Java RMI, CORBA und Firewalls

Rainer Falk

Lehrstuhl für Datenverarbeitung
TU München
falk@ei.tum.de

Zusammenfassung. Verteilte Objekte können nicht ohne Probleme über Firewall-Grenzen hinweg angesprochen werden. Um dies dennoch zu ermöglichen, können unterschiedliche Techniken eingesetzt werden. In diesem Beitrag werden die auftretenden Probleme erläutert und Lösungsansätze vorgestellt.

1 Einleitung

Verteilte Objekte, die z. B. durch CORBA oder Java RMI realisiert werden können, finden zunehmend Interesse für die Entwicklung von Diensten, die über das Internet genutzt werden. Dabei stellt sich das Problem, daß die Nutzung dieser Dienste auch über Firewallgrenzen hinweg möglich sein soll. In diesem Beitrag stellen wir die Problematik dar und stellen unterschiedliche Lösungsansätze vor und vergleichen diese.

Nach einer kurzen Einführung in Verteilte Objekte im Abschnitt 2 und Firewalls im Abschnitt 3 werden in Abschnitt 4 die Probleme dargestellt, die beim Zugriff auf Objekte über Firewallgrenzen hinweg auftreten. In Abschnitt 5 werden unterschiedliche Lösungsansätze und deren Eigenschaften vorgestellt. Abschnitt 6 stellt verfügbare Lösungen vor. Abschnitt 7 endet mit einer Zusammenfassung und einem Ausblick.

2 Verteilte Objekte

Es exisitieren unterschiedliche Varianten von Verteilten Systemen, z. B. CORBA, Java RMI, DSOM, DCOM. Das Grundprinzip dieser unterschiedlichen Ausprägungen ist das gleiche: Es werden Objekte definiert, die über eine festgelegte Schnittstelle angesprochen werden können. Die Schnittstelle besteht aus den Methoden, die aufgerufen werden können, und deren Signaturen, d. h. der Anzahl und der Datentypen der Parameter und Rückgabewerte. Ein solches Objekt kann nicht nur auf dem lokalen Rechner angesprochen werden, sondern auch entfernt über das Netz, siehe Abb. 1.

Um entfernte (oder auch lokale) Objekte auffinden zu können, kann ein Verzeichnisdienst verwendet werden. Im allgemeinen ist die Verteilung der Rollen Client und Server nicht fest vorgegeben, sondern es kann beispielsweise der Client im Aufruf des Server-Objekts Referenzen auf lokale Objekte übergeben. Diese

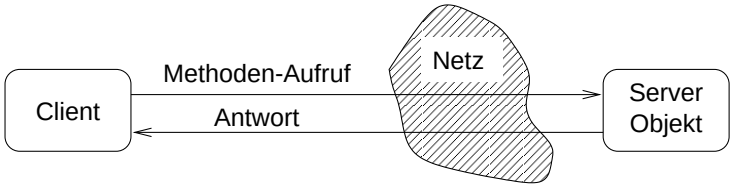

Abb. 1. Verteilte Objekte

können dann vom Server angesprochen werden, wobei sich dabei die Rollen des Aufrufers und des Aufgerufenen vertauschen. Natürlich können auch noch weitere Objekte, die sich auf anderen Rechnern befinden, angesprochen werden.

Im Java-Umfeld sind RMI und CORBA besonders interessant, da beide von JDK 1.2 unterstützt werden. RMI ist im Java-Umfeld entstanden. Alle beteiligten Objekte sind in Java implementiert. Es ist keine Ortstransparenz gegeben, d. h. der Nutzer muß wissen, an welchem Ort (auf welchem Rechner) sich ein Objekt befindet. CORBA hingegen ist nicht auf Java festgelegt, sondern es kann eine Vielzahl von Programmiersprachen verwendet werden. Damit eignet sich CORBA dazu, in unterschiedlichen Sprachen implementierte Teilsysteme zu einem Gesamtsystem zu integrieren. Bei CORBA sind die Objekte ortstransparent, d. h. nicht an einen bestimmten Rechner gebunden. Insgesamt ist CORBA zwar leistungsfähiger als RMI, dadurch aber auch komplexer. Es hängt vom konkreten Anwendungsfall ab, welche der beiden Alternativen die geeignetere ist (vgl. [3]).

3 Firewalls

Viele Firmennetze sind an das Internet angeschlossen, um die Vorteile einer Internetanbindung zu nutzen. Damit verbunden ist aber auch eine Gefährdung des Firmennetzes durch Angriffe aus dem Internet. Um diesen entgegenzuwirken, werden Firewalls an der Grenze zwischen dem Firmennetz und dem Internet verwendet. Diese überwachen und kontrollieren den Verkehr zwischen den beiden Netzen. Es ist keine uneingeschränkte Kommunikation zwischen Rechnern aus dem Firmennetz und Rechnern aus dem Internet möglich, sondern nur solche Verkehrsbeziehungen, die von der Firewall zugelassen werden, können diese passieren.

Firewalls können den Verkehr auf unterschiedlichen Schichten untersuchen. In der Praxis wird meist eine Kombination aus Paketfilterung und Proxys verwendet. Paketfilter untersuchen einzelne IP-Pakete. Aufgrund von im Paketkopf enthaltenen Informationen (Quell- und Zieladresse, Protokoll, Quell- und Zielport) wird entschieden, ob das Paket weitergeleitet wird. Proxys untersuchen den Datenstrom auf den Anwendungsschichten. Es werden Circuit-Proxys unterschieden, die eine Verbindung nur durchschalten, und Application-Proxys, die auf eine spezielle Anwendung zugeschnitten sind und dadurch auch die Anwendungsdaten untersuchen können.

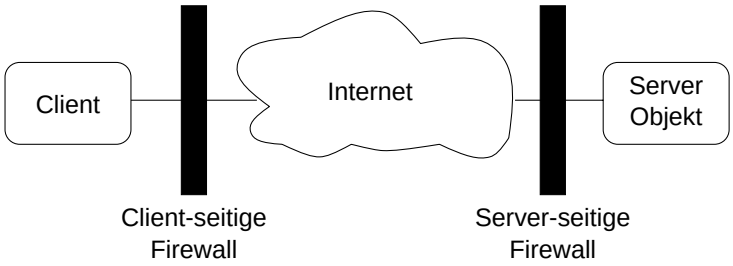

Abb. 2. Szenario

Außerdem besteht die Möglichkeit, den Aufbau des Firmennetztes nach außen – zumindest teilweise – zu verbergen. Dazu können sogenannte *private IP-Adressen* [11] eingesetzt werden, die im Internet nicht gültig sind. Von der Firewall werden diese Adressen dynamisch durch gültige ersetzt (NAT, network address translation). Dadurch können aber die davon betroffenen Rechner im geschützten Netz nicht mehr direkt von außen adressiert werden.

4 Probleme

Abbildung 2 zeigt ein mögliches Szenario. Es soll von einem Client aus ein Server-Objekt angesprochen werden. Sowohl der Client als auch das Server-Objekt befinden sich jeweils hinter einer Firewall. Damit der Aufruf erfolgreich ausgeführt wird, müssen folgende Barrieren überwunden werden:

1. Client-seitige Firewall
2. Server-seitige Firewall
3. evtl. Applet-Beschränkungen (Sandboxing)

Im allgemeinen finden Aufrufe nicht nur von einem Client zu einem Server-Objekt statt, sondern auch das Server-Objekt kann weitere Objekte aufrufen. Diese können sich beim Client (Callback) befinden oder auf weiteren Rechnern. Es genügt für den allgemeinen Fall also nicht, wenn eine Firewall für Methodenaufrufe in nur einer Richtung durchlässig ist. Weiterhin besteht die Möglichkeit, daß auf Client- bzw. Serverseite nicht nur eine Firewall vorhanden ist, sondern mehrere, kaskadierte Firewalls.

4.1 Client-seitige Firewall

Die Client-seitige Firewall soll das Netz, in dem sich der Client befindet, vor Angriffen aus dem Internet schützen. Es ist zu unterscheiden, ob die Möglichkeit besteht, die Konfiguration der Client-seitigen Firewall zu modifizieren, oder ob die Konfiguration nicht verändert werden kann. Der erste Fall kann z. B. vorliegen, falls zwei Partnerfirmen Geschäftsdaten austauschen wollen (business-to-business). Es kann beispielsweise einem Lieferanten die Möglichkeit eingeräumt

werden, den aktuellen Lagerbestand und bestimmte Daten aktueller Aufträge abzufragen. Der zweite Fall liegt vor, falls der Client ein beliebiger Rechner im Internet sein kann (business-to-customer). Es ist dann weder bekannt, welche Konfiguration beim Client vorhanden ist, noch kann diese angepaßt werden.

4.2 Server-seitige Firewall

Hier ist nur der Fall von Interesse, daß das Objekt von außen angesprochen werden können soll[1]. Es kann dann davon ausgegangen werden, daß die Firewall so konfiguriert wird, daß das Objekt von außen angesprochen werden kann. Dazu muß die Kommunikation zum einen zugelassen sein, zum anderen muß der Rechner, auf dem sich das Objekt befindet, auch überhaupt von außen angesprochen werden können. Besondere Schwierigkeiten ergeben sich, falls vom Objektsystem Ortstransparenz unterstützt wird, weil dann nicht festgelegt ist, auf welchem Rechner sich das Objekt befindet.

4.3 Applet-Beschränkungen

Die Kommunikationsmöglichkeiten von Applets sind – im Gegensatz zu denen von Applikationen – eingeschränkt. So können sie nur Netzwerkverbindungen zu dem selben Host aufbauen, von dem das Applet geladen wurde. Weiterhin ist es nicht möglich, daß eine Verbindung zu einem Applet hin aufgebaut wird.

Diese Einschränkungen können aufgehoben werden, falls signierte Applets verwendet werden.

5 Lösungsansätze

5.1 Öffnen eines Ports

Falls das Objekt, auf das zugegriffen werden soll, auf einem bestimmten Rechner auf einem festgelegten Port läuft, besteht die Möglichkeit, die Firewalls sowohl auf Client- wie auch auf Server-Seite so zu konfigurieren, daß Netzwerkverbindungen zu diesem Rechner auf dem festgelegten Port zugelassen werden. Damit kann dann dieses Objekt von außen direkt angesprochen werden.

Dies erfordert eine Konfigurationsänderung beider Firewalls. Bei dieser Lösung werden direkte Verbindungen vom bzw. in das Internet zugelassen. „Direkte Verbindung“ bedeutet, daß die entsprechende Verbindung nicht über einen Proxy geführt wird, der den Datenaustausch überwachen kann. Falls der Client ein Applet ist, müssen das betrachtete Objekt und der WWW-Server, von dem das Applet geladen wurde, auf dem selben Rechner laufen. Bei dieser Konfiguration besteht nicht die Möglichkeit, daß das Server-Objekt auf vom Client bereitgestellte Objekte zugreifen kann, da keine Möglichkeit bereitgestellt wurde, einen Rückkanal aufzubauen.

[1] Andernfalls sollte die Firewall natürlich so konfiguriert werden, daß das Objekt nicht von außen angesprochen werden kann. Das stellt aber i. a. kein Problem dar.

5.2 Einbettung (Tunnelung)

Die Objektanfragen und Antworten können in andere Protokolle, die zugelassen sind, eingebettet (getunnelt) werden. Es bietet sich hierfür die Verwendung von HTTP an. Diese Lösung erfordert bei der Client-seitigen Firewall keine Konfigurationsänderung, da HTTP normalerweise – zumindest über einen Proxy – zugelassen ist. Dabei ist zu unterscheiden, ob HTTP-Anfragen zu einem beliebigen Zielport oder nur zu einigen festgelegten (z. B. 80, 8000, 8080) zugelassen sind.

Auf Serverseite muß die Möglichkeit bestehen, die eingebetteten Aufrufe zu verarbeiten. Entweder ist das Server-Objekt selbst in der Lage, eingebettete Aufrufe auszuwerten, oder es ist ein Proxy vorzusehen, der die in andere Protokolle eingebetteten Nachrichten wieder auspackt und in gewöhnliche Aufrufe übersetzt. Die Antwort muß ebenfalls in das entsprechende Protokoll eingepackt und zurückgesendet werden. Auf der Client-Seite muß die Möglichkeit gegeben sein, die Anfrage in andere Protokolle einzubetten.

Der Aufruf von Methoden vom Server-Objekt aus auf vom Client bereitgestellte Objekte ist schwierig. Es besteht allerdings die Möglichkeit, daß der Client eine Verbindung zum Proxy aufrechterhält, um Methodenaufrufe vom Server-Objekt entgegenzunehmen.

5.3 Proxys

Es kann sowohl auf Server- als auch auf Client-Seite ein Proxy vorgesehen werden, um die jeweilige Firewall überqueren zu können.

Server-seitige Proxys. Proxys auf Server-Seite können nicht nur dazu dienen, Tunnelung über HTTP zu ermöglichen, sondern auch dazu, Applet-Beschränkungen zu umgehen. Methodenaufrufe von einem Client werden an einen Proxy gerichtet, der auf dem selben Rechner wie der WWW-Server läuft. Dieser leitet die Anfrage an das eigentliche Zielobjekt weiter, das sich auch auf einem anderen Rechner befinden kann.

Abhängig von der Konfiguration auf Client-Seite ist es möglich, daß HTTP-Anfragen nur zu einigen festgelegten Portnummern zugelassen sind. Außerdem kann es wegen Applet-Beschränkungen erforderlich sein, daß der Proxy auf dem Rechner läuft, von dem das Applet geladen wurde. Beide Anforderungen lassen sich erfüllen, wenn der Proxy auch als Web-Server arbeiten kann. Beide Funktionen können dann unter *einer* Portnummer angesprochen werden. Abhängig von der Art der Anfrage verhält er sich als Web-Server oder als Proxy.

Durch Verwendung eines Proxys vereinfacht sich die Konfiguration der Server-seitigen Firewall, da alle Anfragen an den Proxy gerichtet werden, der auf einer festgelegten Portnummer laufen kann. Es müssen dann nicht Verbindungen zu allen in Frage kommenden Zielrechnern und Portnummern zugelassen werden, sondern nur zum Proxy. Durch einen Proxy auf Server-Seite kann es ermöglicht werden, solche Objekte anzusprechen, die sich auf Rechnern befinden, die von außen nicht direkt adressierbar sind. Ein Proxy auf Server-Seite

kann auch verwendet werden, um einzuschränken, welche Objekte und Methoden angesprochen werden können.

Client-seitige Proxys. Es besteht auch die Möglichkeit, auf der Firewall, die sich auf der Client-Seite befindet, einen Proxy vorzusehen. Dieser kann verwendet werden, falls die Firewall auf Client-Seite so konfiguriert ist, daß Objekte außerhalb der Firewall nicht direkt angesprochen werden können.

Adressierung der Proxys. Wenn Proxys verwendet werden, müssen auch Vorkehrungen getroffen werden, damit der Proxy anstatt des eigentlichen Zielobjekts angesprochen wird. Dazu kann zum einen die Objektreferenz so modifiziert werden, daß der Proxy angesprochen wird. Dann muß das Objektsystem auf der Client-Seite nicht für die Verwendung eines Proxys ausgelegt sein, da aus Sicht des Clients sich die modifizierte Objektreferenz nicht von einer regulären unterscheidet. Es müssen aber auch die Objektreferenzen modifiziert werden, die als Parameter oder Rückgabewert übergeben werden.

Es kann aber auch das Objektsystem auf der Client-Seite die Möglichkeit bieten, einen Proxy anzusprechen. Dieses muß dazu geeignet konfiguriert werden. In diesem Fall ist es nicht notwendig, die verwendeten Objektreferenzen zu modifizieren.

6 Konkrete Lösungen

6.1 Java RMI

Bei RMI [12] wurde die Firewall-Problematik von vornherein berücksichtigt. Dabei wird davon ausgegangen, daß von einem beliebigen Rechner aus, der sich auch hinter einer Firewall befinden kann, RMI-Aufrufe möglich sein sollen. Es wird dabei auf folgende Arten versucht, das entsprechende Server-Objekt anzusprechen:

1. Zuerst wird versucht, das Zielobjekt direkt anzusprechen.
2. Falls dies scheitert, wird der Aufruf in eine HTTP-POST-Anfrage eingebettet. Der Client-seitige HTTP-Proxy leitet die Anfrage direkt an das Zielobjekt weiter. Dazu muß der HTTP-Proxy so konfiguriert sein, daß Verbindungen zu beliebigen Portnummern erlaubt sind.
3. Schlägt auch diese Variante fehl, weil der HTTP-Proxy nur Anfragen zu bestimmten Ports weiterleitet, dann wird die Anfrage an den HTTP-Server gerichtet. Es wird ein cgi-Skript (`/cgi-bin/java-rmi`) aufgerufen, das die Anfrage entgegennimmt, die gewünschte Methode des entsprechenden Objekts aufruft und die Antwortdaten einpackt, um sie an den Client zurückzuschicken.

Falls die Anfragen in HTTP eingebettet werden, besteht nicht die Möglichkeit, Callbacks vom angesprochenen Server zurück zum Client zu verwenden.

6.2 CORBA-Proxys

Es sind bereits mehrere CORBA-Proxys verfügbar [13, 6]. Diese bieten folgende Funktionalität:

- Aufrufe an den CORBA-Server werden nicht direkt an das Zielobjekt gerichtet, sondern an den Proxy. Dieser leitet die Anfrage an das eigentliche Zielobjekt weiter.
- Es besteht die Möglichkeit, CORBA-Anfragen in HTTP einzubetten. Hierfür existieren keine Standards. Deshalb müssen hierfür spezielle ORBs verwendet werden, die proprietäre Verfahren verwenden.
- Die Proxys können auch als HTTP-Server arbeiten. Diese Funktion ist nützlich, um Applets vom selben Server zu laden, auf dem auch der Proxy läuft.
- Für die Unterstützung von Callbacks werden teilweise proprietäre Verfahren eingesetzt. Falls die Anfragen in HTTP eingebettet werden, sind keine Callbacks möglich.

Bei den einzelnen Produkten werden unterschiedliche Mechanismen verwendet. Teilweise werden spezielle Eigenschaften einer ORB-Implementierungen ausgenutzt. Dadurch ist die Interoperabilität zwischen beliebigen ORBs eingeschränkt.

Durch die Verwendung eines CORBA-Proxys ergeben sich folgende Vorteile:

- Die Server-seitige Firewall kann leichter konfiguriert werden. Es müssen eingehende Verbindungen nicht zu allen in Frage kommenden Zielen für CORBA-Objekte zugelassen werden, sondern nur zum Proxy. Bei [6] kann dieser auch eine Zugriffskontrolle bis auf Methodenebene durchführen.
- Im Zusammenhang mit Applets muß ein Proxy verwendet werden, da Applets nur Netzwerkverbindungen zum gleichen Rechner aufbauen können, von dem das Applet geladen wurde. Anfragen können vom Applet an den Proxy gerichtet werden, der sie an das Zielobjekt weiterleitet.
- Sie ermöglichen es, Aufrufe in HTTP einzubetten. Der Proxy packt die Anfrage aus und reicht sie an das Zielobjekt weiter. Das Zielobjekt muß deshalb nicht die Möglichkeit besitzen, selbst in HTTP eingebettete Anfragen bearbeiten zu können. Ebenso werden die Antwortdaten vom Proxy in HTTP eingepackt.

Im aktuellen CORBA-Standard ist die Verwendung von Proxys nicht vorgesehen. Es werden bei den einzelnen Proxys unterschiedliche Verfahren zur Adressierung verwendet. Bei [13] muß auf der Client-Seite ein spezieller ORB verwendet werden, der dafür ausgelegt ist, einen Proxy ansprechen zu können. Bei [6] werden die Objektreferenzen (IOR) so modifiziert, daß anstatt des eigentlichen Zielobjekts der Proxy angesprochen wird. Es ist bei diesen Proxys nicht möglich, mehrere Proxys zu kaskadieren.

6.3 OMG Request for Proposal

Von der Object Management Group OMG, die CORBA standardisiert hat, wurde die Problematik erkannt, die eine Nutzung von CORBA über Firewallgrenzen

hinweg mit sich bringt. Dies führte zu einem Request for Proposal [8], zu dem bereits Vorschläge eingereicht wurden [9]. Darin werden folgende Arten von Firewalls unterschieden:

- *TCP-Firewalls* sind Firewalls, die nur aufgrund von IP-Adressen und Portnummern filtern.
- *SOCKSv5* [7] ist ein generischer Proxy-Dienst, der von vielen Firewalls unterstützt wird. Die Daten des Anwendungsprotokolls werden nicht untersucht[2].
- *GIOP Proxy* bezeichnet einen CORBA-spezifischen Applikations-Proxy.

Um Server-seitige Firewalls überqueren zu können, werden die Objektreferenzen (IOR) um Information erweitert, wie die unterschiedlichen Arten von Firewalls zu überqueren sind. Dabei ist auch vorgesehen, daß mehrere Firewalls kaskadiert sein können. Außerdem wird vorgeschlagen, das verwendete Protokoll (GIOP) so zu erweitern, daß es bidirektional genutzt werden kann, um die Callback-Problematik zu lösen. Weiterhin wird eine Schnittstelle definiert, mit dem eine Anwendung einem CORBA-Proxy mitteilen kann, daß ein Objekt von außen angesprochen werden können soll. Die Anwendung erhält vom Proxy eine Objektreferenz, mit der das Objekt von außen – über den Proxy – angesprochen werden kann.

Wie die Client-seitigen Firewalls zu überwinden sind, ist Sache des ORBs auf der Client-Seite. In [9] werden nur die Aspekte behandelt, die für eine Interoperabilität notwendig sind. Es wird kein Vorschlag unterbreitet, wie CORBA-Anfragen in andere Protokolle (speziell HTTP) einzubetten sind. Dies ist dann notwendig, wenn die Firewall auf der Client-Seite nicht so konfiguriert ist, daß CORBA-Anfragen durchgelassen werden.

7 Zusammenfassung und Ausblick

Bereits heute lassen sich mit den verfügbaren Lösungen Verteilte Objekte über Firewall-Grenzen hinweg ansprechen. Damit lassen sich viele in der Praxis vorkommende Szenarien abdecken. Es müssen dabei aber Einschränkungen in Kauf genommen werden wie z. B. Leistungseinbußen durch die Verwendung eines Proxys oder von Tunnelung, die Festlegung auf Produkte eines Herstellers, die eingeschränkte Möglichkeit, Ortstransparenz nutzen zu können, oder die fehlende Möglichkeit, Callbacks zu verwenden. Sollen Verteile Objekte auch über Firewall-Grenzen hinweg angesprochen werden, müssen diese Randbedingungen berücksichtig werden.

In diesem Beitrag wurde nicht betrachtet, wie der Verkehr zwischen Objekten gegen Abhören oder Veränderung geschützt werden kann. Dazu besteht sowohl bei RMI als auch bei CORBA u. a. die Möglichkeit, SSL [5] zu verwenden.

[2] Grundsätzlich besteht auch bei SOCKS die Möglichkeit, Applikations-Proxys zu verwenden.

Abkürzungen

CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
DSOM	Distributed System Object Model
GIOP	General Inter-ORB Protocol
HTTP	Hypertext Transfer Protocol
IIOP	Internet Inter-ORB Protocol
IOR	Interoperable Object Reference
JDK	Java Development Kit
NAT	Network Address Translation
OMG	Object Management Group
ORB	Object Request Broker
RFP	Request for Proposal
RMI	Remote Method Invocation
SOCKS	generisches Protokoll, um Proxy anzusprechen
SSL	Secure Sockets Layer

Literatur

1. Chapman, D. B., Zwicky, E. D.: Building Internet Firewalls. O'Reilly & Associates, Inc., Newton, MA (1995)
2. Cheswick, W. R., Bellovin, S. M.: Firewalls und Sicherheit im Internet. Addison-Wesley, Bonn (1996)
3. Curtis, D.: Java, RMI and CORBA. OMG Whitepaper (1997), http://www.omg.org/news/wpjava.htm
4. Farley, J.: Java Distributed Programming. O'Reilly (1998)
5. Freier, A., Karlton, P., Kocher, P.: The SSL Protocol Version 3.0. Netscape Corp. (1996), http://home.netscape.com/eng/ssl3/
6. IONA Technologies, Wonderwall Administrator's Guide (1997), http://www.iona.com/
7. Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., Jones, L.: SOCKS protocol version 5. RFC 1928 (1996)
8. OMG, CORBA/Firewall Security – Request for Proposal (1997), http://www.omg.org/library/schedule/Firewall_RFP.htm
9. OMG, Joint Revised Submission CORBA/Firewall Security (1998), http://www.omg.org/library/schedule/Firewall_RFP.htm
10. Redlich, J.-P.: Corba 2.0. Addison-Wesley (1996)
11. Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, J., Lear, E.: Address Allocation for Private Internets. RFC 1918 (1996)
12. SUN, Java Remote Method Invocation Specification. Revision 1.42, JDK 1.2 beta 1 edn. (1997)
13. Visigenic, Gatekeeper Guide Version 3.0 (1997), http://www.inprise.com/visibroker/
14. Vogel, A., Duddy, K.: Java Programming with CORBA. Wiley, 2nd edn. (1998)