o konkrétní hodnotě se rozhodne až při vyvolání zadané metody). Předem je definováno jediné konkrétní omezení – maximální počet vláken agenta.

3.3.4 Autentifikace při komunikaci

Pro autentifikaci se používá algoritmus s automatickou výměnou veřejných klíčů (certifikátů). Jedná se o algoritmus **výzva-odpověď** (*challenge-response*) pro authetifikaci při vzdáleném volání metody, využívající výměnu a podepisování náhodně generovaných čísel.

3.3.5 Manipulace s agentem

Systém `SMAS` podporuje všechny přechody mezi stavy agenta a tedy i všechny operace s ním. Jedná se o operace vložení, vyjmutí, vyslání, ukončení, migrace a návrat.

3.4 Komunikace

Komunikační systém je oddělenou programovou jednotkou, protože může být součástí jak stoje, tak klienta (případně ještě i dalších programů). Komunikace je používána nejen ke komunikaci se samotnými agenty, ale především pro jejich přenos. Pro veškerou síťovou komunikaci mezi prvky systému je používáno **vzdálené volání metod** (*Remote Method Invocation – RMI*). Největší výhodou této formy komunikace je její standardizace (protokoly jsou široce používány) a především jednoduchost použití. Samotný přenos přes síť je do velké míry transparentní a chová se jako lokální vyvolání metody. V komunikačním subsystému `SMAS`u je tato transparentnost ještě zesílena do té míry, že za určitých okolností může komunikace probíhat dokonce lokálně.

Důležitým požadavkem zohledněným při návrhu komunikace je možnost použití různých komunikačních protokolů. Jazyk Java obsahuje komunikační rozhraní *Java RMI* (nebo jen *RMI*), které je velice efektivní a jednoduché na použití. Je však svázáno pouze s tímto jediným programovacím jazykem a z tohoto důvodu se častěji používá systém *CORBA*. Jeho použití je však výrazně komplikovanější. Definování dalších komunikačních protokolů je dobrovolné, jediný povinný protokol je právě *RMI*. Přidat komunikační subsystém například pro systém *CORBA* by neměl být závažný problém.

3.4.1 Brány

RMI pracuje na úrovni objektů a požadovanou komunikační entitou je agent nebo stroj (popř. i klient, ale komunikace s klientem není nijak definována). Tato entita musí být reprezentována vzdáleně přístupným objektem, který se nazývá **brána** (*gate*). Bran může být několik, ale jejich chování musí být zcela shodné. Liší se pouze komunikačním protokolem Pokud je jediným podporovaným protokolem *RMI*, existuje jen jedna brána.

3.4.2 Jmenné servery

Jmenné servery slouží k nalezení agentů v systému. Jako jmenný server lze použít jakoukoliv adresářovou službu (*directory service*), která je podporována *JNDI* a do které lze ukládat reference. Ve standardní distribuci *JNDI* je jedinou takovou službou *LDAP*, která je také použita.

Při použití jednoho jmenného serveru by byla komunikace příliš závislá na jeho stabilitě, a proto je vhodné použít více jmenných serverů. Adresa jmenného serveru se pak nazývá **kompletní adresa** a její složky (*URL*) jsou **jednoduché adresy** jednotlivých jmenných serverů.

3.4.3 Zástupci

Pro udržení komunikace při migraci agenta je používán mechanismus **lokálních zástupců** (*local proxy*). Používá se při veškeré komunikaci, nejen při komunikaci s agentem (zástupce umožňuje ošetřit jakoukoliv dočasnou nedostupnost druhé strany, nejenom migraci agenta). Komunikační zástupce může být ve dvou stavech: **připojený** (*connected*) a **rozpojený** (*disconnected*). Pokud je *připojený*, obsahuje přímo referenci na cílový objekt. V tomto stavu je zcela transparentní a je možné normálně komunikovat. Pokud je *rozpojený*, není komunikace navázána. Tímto způsobem je zajištěno udržení spojení při migraci agenta. Jakmile agent „odmigruje“ na jiný stroj, další volání selže (brána již neexistuje). Zástupce se však pokusí obnovit spojení, a protože na jmenném serveru je již uložena nová adresa agenta, naváže se spojení s novou branou. Volání metody se potom úspěšně provede na této bráně.

3.4.4 Lokální komunikace

Pokud se agenti vyskytují na stejném stroji, jako výsledek navázání spojení se vrátí **lokální brána** (*local gate*). Při volání metod se pak pouze volají metody této lokální brány a komunikace probíhá čistě na úrovni virtuálního stroje.

3.4.5 Komunikace s agentem

Komunikační brány agenta vytváří a obsluhuje nikoliv agent, ale jeho místo. Aby bylo možné komunikovat přímo s agentem, musí si agent nechat zpřístupnit svůj vlastní objekt, se kterým půjde komunikovat. Tento proces se označuje termínem **zveřejnit** (*publish*). Zveřejněný objekt je nazýván *published interface*.

3.5 Služby

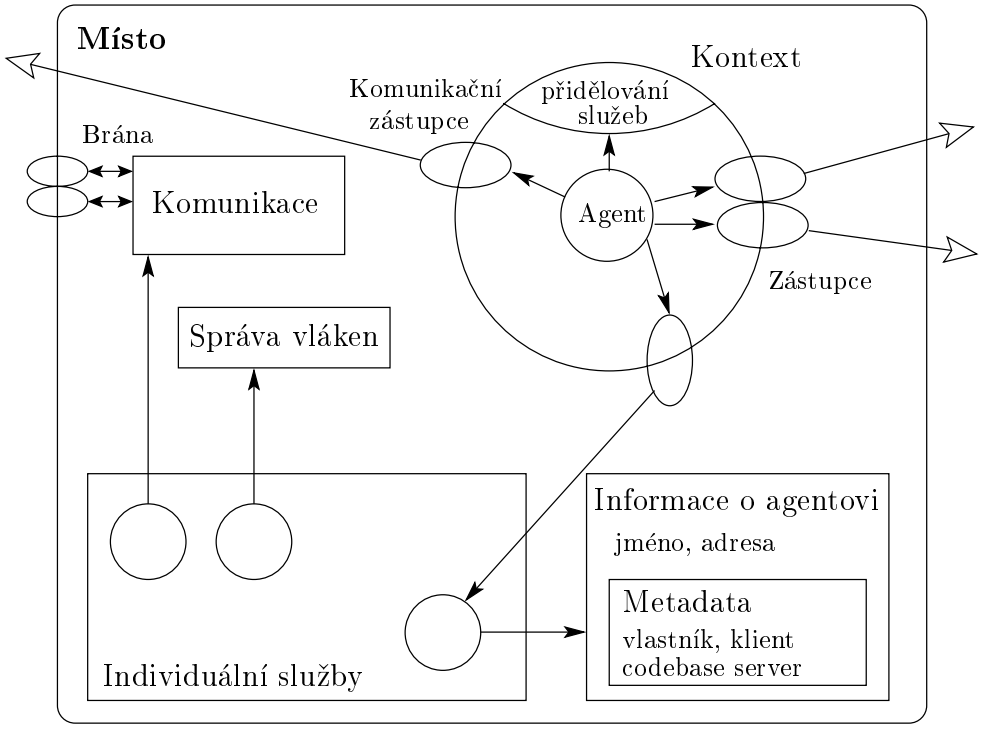
Pro přidělování služeb se využívá koncepce *bean kontextů*.Každá služba je identifikována dvěma parametry: třídou (*class*) a *selectorem*. Třída určuje typ služby, konkrétně pak přímo jaký interface tato služba implementuje. *Selector* dovoluje výběr z více instancí. U mnoha služeb existuje jen jedna instance a *selector* nemá smysl – v takovém případě bývá `null`.

Množina nabízených služeb se během agentova života mění, a to nejen v případech, kdy se služba skutečně stává nedostupnou a opět dostupnou. Nejčastější příčinou této změny je případ, kdy je agent ve stroji spouštěn (vyslání, migrace, aktivace) nebo ukončován (migrace, pasivace, ukončení). V okamžiku vložení do kontextu nejsou k dispozici žádné služby. Ty jsou až poté postupně přidávány. Vyjmutí pak probíhá opačně – služby jsou nejprve všechny zrušeny a teprve potom je agent vyjmut.

O těchto změnách může být agent informován. Oznámení probíhá prostřednictvím tzv. *listenerů*. *Listener* je objekt, který implementuje patřičný interface a zaregistruje se u zdroje události. Když je událost vygenerována, zavolá se jeho patřičná metoda. Agent je od služeb oddělen pomocí **zástupce** (*proxy*), přičemž ve `SMAS`u se využívá standardního mechanismu z *JDK*. Významnou vlastností zástupce je, že může být **odpojen** (*disconnect*). Touto operací se zruší odkaz na službu (nastaví se na `null`) a zástupce je od toho okamžiku nefunkční.

3.5.1 Individuální služby

Místo vytváří speciální tzv. *individuální služby*, které jsou poskytnuty jen agentovi v tomto místě uloženém. Individuální služby vesměs nahrazují přímou komunikaci s místem nebo strojem. Jsou vytvořeny místem při jeho vzniku a při zániku místa jsou zničeny. K individuální službě místa může přistupovat pouze a jen agent umístěný v tomto místě; dochází tak automaticky k autentifikaci a autorizaci. Standardní individuální služby, které musí místo poskytovat, jsou: aktuální informace o agentovi (jméno, adresa, jméno stroje, adresu stroje, jméno skladu a zda je v aktivním stavu), přístup k metadatům, přístup ke komunikačním službám, operace nad sebou samým, přístup ke službám stroje a přiřazování vláken.



Obrázek 2: Struktura místa

4 Zhodnocení

Použití mobilních agentů pro vývoj a vytváření distribuovaných aplikací přináší výhody, které z nich dělají atraktivní alternativu k tradičnímu pojetí komunikace v modelu klient-server. Jedná se zejména o větší autonomii při jejich vykonávání, snížení množství předávaných dat, dynamická aktualizace služeb, atd. Mnoho nových aplikací, které vznikají spolu s dynamickým vývojem Internetu – jako jsou elektronický obchod, online výuka, získávání informací ze sítě nebo dolování dat (*data mining*) – jsou vhodné pro použití mobilních agentů již ze své podstaty. Stejně tak se dají použít i jako alternativní řešení při vývoji distribuovaných vědeckých výpočtů, kdy se využijí pro distribuci zatížení na jednotlivé uzly sítě. Přesto se zatím mobilní agenti v praxi příliš nepoužívají, neboť nemají dostatečné zabezpečení a také nejsou navzájem kompatibilní. Zavedení mobilních agentů do praxe není možné bez kontroly integrity jejich kódu a dat, spolehlivé autentifikace agentů, ochrany zdrojů na uzlech atd. I když bylo vytvořeno mnoho experimentálních (i komerčních) systémů mobilních agentů, žádný z nich nezohlednil bezpečnost jako hlavní část návrhu. Tato práce se proto zabývala návrhem systému mobilních agentů skládajícího se z několika vrstev, což dovoluje přesně definovat a oddělit jednotlivé úrovně bezpečnostní politiky.

5 Vlastní přínos práce

Byla navržena a implementována infrastruktura systému mobilních agentů, tak jak byla popsána v předešlých kapitolách. Systém podporuje všechny operace s agenty, jejich migraci a poskytuje přístup ke službám na uzlech.

- *Vícevrstvý návrh:* Při návrhu struktury se vycházelo hlavně z bezpečnostních požadavků. Agent je tak naprosto oddělen od okolí svým *místem*. Veškerý styk s okolím je prováděn pomocí *zástupců*. Kromě přístupu k externím službám nabízí místo i speciální *individuální služby*. Tyto služby slouží jen pro zprostředkování komunikace s různými subsystémy místa. Jsou vytvořeny místem při jeho vzniku a při zániku místa jsou zničeny. K individuální službě místa může přistupovat pouze a jen agent umístěný v tomto místě; dochází tak automaticky k autentifikaci a autorizaci.

  Nadřazená vrstva (*sklad*) zajišťuje vkládání a rušení agentů, vytváření a rušení míst, migraci, pasivaci a aktivaci agentů. Udržuje také množinu nabízených služeb. Jednotlivé sklady se liší množinou nabízených služeb a bezpečnostní politikou.

- *Přenos kódu na vyžádání:* Informace o veškerém programovém kódu agenta je uchována odděleně v tzv. *metadatech* a nazývá se *codebase*. Každý agent

má svůj vlastní *classloader* což znamená, že má také vlastní instanci tohoto kódu. Nemůže tak dojít ke konfliktu verzí ani k úmyslnému napadení pomocí upraveného kódu.

- *Vlastník a tvůrce:* Pro autorizaci akcí agenta (především jeho přístupu ke službám) se jen výjimečně použije identita jeho samého. Téměř výhradně se používají identity *vlastníka* a *tvůrce* uložené v *metadatech*. Obě identity obsahují objekty *principals* a *credentials*. Principals obsahuje jméno entity a slouží k ověření identity vlastníka nebo tvůrce. Credentials se používá k autentifikaci pomocí certifikátů.

- *Práva a omezení:* Jazyk Java verze 2 obsahuje sám o sobě poměrně propracovanou bezpečnostní architekturu založenou na *právech*. Ta je však navržena pouze pro případ mobilního kódu (tzv. appletů) a proto autorizace probíhá jen na úrovni kódu. Doplňkový balík *JAAS* sice tuto architekturu rozšiřuje o autorizaci na základě identity, ovšem jen na úrovni vláken.

  Pro autorizaci přístupu ke službám je celá bezpečnostní architektura Javy obtížně použitelná a proto byly zavedeny tzv. *omezení*. Omezení jsou vytvářena během života agenta *skladem*. Obě tyto architektury se navzájem doplňují.

- *Zástupce a komunikační zástupce:* Oddělení agenta od služeb je zajištěno pomocí *zástupců*, přičemž byl použit standardní mechanismus *JDK*.

  Další významnou vlastností zástupce je, že může být *odpojen*. Touto operací se zruší odkaz na službu a zástupce je od toho okamžiku nefunkční. Použití je významné především při uvolnění služby, kdy se tato operace provede a zabrání se tak používání již uvolněného zástupce.

  Pro udržení komunikace při migraci agenta je používán mechanismus *lokálních zástupců*. Používá se však při veškeré komunikaci v systému, nejen při komunikaci s agentem. Komunikační zástupce může být ve dvou stavech: *připojený* a *rozpojený*. Pokud je *připojený*, obsahuje přímo referenci na cílový objekt. V tomto stavu je zcela transparentní a je možné normálně komunikovat. Pokud je *rozpojený*, není komunikace navázána.

- *Komunikace:* Pro komunikaci se používá vzdálené volání metod *RMI*. Tato metoda pracuje na úrovni objektů, zatímco požadovanou komunikační entitou je agent nebo stroj. Tato entita musí být reprezentována vzdáleně přístupným objektem, který se nazývá *brána*.

  Bran může být několik, ale jejich chování musí být zcela shodné. Liší se pouze komunikačním protokolem (pokud je jediným podporovaným protokolem *RMI*, existuje jen jedna brána).

- *Autentifikace při komunikaci:* Při komunikaci je nutné, aby si strany navzájem prokázaly svou identitu (autentifikace). Teprve podle prokázané identity pak mohou rozhodnou, zda se druhé straně dá důvěřovat (autorizace).

  Pro autentifikaci se používá algoritmus s automatickou výměnou veřejných klíčů (certifikátů). Jedná se o algoritmus *výzva-odpověď* pro authetifikaci při vzdáleném volání metody, využívající výměnu a podepisování náhodně generovaných čísel.

S využitím prostředků jazyka Java a standardních rozšiřujících balíků byl navržen a vytvořen robustní a bezpečný systém splňující cíle práce. Jeho funkčnost byla otestována na několika typických aplikacích mobilních agentů.

6 Možná rozšíření návrhu

- Z bezpečnostních důvodů je agent při požadavku *návratu* vrácen pouze v serializované formě tj. bez metadat, jména a adresy. Tyto informace totiž nejsou chráněny žádnou autentifikací a tudíž by je mohl získat kdokoli bez autorizace. Pro některé aplikace např. vyrovnávání zátěže by však bylo lepší, aby se do serializované podoby přidaly i informace o agentovi.

- V návrhu není řešena podpora pro vzdálené monitorování činnosti agentů. Hostitel nemůže vzdáleně získávat informace o stavu agenta, pokud je agent sám neposkytuje. Při návrhu nebyla tato podpora uvažována, neboť by tím vzniklo slabé místo k proniknutí do systému – v tomto případě ze strany nebezpečného hostitele.

- Z hlediska perzistentního uchování agentů by bylo vhodné rozšířit návrh o možnost ukládání objektů do vhodné databáze. Tím by se usnadnila správa celého systému.

- S předchozím bodem úzce souvisí správa verzí agentů a obecně objektů v systému. Tím by vznikla možnost existence několika verzí téhož kódu, což by jistě významnou měrou usnadnilo správu systému.

7 Shrnutí

Tato disertační práce poskytuje přehled typů softwarových agentů. Agenti jsou rozděleni na základě svých vlastností, přičemž důraz je kladen na mobilní agenty. Jedná se o moderní trend realizace distribuovaných aplikací, který je prozatím nejobecnějším modelem komunikace mezi uzly distribuovaného systému. Pro využití mobilních agentů je třeba zajistit podporu ze strany výpočetního prostředí.

Systém mobilních agentů musí mít pevně definovanou strukturu, aby mohl poskytovat služby pro přístup ke svým prostředkům. Při návrhu systému je však třeba vyřešit i otázku bezpečného přístupu k prostředkům. Řešením těchto zdánlivě protichůdných požadavků je systém skládající se z několika vrstev, které jsou navzájem striktně oddělené. Jednotlivé komponenty systému řeší identifikaci a adresování objektů, přístup ke službám na uzlech, autentifikaci a autorizaci objektů a komunikaci mezi objekty.

Hlavním cílem práce byl návrh systému pro podporu mobility agentů v prostředí Internetu. Z toho vyplývá, že bezpečnost systému se v průběhu návrhu stala hlavním požadavkem. Ostatní vlastnosti se systému se jí musely přizpůsobit tj. všechny akce v systému jsou autorizovány. Hlavním bodem práce byl tedy návrh takového systému, jehož programová realizace byla ověřena na typické aplikaci mobilních agentů. V závěru práce je realizace systému otestována a výsledky srovnány se systémem RPC a systémem Aglets.

Abstrakt

Cílem disertační práce je návrh objektově orientovaného systému pro podporu mobility agentů v prostředí Internetu. V úvodu práce je poskytnut přehled typů softwarových agentů. Agenti jsou rozděleni na základě svých vlastností, přičemž důraz je kladen na mobilní agenty. Jedná se o moderní trend realizace distribuovaných aplikací, který je prozatím nejobecnějším modelem komunikace mezi uzly distribuovaného systému. Tento nový přístup přináší nové možnosti při návrhu aplikací pro elektronický obchod, dolování dat nebo monitorování chování sítě. Je zřejmé, že v prostředí Internetu je bezpečnost takového systému klíčovým prvkem. Navržený systém mobilních agentů se skládá z několika vrstev, které jsou navzájem striktně odděleny, což dovoluje přesně definovat úrovně bezpečnostní politiky. Jednotlivé komponenty systému pak řeší identifikaci a adresování objektů, přístup ke službám na uzlech, autentifikaci a autorizaci objektů a komunikaci mezi objekty. Pro bezpečný přístup ke službám systému je použit mechanismus zástupců. Každý mobilní objekt v systému má svou identitu, kterou se prokazuje při všech svých akcích. Identita objektů je založena na certifikátech dle normy X.509. Pro realizaci byl zvolen programovací jazyk Java spolu se standardními rozšiřujícími balíky. V závěru práce je ověřena programová realizace systému na typických aplikacích mobilních agentů a výsledky experimentů jsou srovnány s existujícím systémem Aglets. Z výsledků vyplývá, že navržený systém má poněkud větší vlastní režii než odpovídající systémy, což souvisí s navrženou strukturou, kdy se při migraci vytváří nové objekty nezbytné pro běh systému.

Abstract

Main purpose of the thesis is design of object-oriented mobile agent system in the Internet environment. At first view of the software agent is presented. Agents are classified into classes according to their properties, mainly according to the mobility. The used of mobile agents is a new paradigm for structuring distributed systems. This new paradigm of distributed computation offers great opportunities for electronic commerce, data mining or network management. The key feature of the mobile agent system in the Internet environment is a security. The designed system is composed of a few strictly separated layers. This structure allows define level of secure policy accurately. Components of the system perform object identification and addressing, access to services in the network nodes, object authentification and authorization and communication among objects. A proxy-based access control mechanism is used for secure access to services. Each mobile object in the system has its own identity based on X.509 certificate standard. This identity is used for authentication of all its actions. The Java language with standard extension packages is used for implementation of proposed system. Finally, the implementation is verified by typical applications of mobile agents. Results are compared with existing Java-based mobile agent system-Aglets. Designed system has higher overhead than comparable system because it has more complex structure and it is necessary to create new objects during the migration.

Abstrakt

Die vorgelegte Dissertation beschäftigt sich mit Entwurf und Entwicklung des objekt-orientiertes Unterstützungssoftwaresystem mit dem Ziel, die Mobilität von Softwareagenten in der Internet–Umgebung zu unterstützen. Im ersten Teil der Dissertation wird die Übersicht der möglichen Typen von Agenten angeführt; die Agenten werden auf Basis ihrer Funktionsmerkmale klassifiziert, die größte Aufmerksamkeit wird auf die mobilen Agenten konzentriert. Es geht um den modernen Zugang zur Programmrealisation von verteilten Applikationen, der in dieser Zeit als das allgemeinste Modell der Kommunikation zwischen Knoten des verteilten Systems in Betracht genommen werden kann. Solcher Zugang ist sehr attraktiv beim Entwurf von Applikationen in Bereichen der e-Commerce und e-Business, Data Mining und Monitoring des Verhaltens von verteilten Systemen. Dabei ist es offenkundig, daß die Sicherheit des verteilten Systems im Internet–Umgebung der Schwerpunkt der erfolgreichen und zuverlässigen Applikation ist. Das entwickelte System von mobilen Agenten besteht aus mehreren Applikationsschichten, die untereinander scharf getrennt sind und diese Trennung eine eindeutige Bestimmung von Sicherheitsebenen des entwickelten Systems ermöglicht. Einzelne Systemkomponenten gewährleisten dann die Identifizierung und Adressierung von Objekten, Zugriff zu den von Netzknoten angebotenen Systemdiensten, Authentifizierung und Authorisierung von Objekten und zuverlässige Kommunikation zwischen den Objekten. Um den sicheren Zugang zu den Systemdiensten zu realisieren, wurde der proxy–control–Mechanismus verwendet. Jedes mobile Objekt besitzt seine Identität, mit der es bei jeder Systemaktion eindeutig identifiziert wird. Identität von Objekten wird auf Basis der Zertifikate des Standards X.509 definiert. Die Implementation wurde in der Programmiersprache Java, einschl. Standard–Erweiterungspaketen, realisiert. Anschließend werden die Eigenschaften der Implementation an der typischen Aufgabe der Anwendung von mobilen Agenten demonstriert, Ergebnisse von ausgeführten Experimenten werden mit existierendem System „Aglets“ verglichen. Es folgt aus den Ergebnissen von Experimenten, daß das entwickelte und in der Dissertation präsentierte System weisst ein bißchen höhere Werwaltungskosten als die vergleichbaren existierenden Systeme auf; diese Erhöhung hängt mit der entwickelten Systemstruktur zusammen, da bei der Migration die neuen, für den Systemlauf notwendigen Objekte erzeugt werden sollen.

Publikace autora

[1] J. Ledvina, M. Otta, M. Šimek, T.Q. Trung, V. Vavřička: *Systém pro autorizaci přístupu.*
Sborník celostátní konference technických univerzit a průmyslu TRANSFER 98, 1998

[2] M. Šimek: *Plánování procesů v distribuovaném výpočetním prostředí.*
Proceedings of XXIst International Colloquium ASIS 1999 Advanced Simulation of Systems, ISBN 80-85988-41-0, TU Ostrava 1999, str. 249–254

[3] P. Brada, P. Lederbuch, J. Ledvina, M. Otta, L. Petrlík, V. Skala, M. Šimek: *Počítačová grafika a vizualizace dat v paralelním a distribuovaném prostředí.* Výzkumná zpráva projektu VZ 97 155, Plzeň, 1999

[4] M. Otta, M. Šimek: *Debugging Distributed Computations.*
Proceedings of XXIInd International Colloquium ASIS 2000 Advanced Simulation of Systems, ISBN 80-85988-51-8, TU Ostrava, 2000, str. 213–218

[5] M. Otta, M. Šimek: *SMAS - A Simple Mobile Agent System.*
Proceedings of XXIIIrd International Colloquium ASIS 2001 Advanced Simulation of Systems, ISBN 80-85988-61-5, TU Ostrava, 2001, str. 245–250

[6] M. Šimek: *Mobile Agent System Facilities.*
Proceedings of International Conference on Software, Telecommunications and Computer Networks SoftCOM 2001, ISBN 953-6114-44-5, FEST Split, 2001, str. 183–190

Reference

[Agr1987] P.E. Agre, D. Chapman: *Pengi: An Implementation of a Theory of Activity*. Proceedings of the 6th National Conference on Artificial Intelligence, San Mateo, 1987

[Ban1986] J.S. Banino: *Parallelism and Fault Tolerance in Chorus*. Journal of Systems and Software, 1986

[Bar1996] M. Barbuceanu, M.S. Fox: *The design of a coordination language for multi-agent system*. Inteligent Agent III. Agent Theories, Architectures, and Languages, 1996

[Bau1997a] J. Baumann, N. Radouniklis: *Agent groups for mobile agent systems*. Distributed Applications and Interoperable Systems, Chapman & Hall, 1997

[Bau1997b] J. Baumann, N. Radouniklis, K. Rothermel: *Communication Concepts of Mobile Agent System*. Proceedings of the 1st International Workshop, Mobile Agent 97, Springer, 1997

[Bra98] T. Bray, J. Paoli, C. M. Sperberg: *Extensible Markup Language*. W3C, February 1998
`http://www.w3.org/TR/1998/REC-xml19980210`

[Bro1986] R.A. Brooks: *A Robust Layered Control System for a Mobile Robot*. IEEE Journal of Robotics and Automation 2(1), 1986

[Bro1991] R.A. Brooks: *Inteligence without Representation*. Artificial Inteligence 47, 1991

[Cab1998] L. Cable: *Extensible Runtime Containment and Services Protocol for JavaBeans*. Sun Microsystems, Inc., 1998

[Cmu2001] CMU: *A CMU Common Lisp Documentation Collection*.
`http://www.cons.org/cmucl`, 2001

[Dav1995] N.J. Davies, R. Weeks: *Jasper: Communicating Information Agents*. Proceedings of the 4th International Conference on the World Wide Web, 1995

[Die1999] T. Dierks, C. Allen: *The TLS Protocol Version 1.0*. RFC 2246, January 1999

[Deu1999] D. Deugo, M. Weiss: *A Case for Mobile Agent Patterns*. Mobile Agents in the Context of Competition and Cooperation MAC^3, 1999

[Fra1996] S. Franklin, A. Graesser: *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents*. Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, 1996

[Gil1995] D. Gilbert, M. Aparicio: *IBM Intelligent Agent Strategy*. IBM Corporation, 1995

[Gen1997] General Magic, Inc.: *Odyssey web page*. General Magic, 1997
`http://www.genmagic.com/technology/odyssey.html`

[Gos1996] J. Gosling, B. Joy, G. Steele: *The Java Language Specification*. Addison-Wesley, August 1996

[Gra1996] R.S. Gray: *Agent Tcl: A flexible and secure mobile-agent system*. Proceedings of the Fourth Annual Tcl-Tk Workshop TCL'96, July 1996

[Har1995] C.G. Harrison, D.M. Chess, A. Kershenbaum: *Mobile Agents: Are they a good idea?*. Technical Report, IBM Research Division, T.J. Watson Research Center, March 1995

[Hoh1998] M. Hohlfield, B. Yee: *How to migrate agents*. Technical Report CS98-588, Computer Science and Engineering Department, University of California, June 1998

[Hou1999] R. Housley, W. Ford, W. Polk, D. Solo: *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. RFC 2459, January 1999

[Huh1994] M.N. Huhns, M.P. Singh: *Distributed Artificial Intelligence for Information Systems*. CKBS-94 Tutorial, June 1994

[Cha1994] B. Chaib-Draa: *Distributed Artificial Inteligence: An Overview*. Encyclopedia of Computer Science and Technology, Vol 29, 1994

[Chi1997] T.H. Chia, S. Kannapan: *Strategically Mobile Agents*. Proceedings of the First International Workshop on Mobile Agents MA'97, Springer, 1997

[Ibm1997] IBM Tokyo Research Laboratory: *Aglets-based e-marketplace: Concept, architecture and applications*. IBM Tokyo Research Laboratory, Research Report RT-0253, 1997

[Ibm1998] IBM, Inc.: *IBM Aglets Documentation Web Page*.
`http://aglets.trl.ibm.co.jp/documentation.html`, 1998

[Joh1995] D. Johansen, R. van Renesse, F.B. Schneider: *An Introduction on TACOMA Distributed System*. Technical Report 95-23, Department of Computer Science, University of Tromsø, June 1995

[Joh1996] D. Johansen, R. van Renesse, F.B. Schneider: *Operating System Support for Mobile Agents*. Proceedings of the 5th IEEE Workshop on Hot Topics in Operating Systems HotOS-V, May 1996

[Jul1988] E. Jul, H. Levy, N. Hutchinson, A. Black: *Fine-Grained Mobility in the Emerald System*. ACM Transactions on Computer Systems 6(1), February 1988

[Kar1997] G. Karjoth, D. Lange, M. Oshima: *A Security Model for Aglets*. IEEE Internet Computing, July-August 1997

[Kar1998a] N.M. Karnik, A.R. Tripathi: *Design Issues in Mobile Agent Programming Systems*. IEEE Concurrency 6(6), July-September 1998

[Kar1998b] N.M. Karnik, A.R. Tripathi: *Agent Server Architecture for the Ajanta Mobile-Agent System*. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA‘98, 1998

[Kar1998c] N.M. Karnik: *Security in Mobile Agent Systems*. Department of Computer Science, University of Minnesota, 1998

[Kin1995] J.A. King: *Intelligent Agents: BringingGood Things to Life*. AI Expert, February 1995

[Kon1996] D. Konstantas, J.H. Morin, J. Vitek: *MEDIA: a platform for the commercialization of electronic documents*. Object Applications, University of Geneva, 1996

[Kot1997] D. Kotz, R. Gray, S. Nog, D. Rus: *Agent Tcl: Targeting the Needs of Mobile Computers*. IEEE Internet Computing, July-August 1997

[Lab1998] Y. Labrou, T. Finin, Y. Peng: *Mobile agents can benefit from standards efforts on inter-agent communication*. IEEE Communications, Vol. 36, Nr. 7, 1998

[Lab1999] Y. Labrou, T. Finin, Y. Peng: *The current landscape of Agent Communication Languages*. IEEE Intelligent Systems, Vol. 14, No. 2, March-April 1999

[Lan1998] D. Lange, M. Oshima: *Seven Good Reason for Mobile Agent*. Communication of ACM 42, March 1998

[Lan1999a] D. Lange, M. Oshima: *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, 1999

[Lan1999b] D. Lange: *Java Mobile Agents*. JavaOne '99 BOF, 1999

[Lev1995] J.Y. Levy, J.F. Ousterhout: *A Safe Tcl Toolkit for Electronic Meeting Places*. Proceedings of the First USENIX Workshop on Electronic Commerce, July 1995

[Lin1996] T. Lindholm, F. Yellin: *The Java Virtual Machine Specification*. Addison-Wesley, 1996

[Mae1994] P. Maes: *Agents that Reduce Work and Information Overload*. Communications of ACM 37, 1994

[Mat1999] V. Matena, M. Hapner: *Enterprise JavaBeans Specification Version 1.1 Final*. Sun Microsystems, Inc., 1999

[Mil1999] D. Milojičić, F. Douglis, R. Wheeler: *Mobility: Processes, Computers, and Agents*. Addison-Wesley, February 1999

[Mit1997] Mitsubishi Electric: *Concordia: A Infrastructure for Collaborating Mobile Agents*. Proceedings of the 1st International Workshop of Mobile Agents MA' 97, April 1997

[Moc1987] P. Mockapetris: *Domain Names – Concepts and Facilities*. RFC 1034, November 1987

[Obj1997] ObjectSpace, Inc.: *ObjectSpace Voyager Core Package Technical Overview*. ObjectSpace, Inc., 1997

[Sho1994] Y. Shoam, B. Thomas: *Agent-Oriented Programming*. Encyclopedia of Computer Science and Technology, Vol 29, 1994

[Smi1988] J.M. Smith: *A Survey of Process Migration Mechanisms*. ACM Operating System Review 22(3), 1988

[Sta1990] J.W. Stamos, D.K. Gifford: *Remote Evaluation*. ACM Transactions on Programming Languages and Systems 12(4), October 1990

[Str1997] M. Straßer, M. Schwehm: *A Performance Model for Mobile Agent Systems*. International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA' 97, 1997

[Sun2001a] Sun Microsystems, Inc.: *Java 2 SDK Documentation*. Sun Microsystems, Inc., 2001

[Sun2001b] Sun Microsystems, Inc.: *Java 2 Platform API*. Sun Microsystems, Inc., 2001

[Sun2001c] Sun Microsystems, Inc.: *Object Serialization*. Sun Microsystems, Inc., 2001

[Sun2001d] Sun Microsystems, Inc.: *JavaBeans*. Sun Microsystems, Inc., 2001

[Sun2001e] Sun Microsystems, Inc.: *Java Authentication and Authorization Service*. Sun Microsystems, Inc., 2001

[Sun2001f] Sun Microsystems, Inc.: *Java Naming and Directory Interface*. Sun Microsystems, Inc., 2001

[Tar1996] J. Tardo, L. Valente: *Mobile Agent Security and Telescript*. Proceedings of COMPCON Spring '96, 1996

[Tay1990] B.H. Tay, A.L. Ananda: *A Survey of Remote Procedure Calls*. Operating Systems Review 24(3), July 1990

[The1999] W. Theilmann, K. Rothermel: *Maintaining specialized search engines through mobile filter agents*. Proceedings 3rd Int. Workshop on Cooperative Information Agents CIA'99, July 1999

[Tho1997] T. Thorn: *Programming Languages for Mobile Code*. ACM Computing Surveys 29(3), September 1997

[Tri1998] A.R. Tripathi, N.M. Karnik, M.K. Vora, T. Ahmed: *Ajanta – A System for Mobile Agent Programming*. Technical Report TR98-016, Department of Computer Science, University of Minnesota, April 1998

[Vit1981] J. Vittal: *Active Message Processing: Messages as Messengers*. Computer Message System, 1981

[Wah1997] M. Wahl, T. Howes, S. Kille: *Lightweight Directory Access Protocol*. RFC 2251, December 1997

[Whi1995] J.E. White: *Mobile Agents*. Technical Report, General Magic, Inc., October 1995

[Woo1995] M. Wooldridge, N.R. Jennings: *Agent Theories, Architectures, and Languages: a Survey*. Intelligent Agents, 1995

[Zim1995] P.R. Zimmermann: *The Official PGP User's Guide*. The MIT Press, 1995