

# **Client/Server-Systeme**

**Prof. Dr.-Ing. Wilhelm G. Spruth**

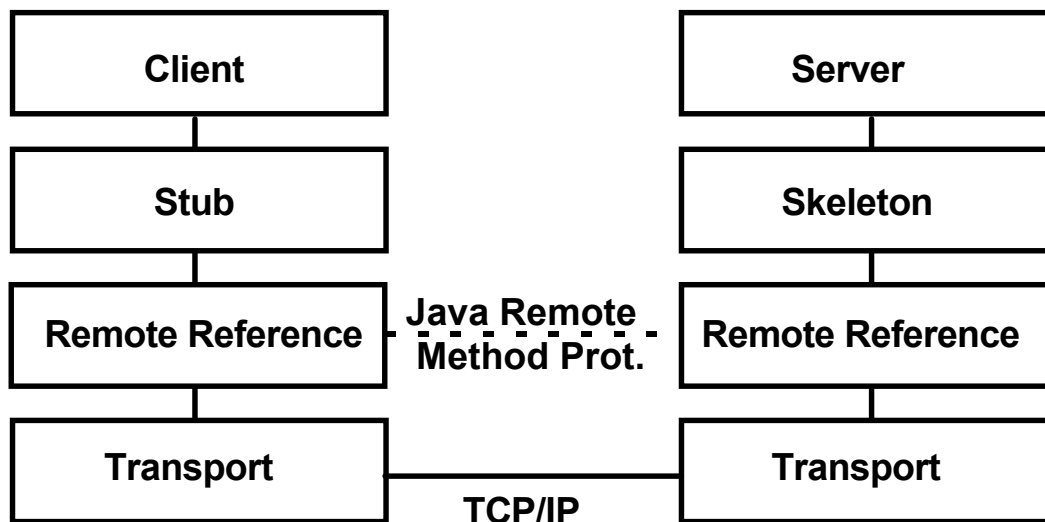
**SS 2004**

**Teil 15**

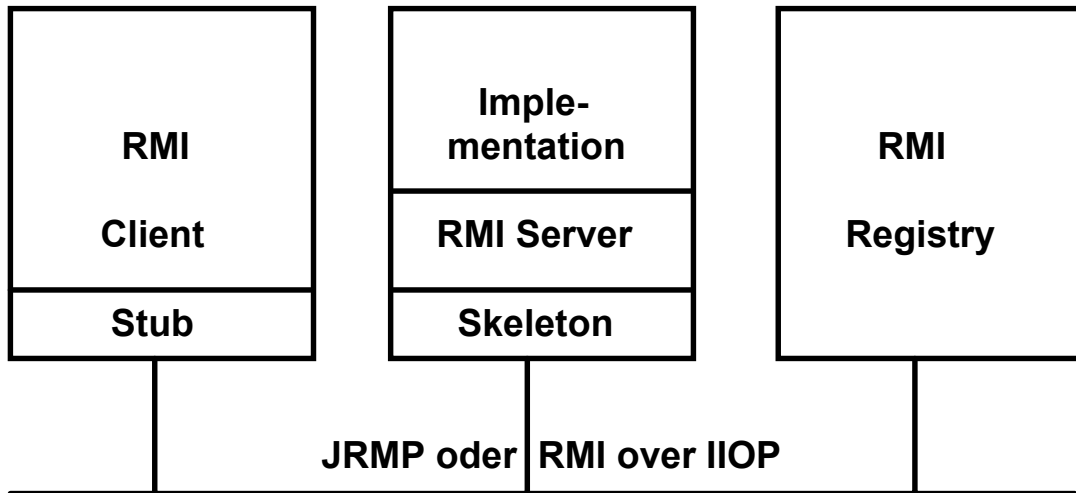
**Intersystem-Kommunikation  
RMI, DCOM, Webservices, SNA**

# Remote Method Invocation (RMI)

Aufruf von Java Programmen auf geographisch entfernten Rechnern



## Remote Method Invocation (RMI)



Drei verschiedene Prozesse, die auf dem gleichen Rechner oder auf entfernten Maschinen laufen können. RMI Registry ist ein einfacher RMI Namensdienst. Die Alternative ist JNDI

Der Klient besorgt sich eine Handle für das entfernte Objekt, indem er RMI Registry aufruft (`///URL/registered name`).

Eine Referenz auf das entfernte Objekt wird zurückgegeben. Jetzt kann eine Methode des entfernten Objektes aufgerufen werden.

Dieser Aufruf erfolgt zum Stub, der das entfernte Objekt repräsentiert.

Der Stub verpackt die Argumente (marshaling) in einen Datenstrom, der über das Netzwerk geschickt wird.

Das Skeleton unmarshals die Argumente, ruft die Methode, marshals die Ergebnisswerte und schickt sie zurück.

Der Stub unmarshals die Ergebnisswerte und übergibt sie an das Klientenprogramm.

## Erstellen einer RMI Remote Class

RMI benötigt wie Corba einen RMI Server, unter dem die RMI Implementation läuft.

Zu kodieren sind:

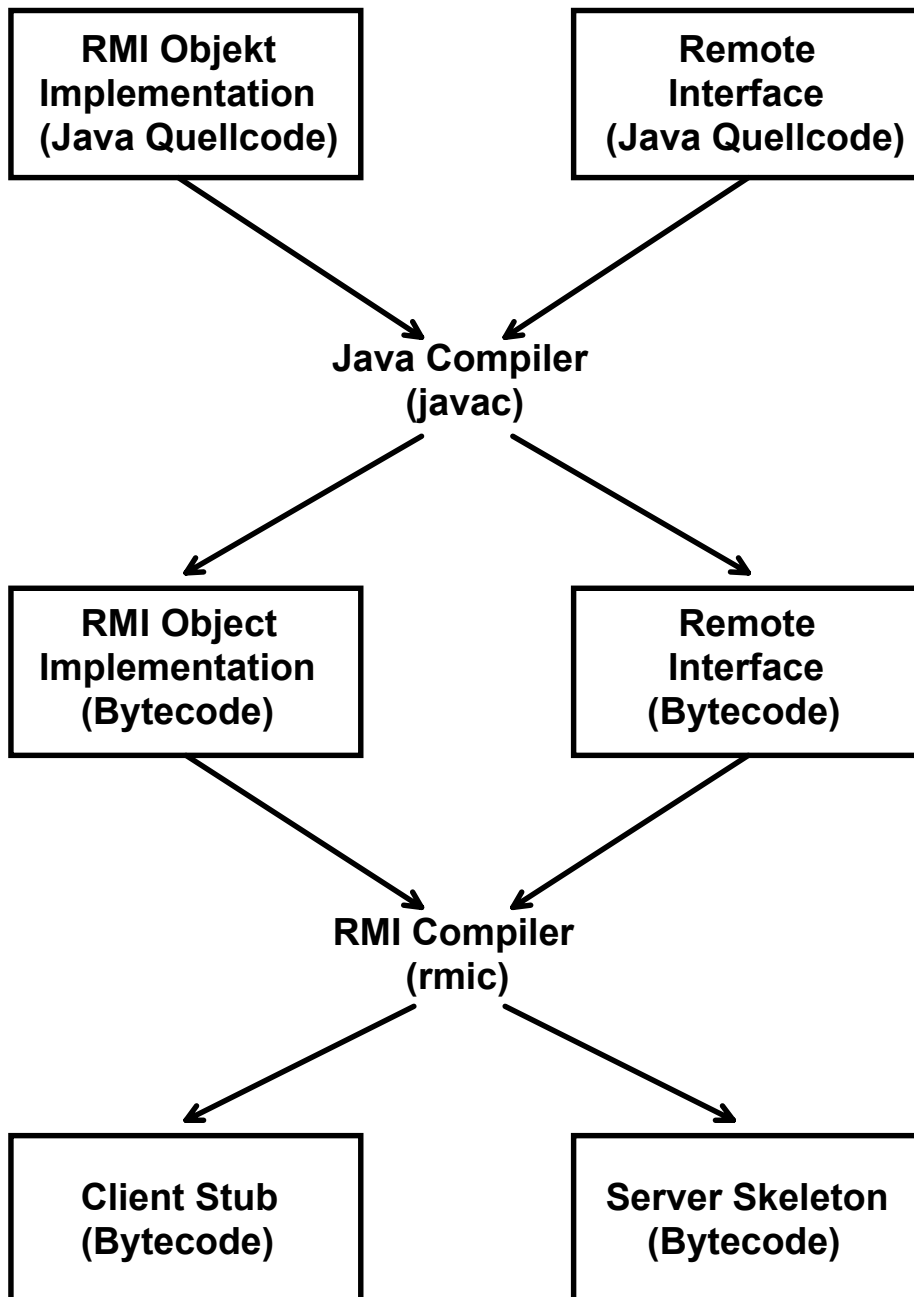
- Interface
- Client
- Implementation
- Server

Die Remote Interface enthält die Namen aller Methodenaufrufe und die dazugehörigen Parameter. Beispiel für die Interface einer Methode Addition, die zwei Zahlen a und b addiert:

```
public interface Addition extends java.rmi.Remote {
    public long add(long a, long b)
        throws java.rmi.RemoteException;
}
```

### Schritte zur Erstellung und Ausführung einer Anwendung

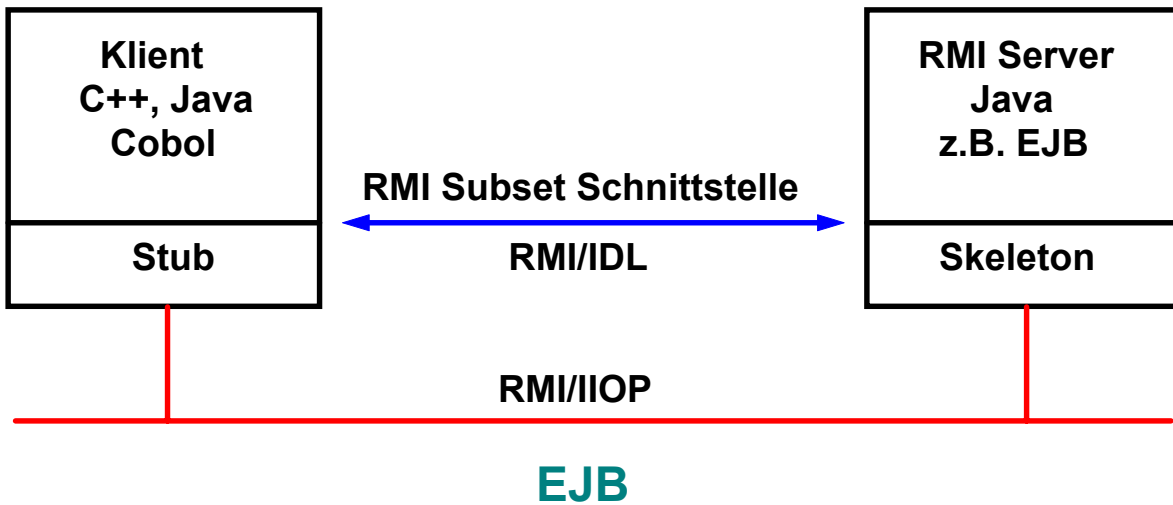
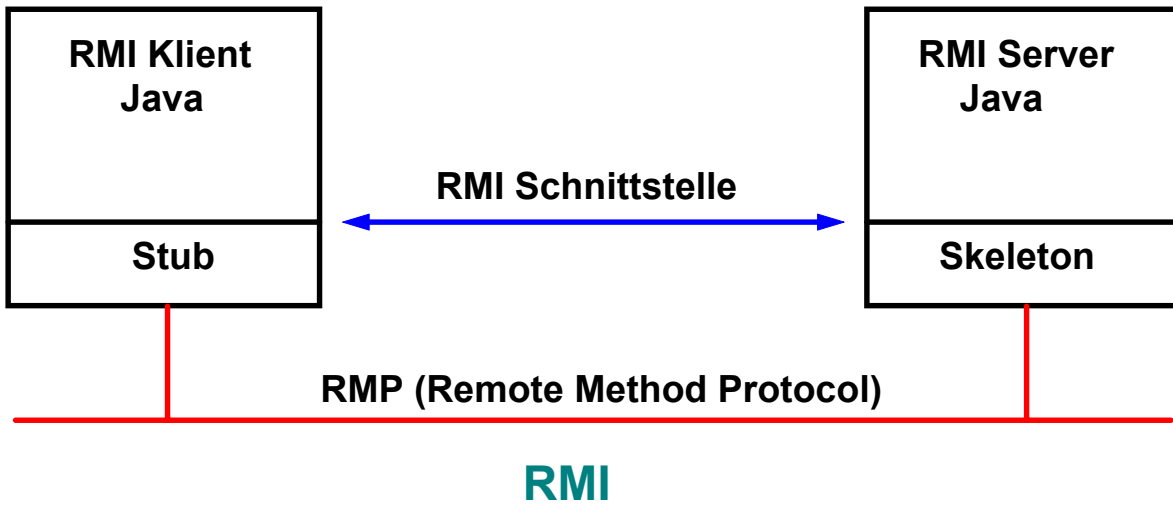
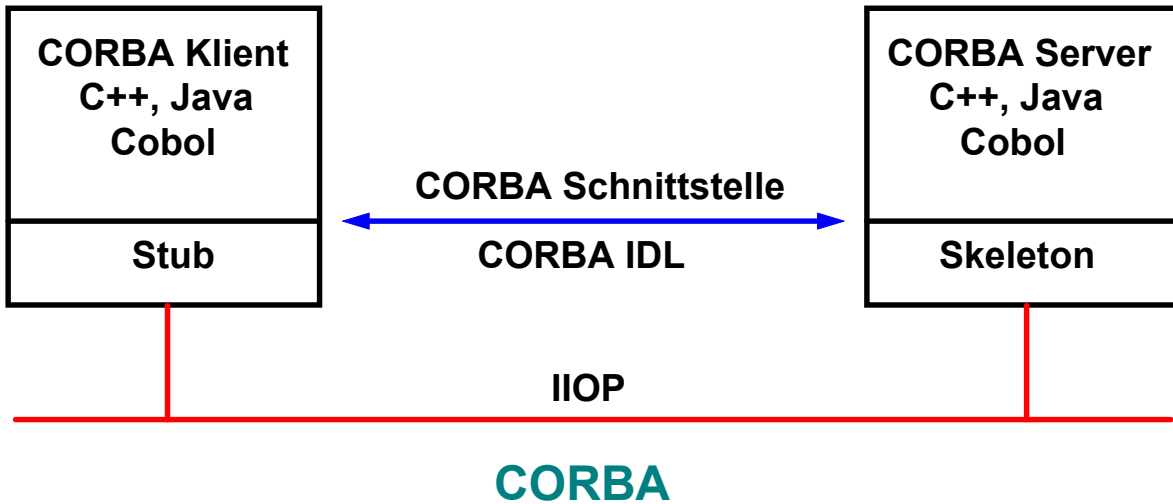
1. Interface definieren, mit der das Remote Object aufgerufen wird.  
> extends interface java.rmi.Remote .
2. Implementierung der Server Anwendung schreiben. Muss die Remote Interface implementieren. .  
> extends java.rmi.server.UnicastRemoteObject
3. Klassen kompilieren.
4. Mit dem Java RMI Compiler Client Stubs und Server Skeletons erstellen. > rmic xyzServerImpl
5. Klient implementieren und übersetzen.
6. Start Registry, Server starten, Klienten starten.

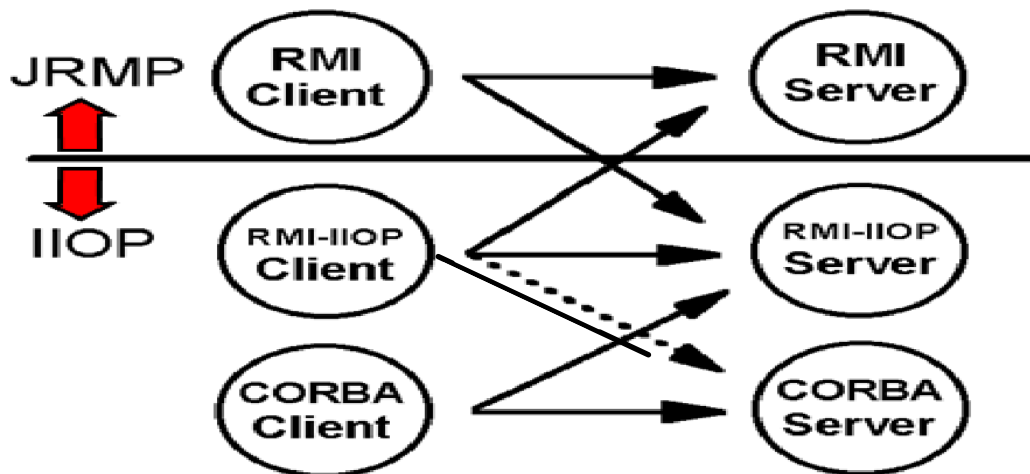


## Entwicklungsprozess für RMI Objekte

## Vergleich CORBA - RMI

- **Corba Anwendungen können in vielen unterschiedlichen Sprachen geschrieben werden, solange für diese Sprachen ein „interface definition language (IDL) mapping“ vorhanden ist. Dies schließt C/C++, Ada, Fortran, und Cobol ein; weitere Sprachen sind in Vorbereitung. RMI ist auf Java beschränkt.**
- **Mit der IDL ist die Interface von der Implementation sauber getrennt. Es können unterschiedliche Implementierungen unter Benutzung der gleichen Interface erstellt werden.**
- **RMI Anwendungen sind einfacher zu erstellen als Corba Anwendungen, weil die Notwendigkeit der IDL Definition entfällt-**
- **RMI ermöglicht serialisierbare Klassen. Code und Objekte können über das Netz übertragen werden (can be marshaled), solange der Empfänger über eine Java Virtuelle Maschine (JVM) verfügt. CORBA erlaubt keine Übertragung von Code oder Objekten; es können nur Datenstrukturen übertragen werden.**
- **Corba hat ein besseres Leistungsverhalten als RMI (keine Interpretation).**





## RMI-over-IIOP

Corba verwendet das IIOP Protokoll für die Client-Server Kommunikation

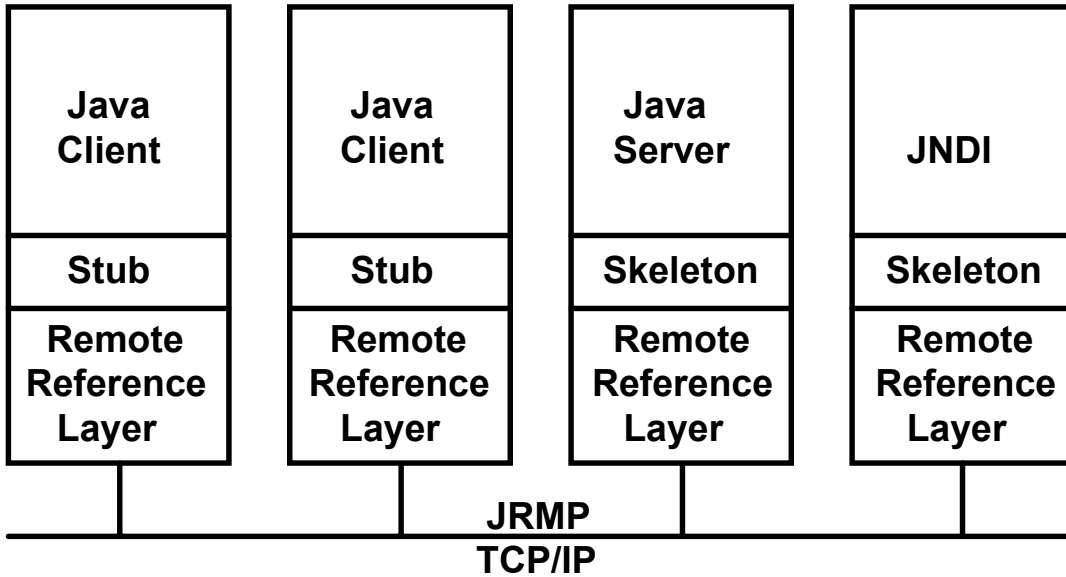
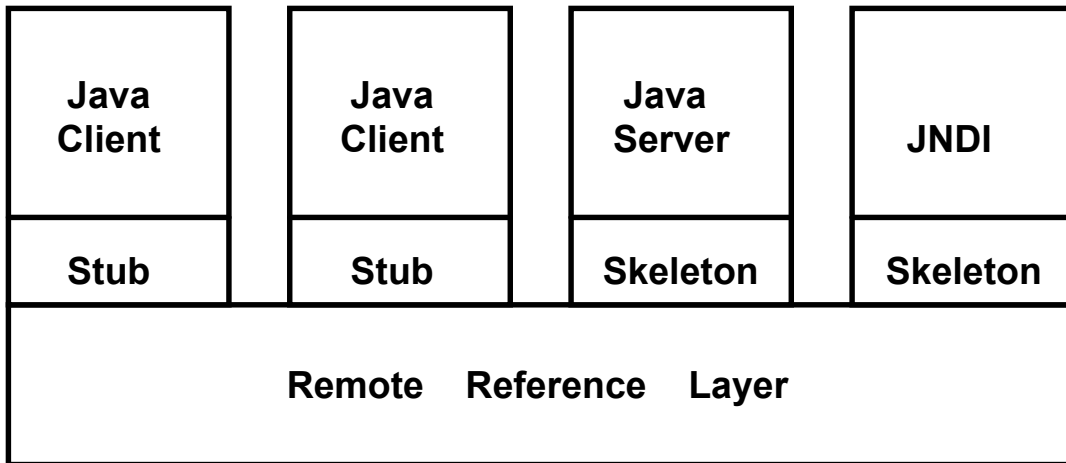
RMI verwendet das JRMP Protokoll für die Client-Server Kommunikation

Der Java J2EE Standard und sein JDK unterstützt „RMI-over-IIOP“ als zusätzliche Alternative zu dem bisherigen „RMI-over-JRMP“. Dies bedeutet, es wird IIOP an Stelle von JRMP als Übertragungsprotokoll eingesetzt.

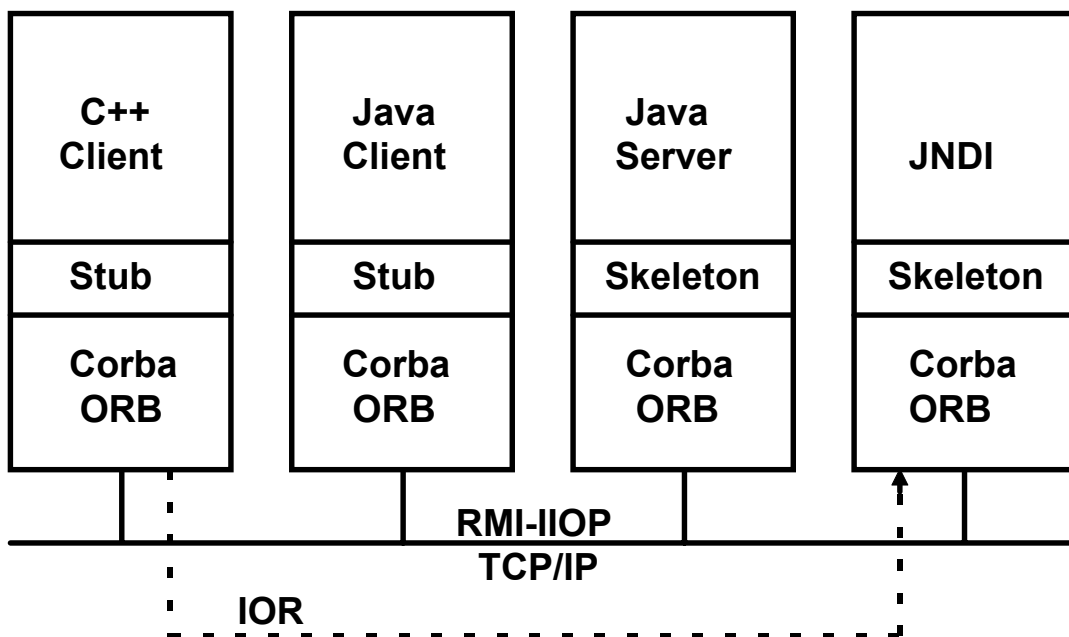
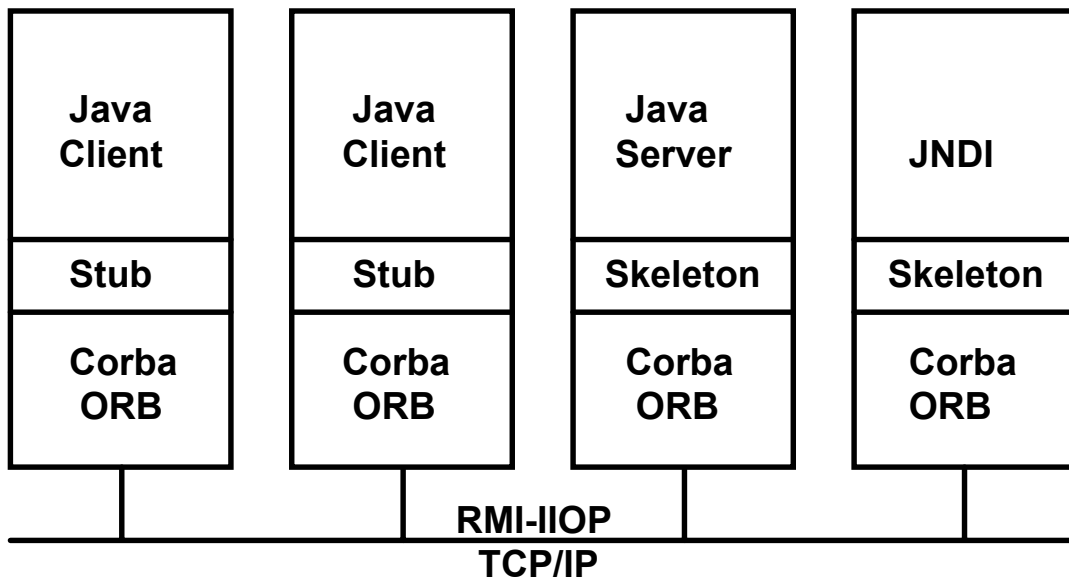
Es wird der `rmic-iiop` Compiler an Stelle des bisherigen `rmic` Compilers eingesetzt, um die RMI Stubs und Skeletons zu erzeugen. Davon abgesehen ändert sich wenig an der Erzeugung und dem Betrieb von RMI - Java Anwendungen. Einige Hersteller, z.B. IBM, wollen deshalb RMI-over-JRMP ganz durch RMI-over-IIOP ablösen.

Soll eine RMI Komponente (in Java) mit einer Corba Komponente (z.B. in C++) kommunizieren, so ist es erforderlich, aus der Java Interface Definition mit Hilfe des `rmic-idl` Compilers eine IDL File zu erstellen. Letztere erzeugt dann mit Hilfe des normalen IDL Compilers Stubs und Skeletons.



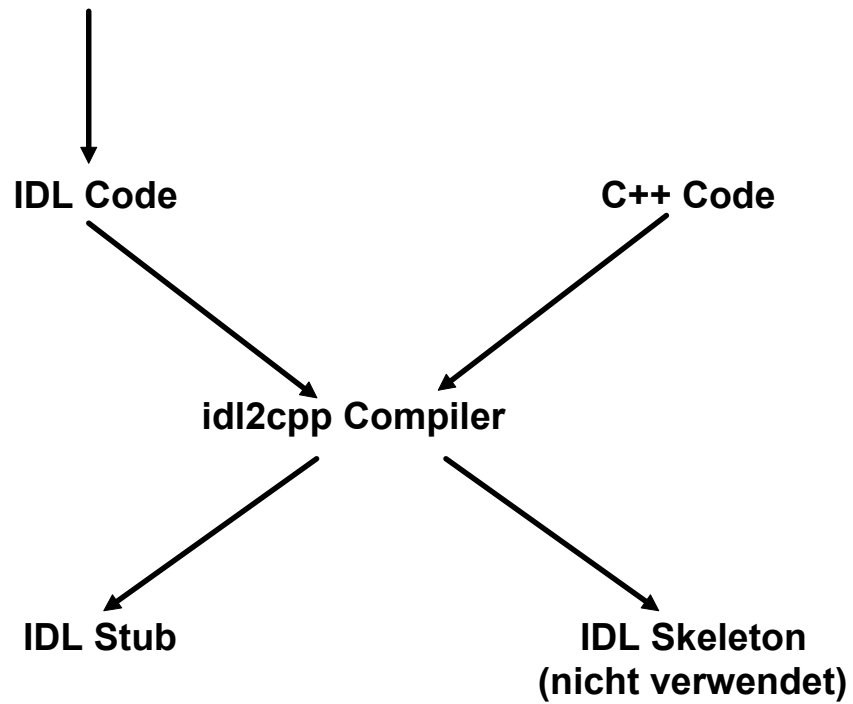


## Java Remote Methode Invokation (RMI)



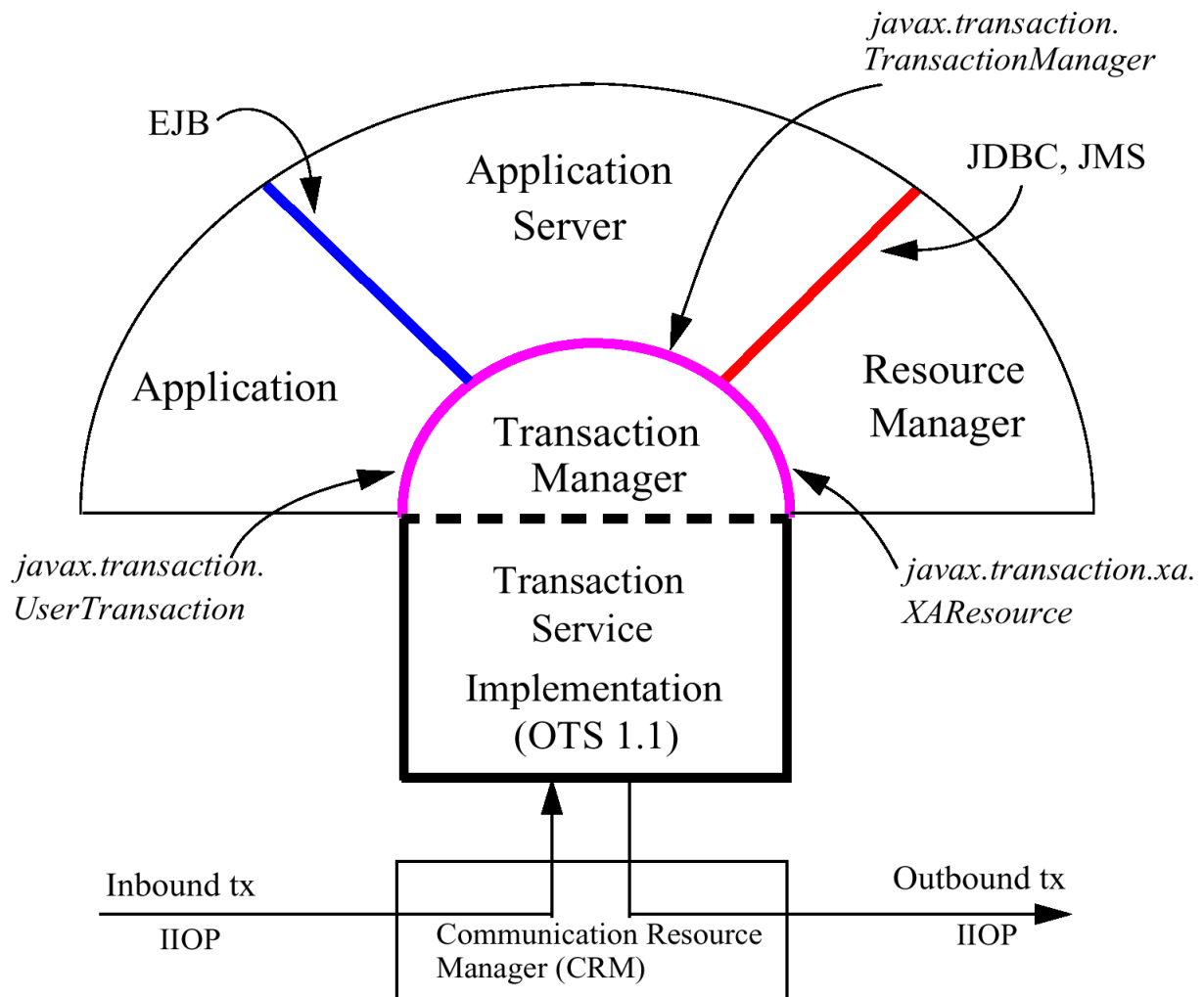
## RMI over IIOP

**Interface des Java Server Objects**  
`public interface xxx extends java.rmi.Remote { .....`



**Erstellen des Stubs für einen C++ Client  
in einer Java RMI-IIOP Umgebung**

## Java Transaction Service (JTS) Java Transaction API (JTA)



**Der Java Transaction Service (JTS) ist eine Java Abbildung (mapping) des OTS. JTS Kommunikation zwischen zwei Web Application erfordert daher IIOp (Beispiel 2 Phase Commit) .**

**Der Transaction Manager implementiert die JTA. Die Schnittstelle zwischen Transaction Manager und Transaction Service ist Hersteller abhängig. In der Praxis wird der Transaction Service häufig durch einen existierenden Transaktionsmonitor implementiert:**

- **BEA Web Logic benutzt Tuxedo**
- **IBM WebSphere benutzt CICS und Encina**

<https://jsecom15c.sun.com/ECom/EComActionServlet/LegalPage::~:com.sun.sunit.sdlc.content.LegalWebPageInfo;jsessionid=jsecom15c.sun.com-23665%3A40c1ecfb%3A7c3aab86ae6384cd>

## **Microsoft DCOM**

**Das DCOM Objektmodell ist Microsoft's proprietäre Alternative zu CORBA.**

**Verwendet (leicht modifizierten) DCE RPC.**

**Integriert (wie Corba) Komponenten unterschiedlicher Hersteller in binärer Form, die in unterschiedlichen Sprachen geschrieben sein können. Verwendet hierzu eine IDL.**

**Keine Vererbung.**

**Verfügbar unter OS/390 und den meisten Unix Dialekten (incl. LINUX), jedoch bisher kaum Einsatz außerhalb der Windows Betriebssystemfamilie.**

## **Vergleich CORBA - DCOM**

**Beide Technologien werden vermutlich für längere Zeit in Wettbewerb miteinander stehen.**

**In reinrassigen Microsoft Umgebungen (z.B. mittelständische Unternehmen) hat DCOM Vorteile.**

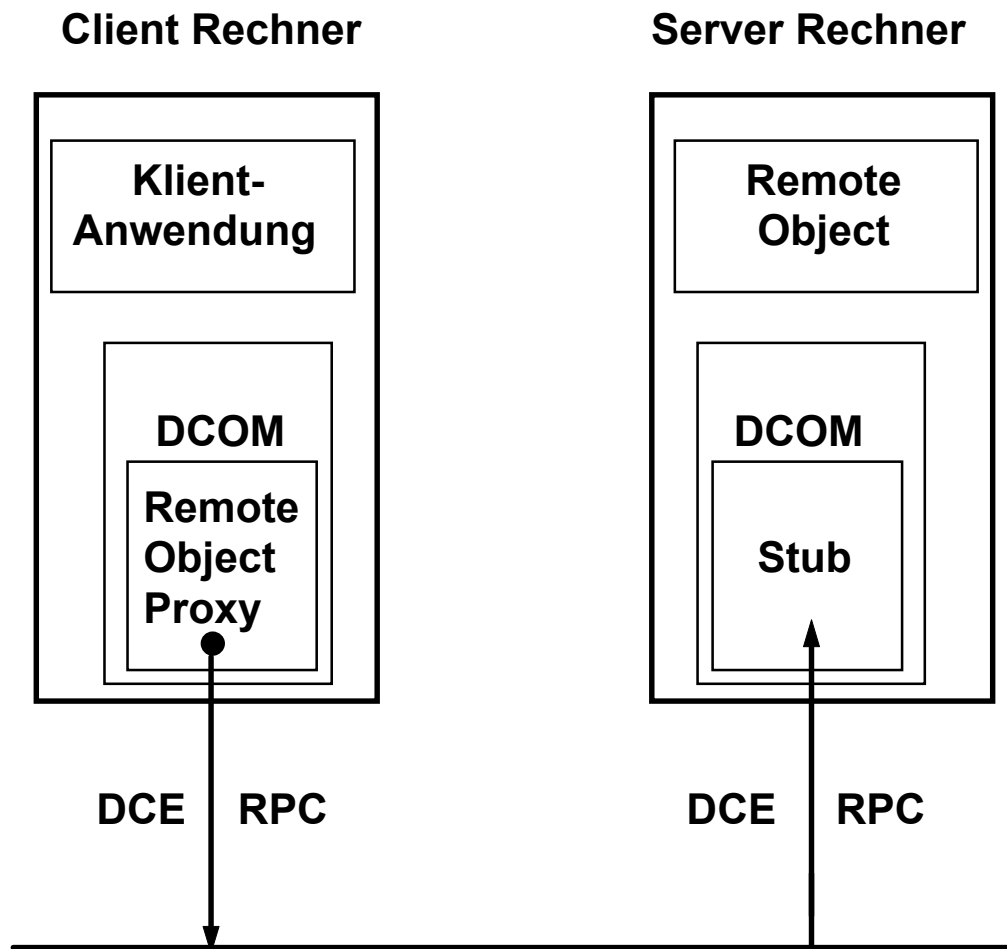
**In heterogenen Umgebungen ist DCOM bisher wenig vertreten.**

**CORBA ist besser geeignet, bestehende Altanwendungen einzubinden.**

**Es ist denkbar, auf weitverbreitete Echtzeitbetriebssysteme wie VxWorks oder Psos eine CORBA basierte Anwendung aufzusetzen.**

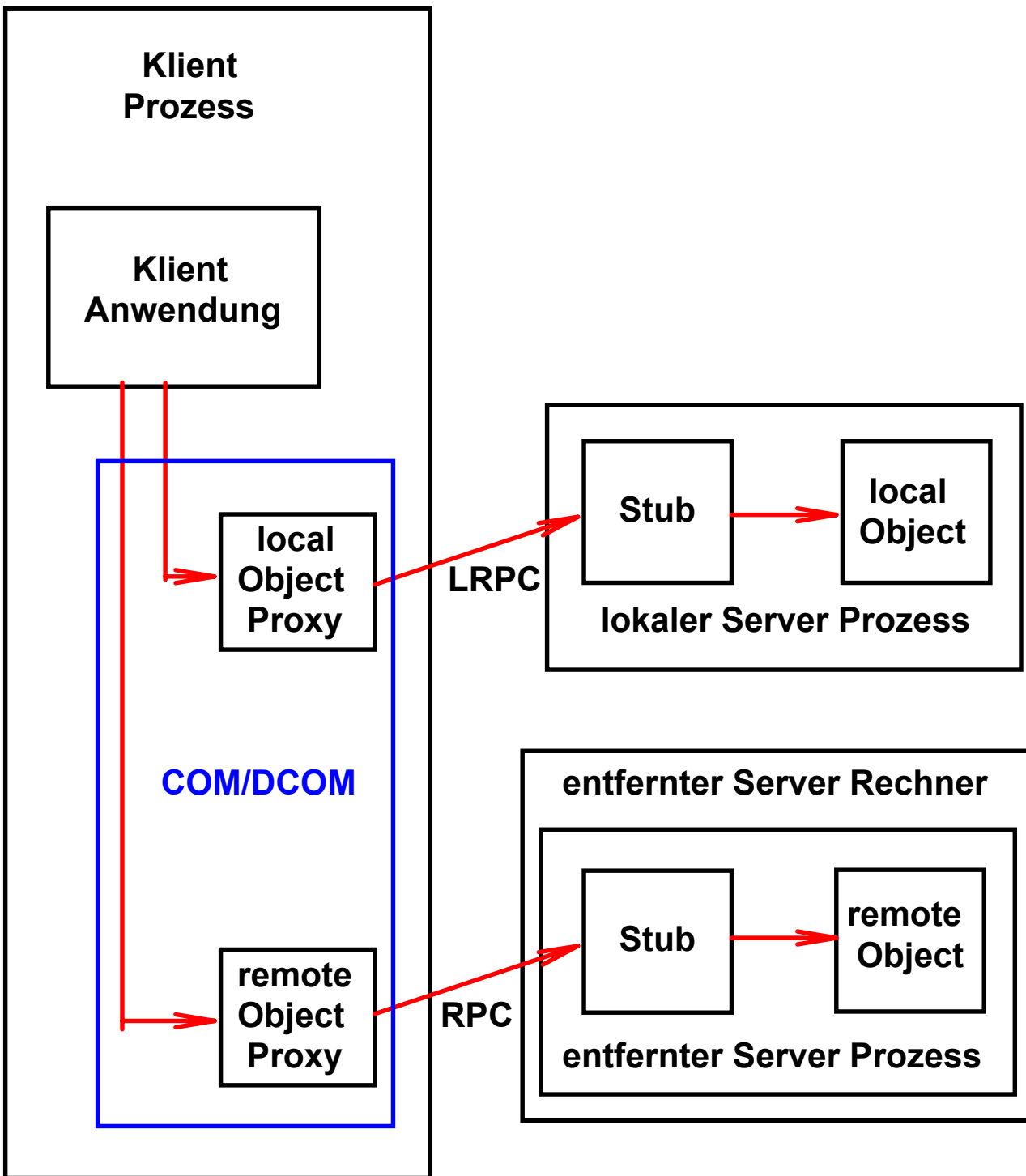
**Java Browser Sicherheit (Sandbox) ist besser als DCOM (vertrauensbasierte Verfahren).**

**Gartner: DCOM wird 2002 den CORBA Stand von 1998 erreichen**

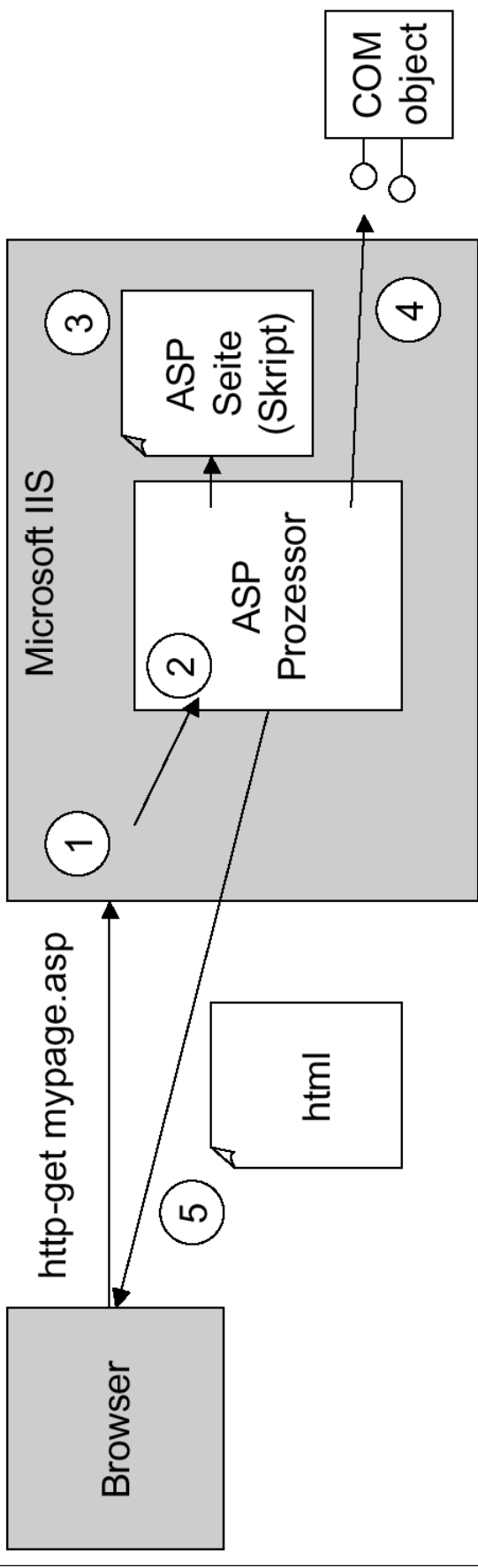


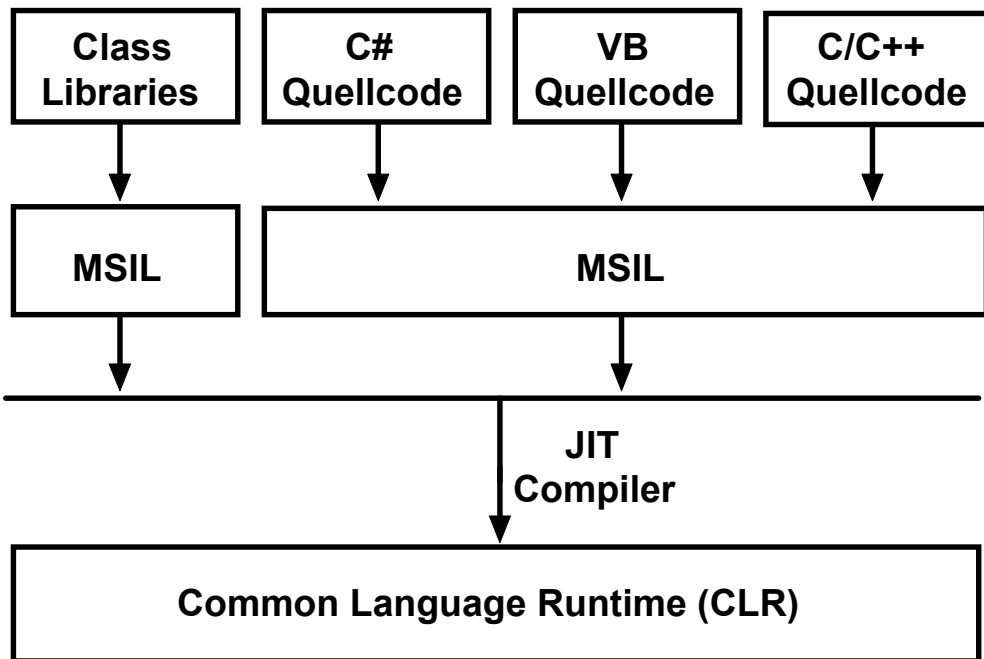
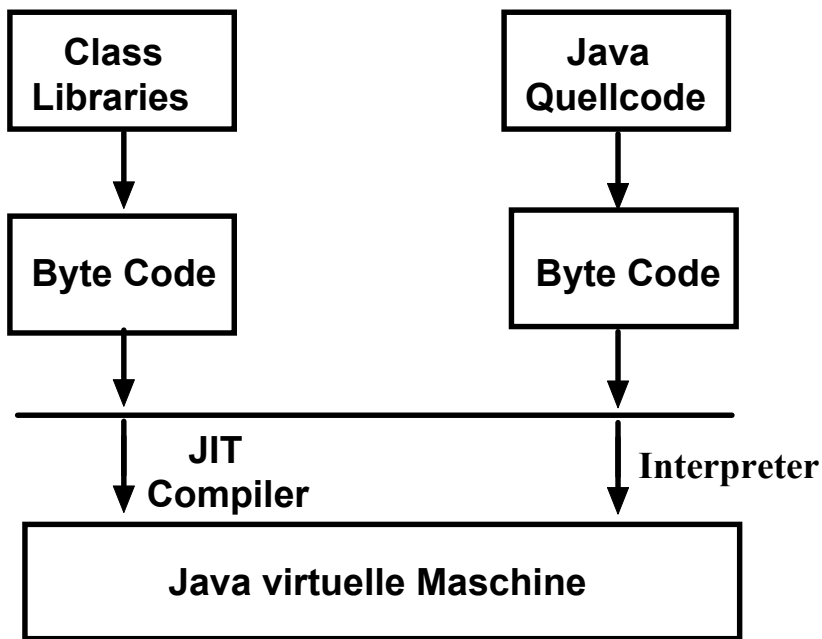
## DCOM Kommunikation über Remote Procedure Calls

Arbeitet der Server auf dem gleichen Rechner wie der Klient, werden „Lightweight RPC's“ eingesetzt (LPC, Local Procedure Call, des NT Überwachers).



**COM/DCOM kommunizieren über den (DCE) RPC. Befinden sich Klient und Server auf der gleichen Maschine, werden Lightweight RPC's (LRPC) eingesetzt.**





## Microsoft Common Language Runtime (CLR)

# **DCOM Leistungsverhalten**

**Beispiel: Pentium 120 Mhz, NT 4.0 Betriebssystem,  
50 Bytes an Parametern zwischen 2 Prozessen  
transportieren**

- 1. Transport zwischen 2 Prozessen auf dem gleichen  
Rechner ohne DCOM  
3 Millionen Aufrufe/s**
- 2. Transport zwischen 2 Prozessen auf dem gleichen  
Rechner unter Verwendung von DCOM und dem MT LPC  
2000 Aufrufe/s**
- 3. Transport mit DCOM zwischen 2 Prozessen auf  
unterschiedlichen Rechnern  
500 Aufrufe/s**

## **Was ist ein Webservice?**

**ein Service, den man über ein Internet Standard –  
Protokoll aufrufen kann**

**Gemeint aber eher ein Service der über http mit dem Simple  
Object Access Protocol (SOAP) aufgerufen wird.**

**SOAP ist eine XML Definition**

# Web Services

## Beispiel Yahoo Portal

Die meisten Dienste kommen in Wirklichkeit von anderen Web Sites: Reisen, Wetter, Landkarten, oder Web Suche mit Hilfe von Google. Diese Dienste sind in das Yahoo Portal als eigene Dienste integriert.

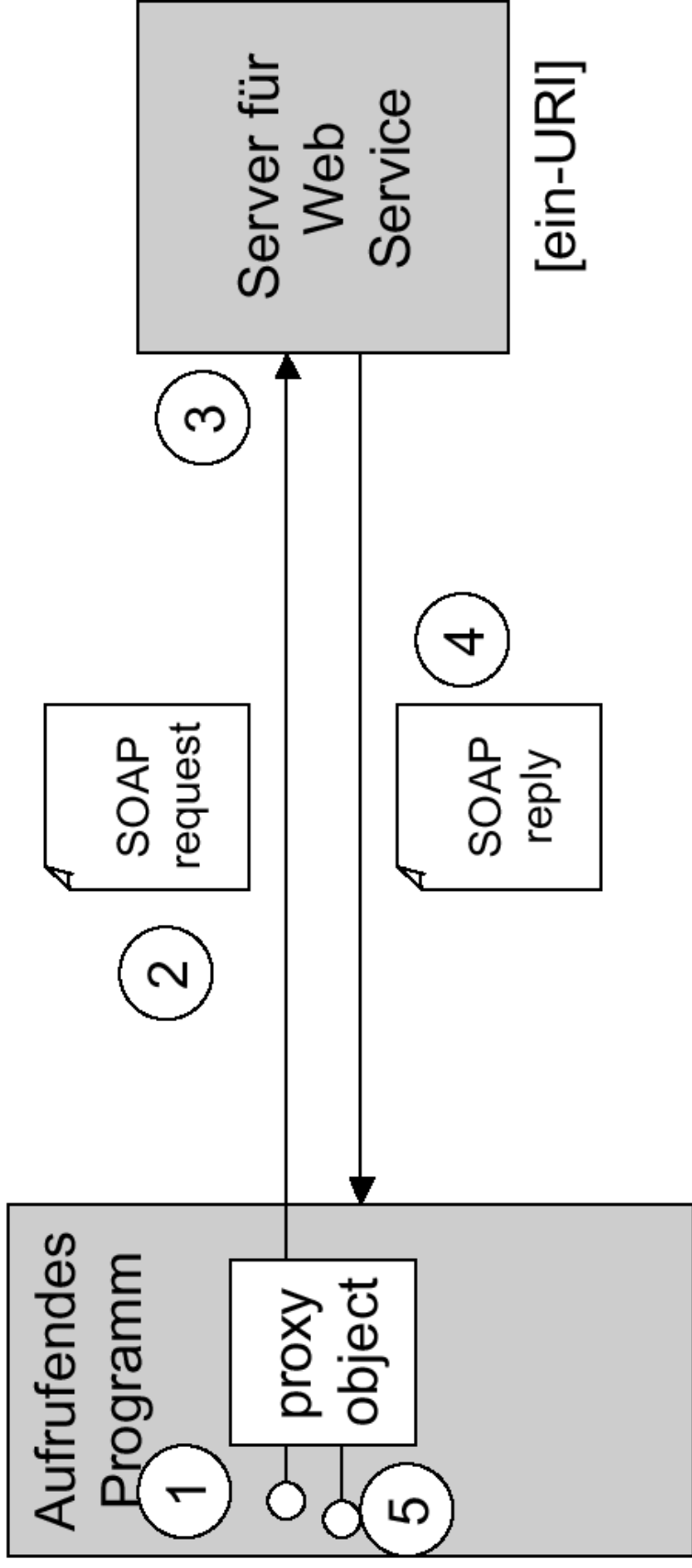
- **SOAP**            Simple Object Access Protocol
- **WSDL**           Web Service Description Language
- **UDDI**            Universal Description, Discovery, and Integration

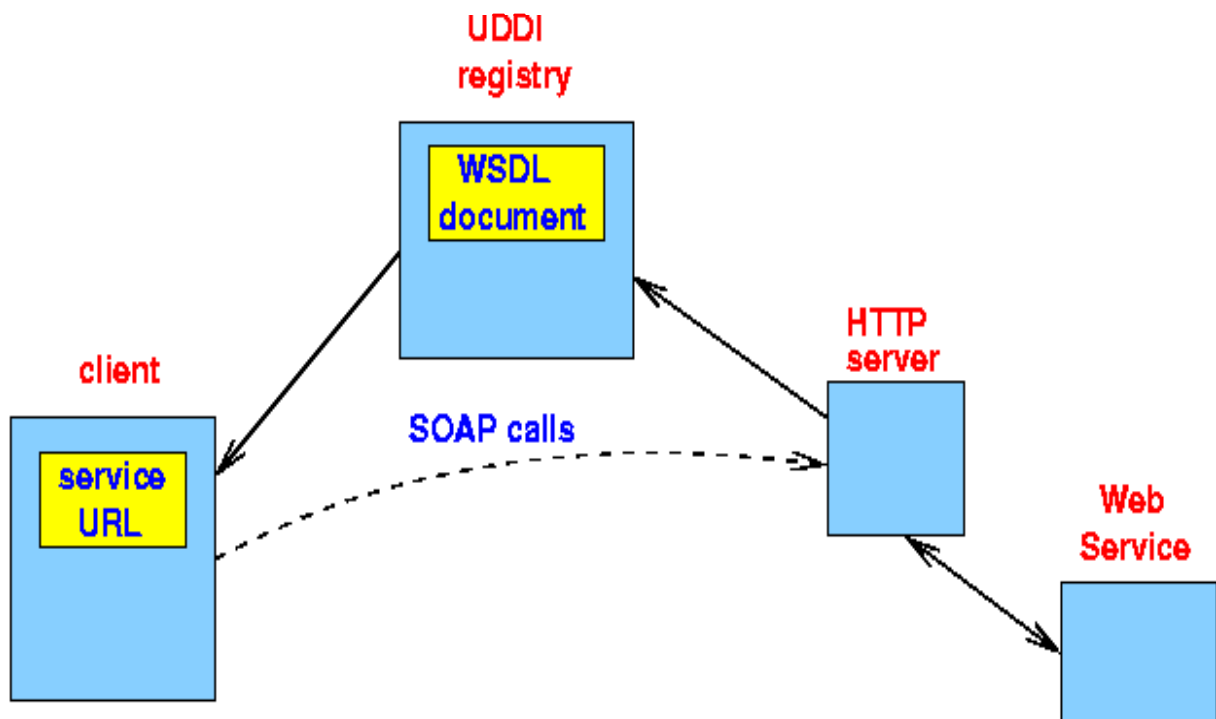
SOAP ist ein Remote Procedure Call (RPC) Protokoll. Es benutzt Standard Internet Protokolle für den Transport - entweder HTTP für synchrone Aufrufe oder SMTP für asynchrone Aufrufe. XML beschreibt das Format der übertragenen Daten. SOAP ist unabhängig von der verwendeten Programmiersprache, dem Objekt Modell und dem jeweiligen Betriebssystem.

WSDL beschreibt die Schnittstellendefinitionen eines Web Service. Beschreibt Formate der Anforderungs- und Antwort-Nachrichtenströme, mit denen Funktionsaufrufe an andere Programm-Module abgesetzt werden.

UDDI ist ein Dienstekatalog. Stellt ein Verzeichnis von Adress- und Produktdaten sowie Anwendungsschnittstellen der verschiedenen Web Service Anbieter dar.

**IBM - Microsoft Kooperation**  
**Firewall - HTTP**  
**Reifegrad: Sicherheit, Transaktionsdienste.**





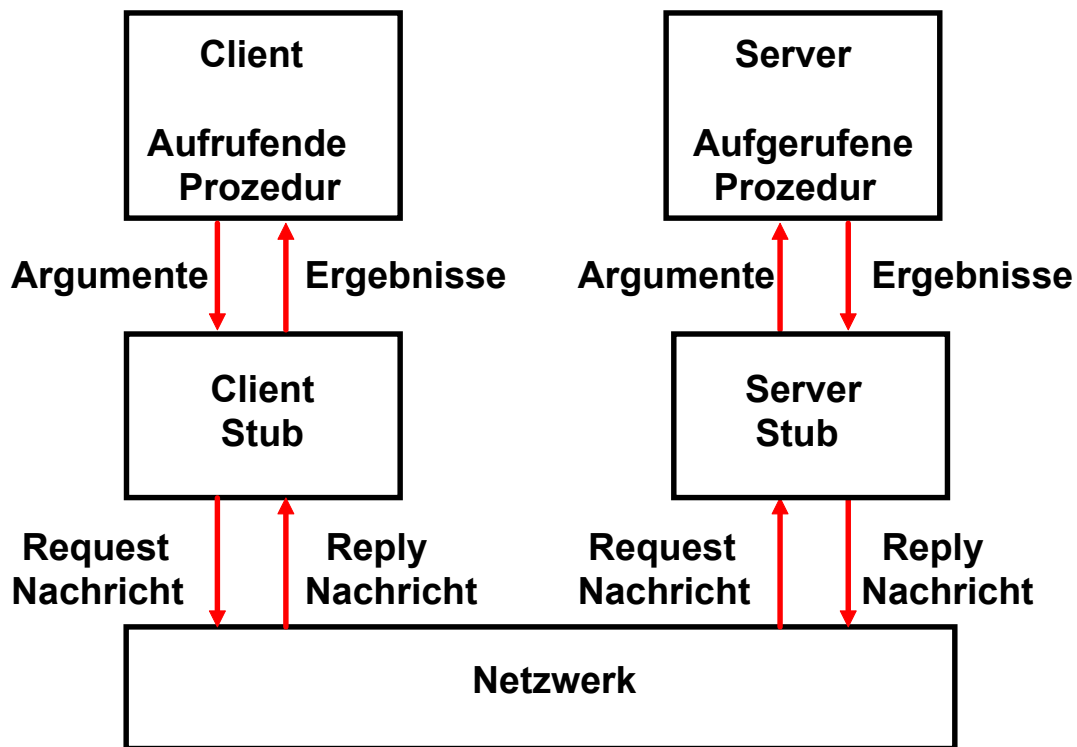
## Web Services Struktur

### Eigenschaften:

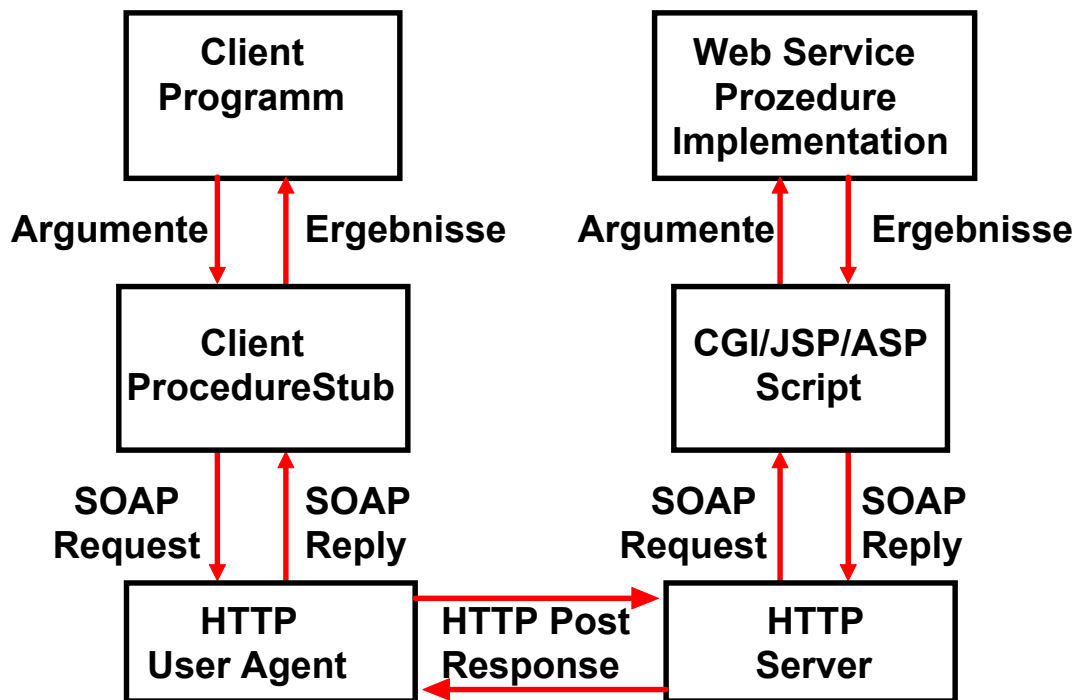
- Quick and dirty Implementierung von unkomplizierten Aufgaben
- Remote Procedure Call über Port 80
- WSDL an Stelle von IDL - automatische Erstellung, z.B. IBM's Web Services ToolKit (WSTK)
- Java → WSDL → Java - identische Ergebnisse ?
- keine Objektorientierung
- skaliert schlecht

### Was fehlt:

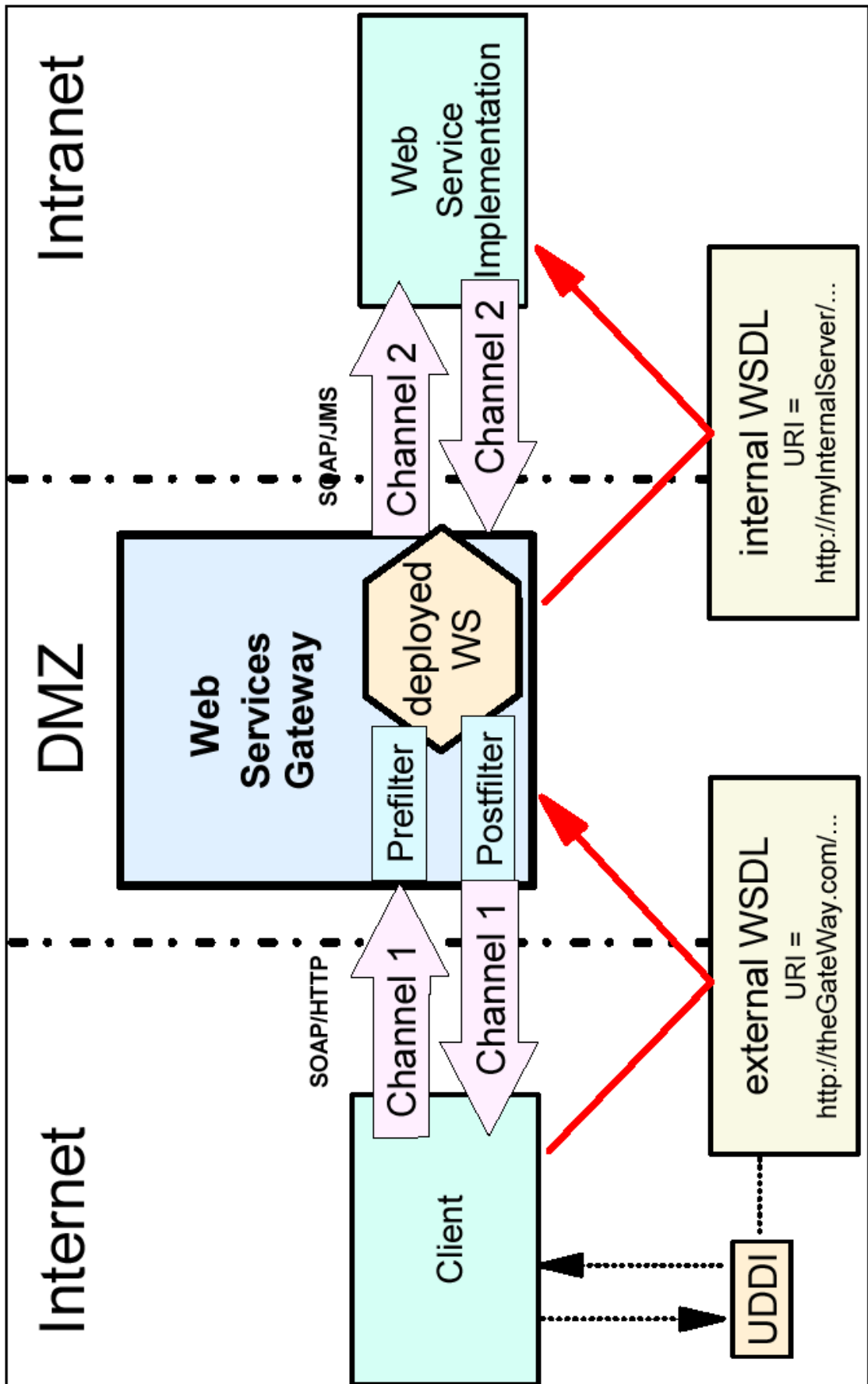
- Sicherheit - HTTPS kann benutzt werden, ist aber unabhängig vom Web Service Mechanismus
- Transactionssteuerung, Flußsteuerung

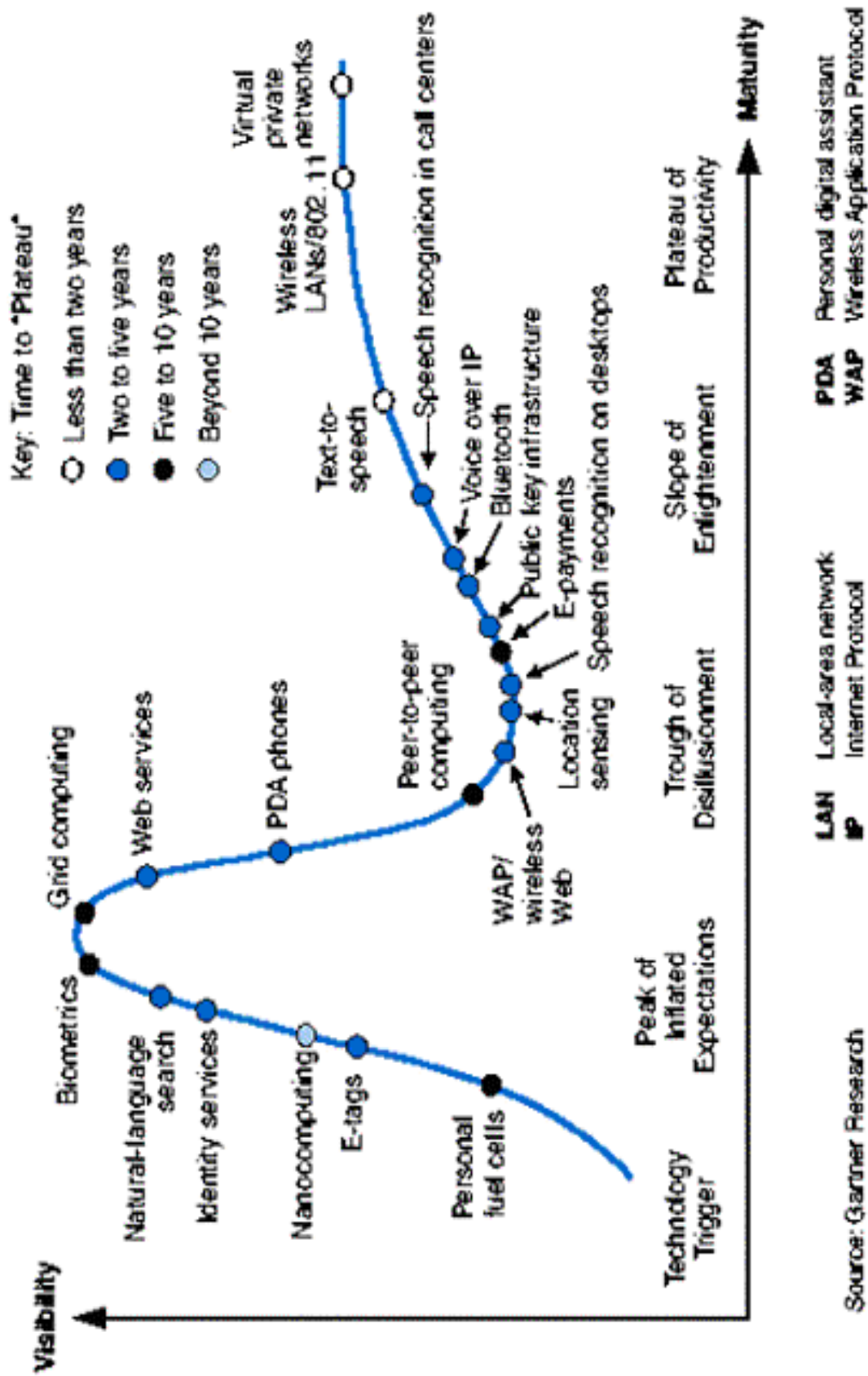


### Remote Procedure Call



### Web Services Remote Procedure Call

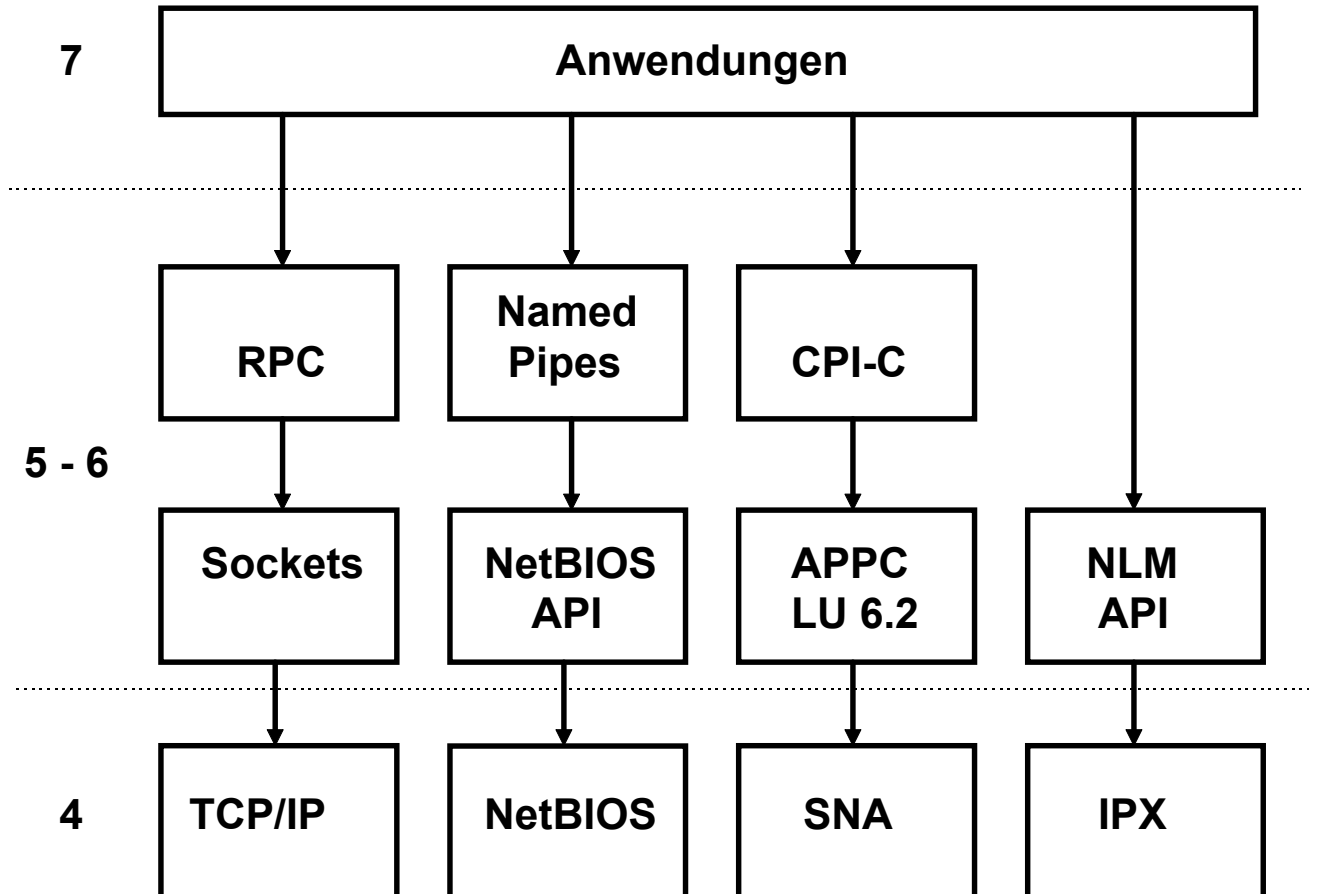




**SNA**

**System Network Architecture**

## Schicht



## Gebräuchliche Programmierschnittstellen (zahlreiche Querverbindungen)

weitere Protokolle: HTTP, 3270 (verwendet SNA)

Corba / IIOP, RMI, DCOM, verwenden Sockets und/oder RPC. Microsoft SMB verwendet Named Pipes.

CICS Distributed Program Link

# SNA-Vokabular

**SNA ist ein Session-orientiertes Protokoll. Eine Session wird zwischen zwei Knotenrechnern aufgebaut (open), Daten werden ausgetauscht, und die Session wird geschlossen (close).**

**Knotenrechner in einem SNA-Netzwerk werden LU's (Logical Units) genannt. Der ältere Typ LU 2 wurde ursprünglich für nicht-intelligente Terminals entwickelt, und eignet sich nur für die Kommunikation zwischen einem Arbeitsplatzrechner und einem zentralen Server. Es existiert eine RPC ähnliche Funktionalität. Unintelligente 3270 Terminals oder PC's mit einem 3270 Emulator verwenden die LU2 um mit einer OS/390 TSO oder CICS Session zu kommunizieren.**

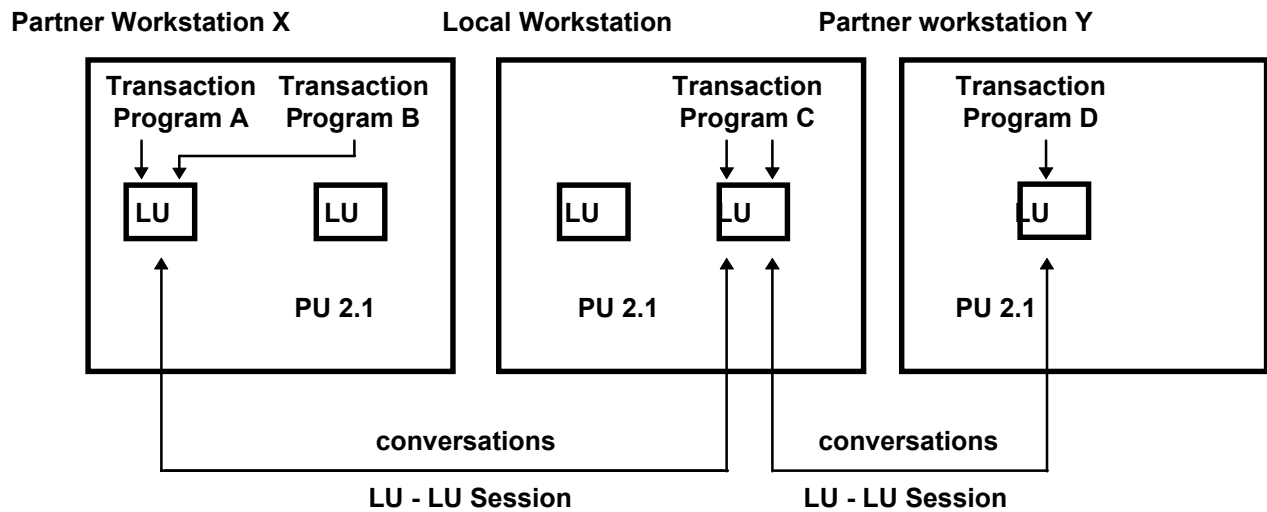
**Der derzeitig gängigste Typ wird als LU 6.2 bezeichnet. Er hat volle Peer-to-Peer Funktionalität in beiden Richtungen.**

**Die Hardware und Software eines Knotenrechners wird als PU (Physical Unit) bezeichnet. Eine PU steuert die Verbindungsleitungen zu einer anderen PU. Der derzeitig gängige Typ wird als PU 2.1 bezeichnet. Eine PU kann eine oder mehrere LU's unterstützen; die letzteren verwalten vor allem die „Sessions“**

**(Eine PU 4 ist direkt an einen /390-Rechner (über einen „Kanal“) angeschlossen und hat erweiterte Funktionen. Die auf einem /390-Rechner selbst ablaufende Netzsoftware, z.B. „VTAM“, wird als PU 5 bezeichnet)**

**Eine SNA „Session“ ist eine zuverlässige virtuelle Verbindung zwischen zwei LU's. Mehrere parallele Sessions zwischen zwei LU's sind möglich**

**Eine „APPC Conversation“ ist eine (von vielen) Transaktionen, die - unter Verwendung des APPC API - über eine Session mit einer anderen Transaktion kommuniziert. CICS IPC's können über APPC durchgeführt werden. In der Regel laufen viele Transaktionen der Reihe nach über eine Session**



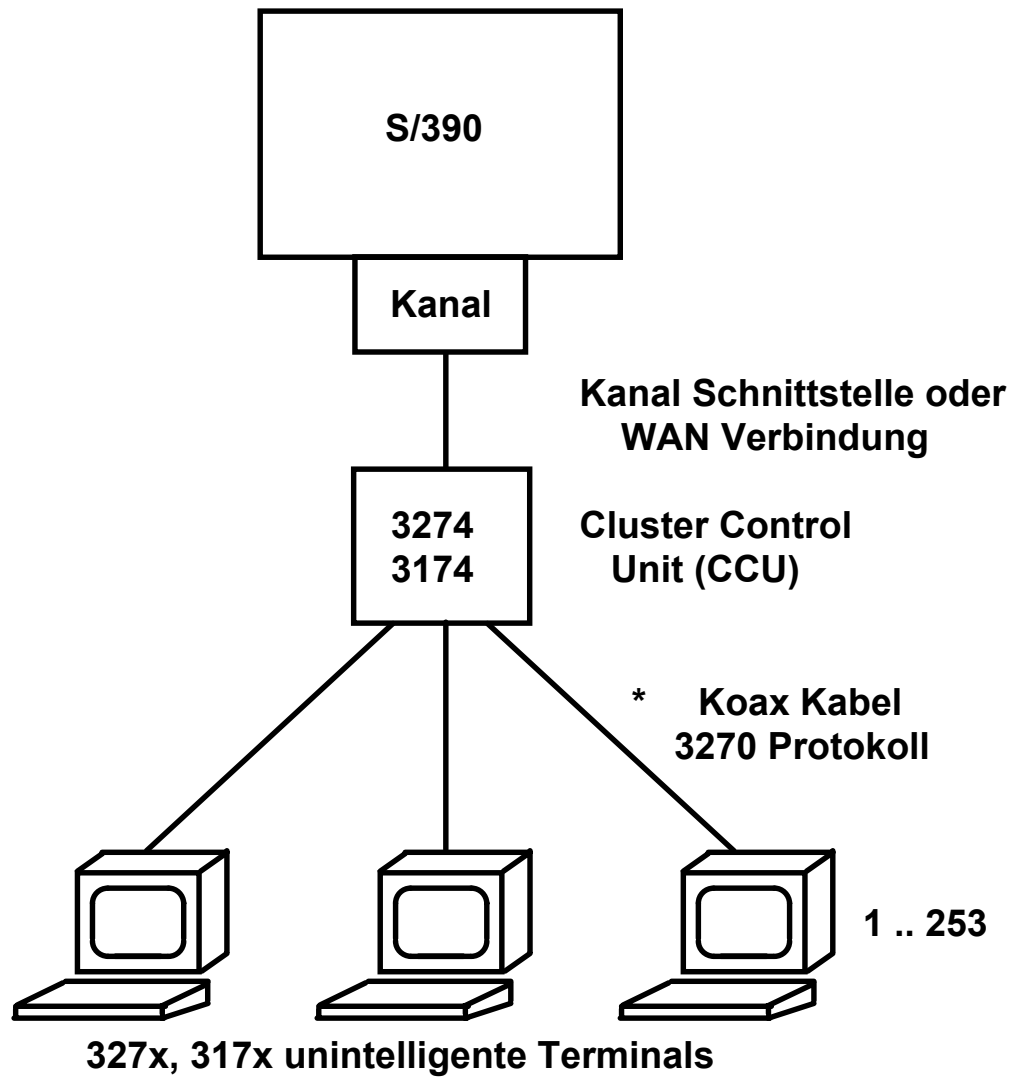
## Transaktionsverarbeitung über ein SNA Netz

Beispiel für eine Transaktionsverarbeitung über ein SNA Netz. Gezeigt ist eine lokale LU, die mit 2 Partner LU's kommuniziert. Der Zugriff zu den LU's erfolgt über die APPC API

TCP/UDP stellt TSAP's in der Form von Sockets zur Verfügung. Mit Hilfe der Socketschnittstelle können 2 Anwendungen auf 2 Rechnern miteinander kommunizieren

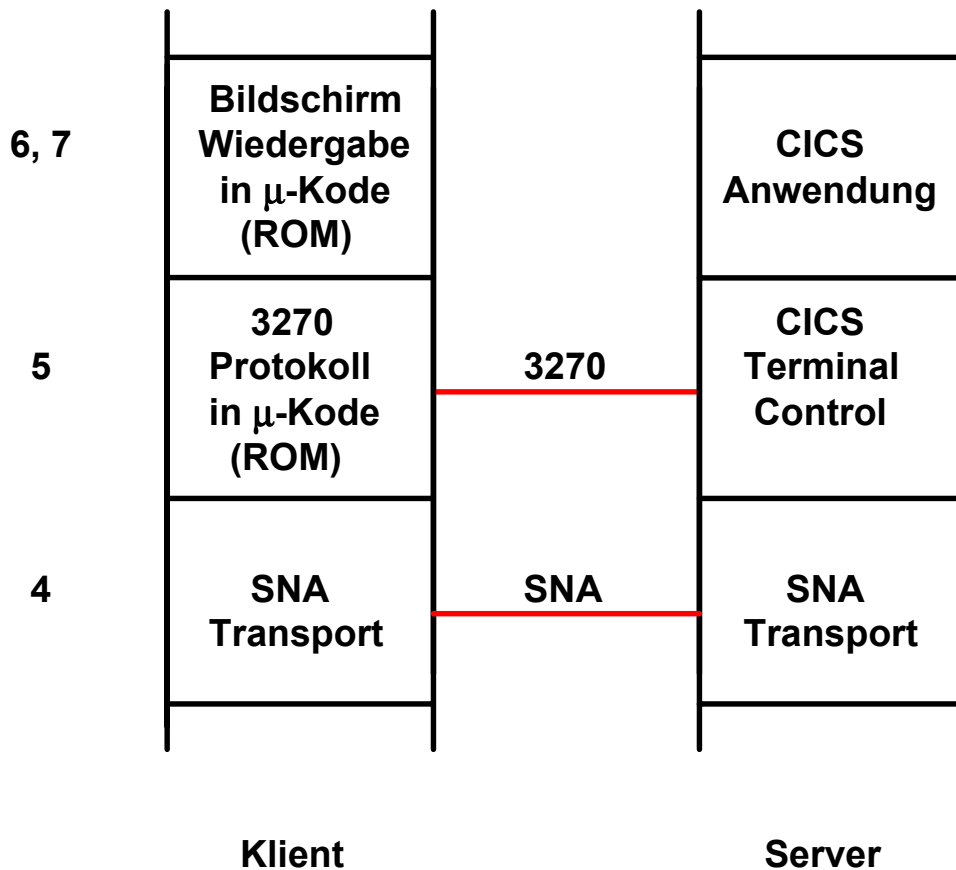
LU 6.2 stellt ähnliche TSAP's zur Verfügung. Mit Hilfe der APPC-Schnittstelle können 2 Anwendungen auf 2 Rechnern miteinander kommunizieren

„SNA-Sessions“-äquivalente Einrichtungen sind unter RPC und Named Pipes verfügbar

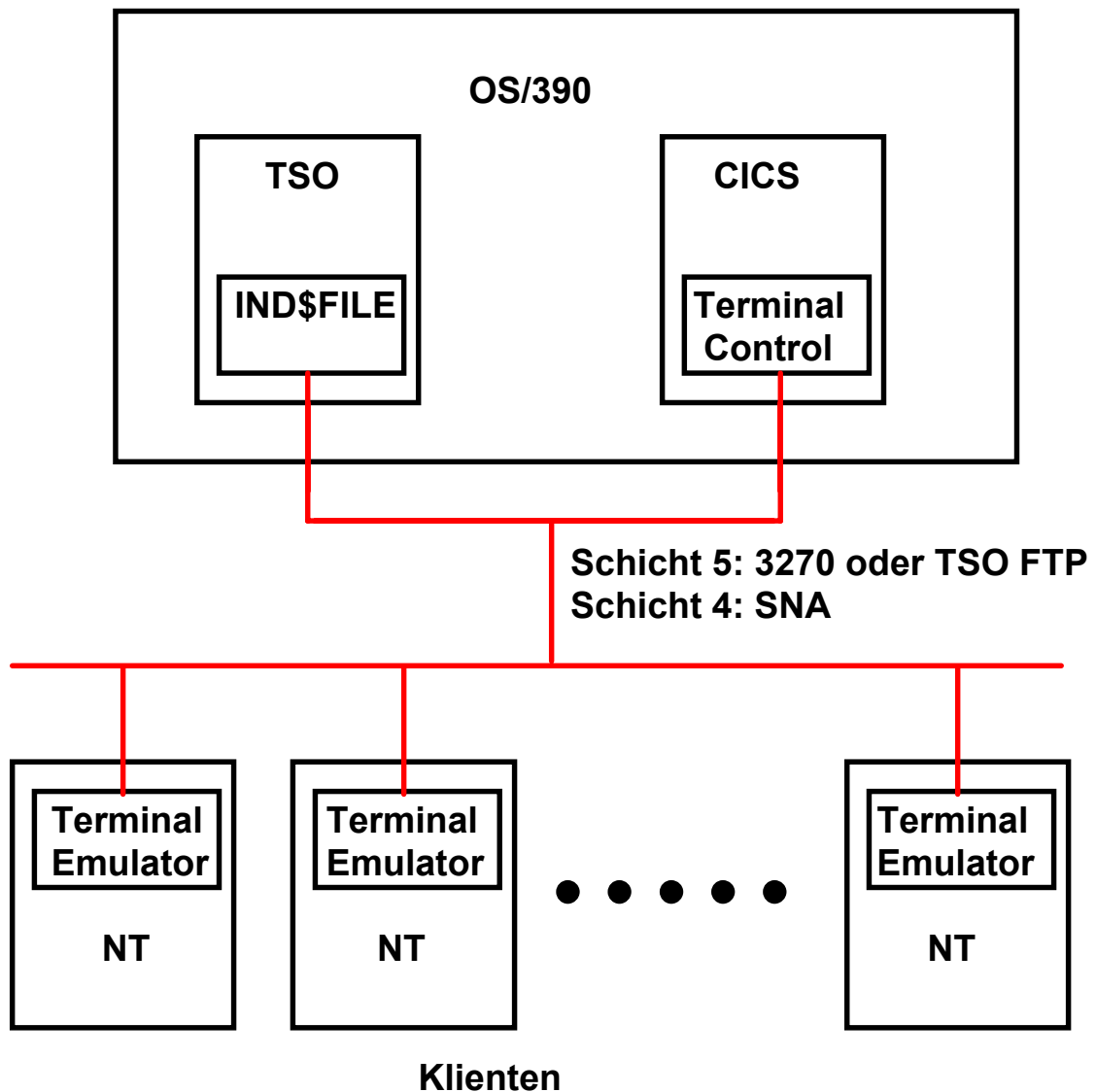


## Ursprüngliche 327x Konfiguration

## Schicht

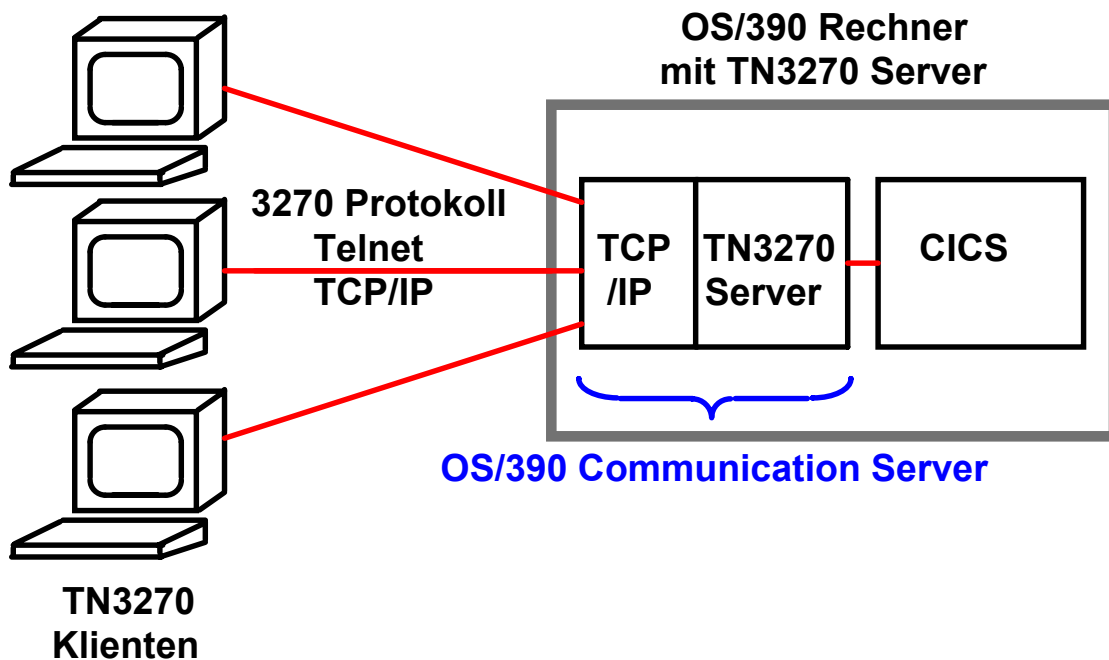


## CICS Client/Server Kommunikation ursprüngliche 327x Terminals



## CICS Client/Server Kommunikation über ein SNA Netz

**Windows Klienten mit Terminal Emulation  
und zusätzlicher File Transfer Einrichtung**



## TN3270 Protokoll

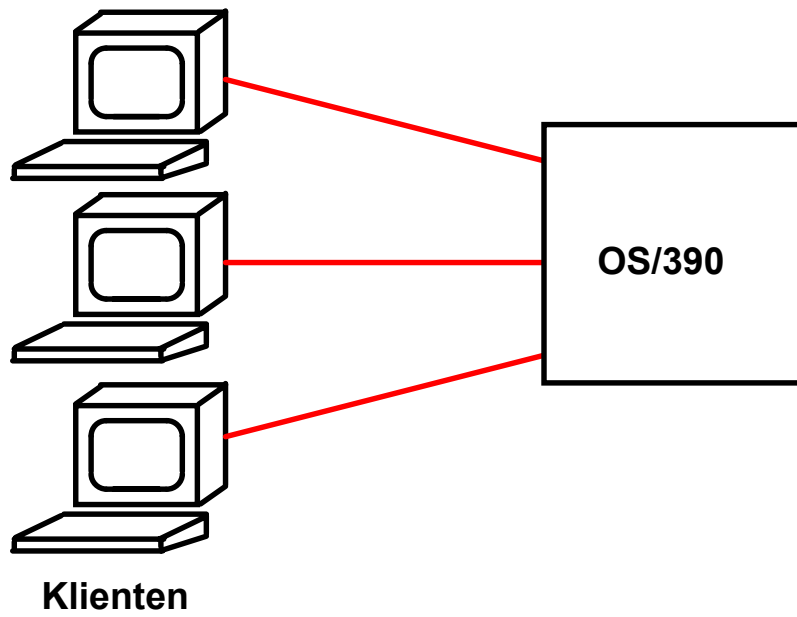
TN3270 Klienten sind über ein TCP/IP Netzwerk mit dem TN3270 Server verbunden.

Der TN3270 Server simuliert jeden angeschlossenen TN3270 Klienten als ein logisches SNA Terminal (LU Typ 2 für Bildschirme, LU Typ 1 und 3 für Drucker).

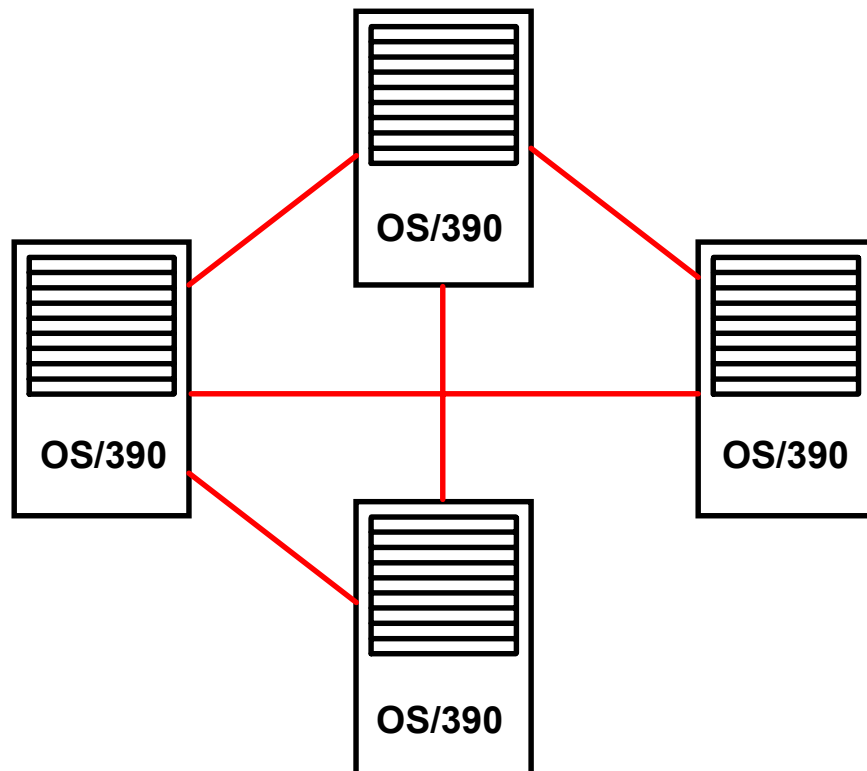
Zwischen dem TN3270 Server und der OS/390 Anwendung wird das SNA Protokoll benutzt.

Zwischen dem TN3270 Server und dem TN3270 Klienten wird das IP Protokoll benutzt.

TN3270e erweitert das ursprüngliches TN3270 Protokoll und ist in RFC 1647 spezifiziert.



**Client - Server Konfiguration**  
Verbindung über TN3270



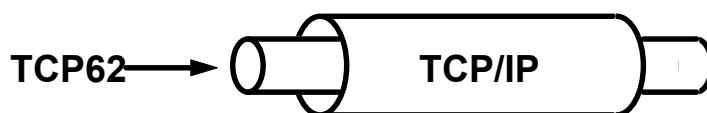
**Server - Server Konfiguration**  
Verbindung über TCP62

# AnyNet

**AnyNet ist eine OS/390 Komponente, die es SNA Anwendungen ermöglicht, über nicht-SNA Netze miteinander zu kommunizieren.**

**TCP62 ist eine AnyNet Komponente, die es zwei LU6.2 Einheiten ermöglicht, über TCPIP miteinander zu kommunizieren.**

**TCP/IP Pakete transportieren SNA Pakete als Nutzlast. Der Sender verpackt ein SNA Paket als TCP/IP Paket; der Empfänger entpackt es wieder.**



## **Parallelbetrieb von SNA und TCP/IP TN3270 und TCP62**

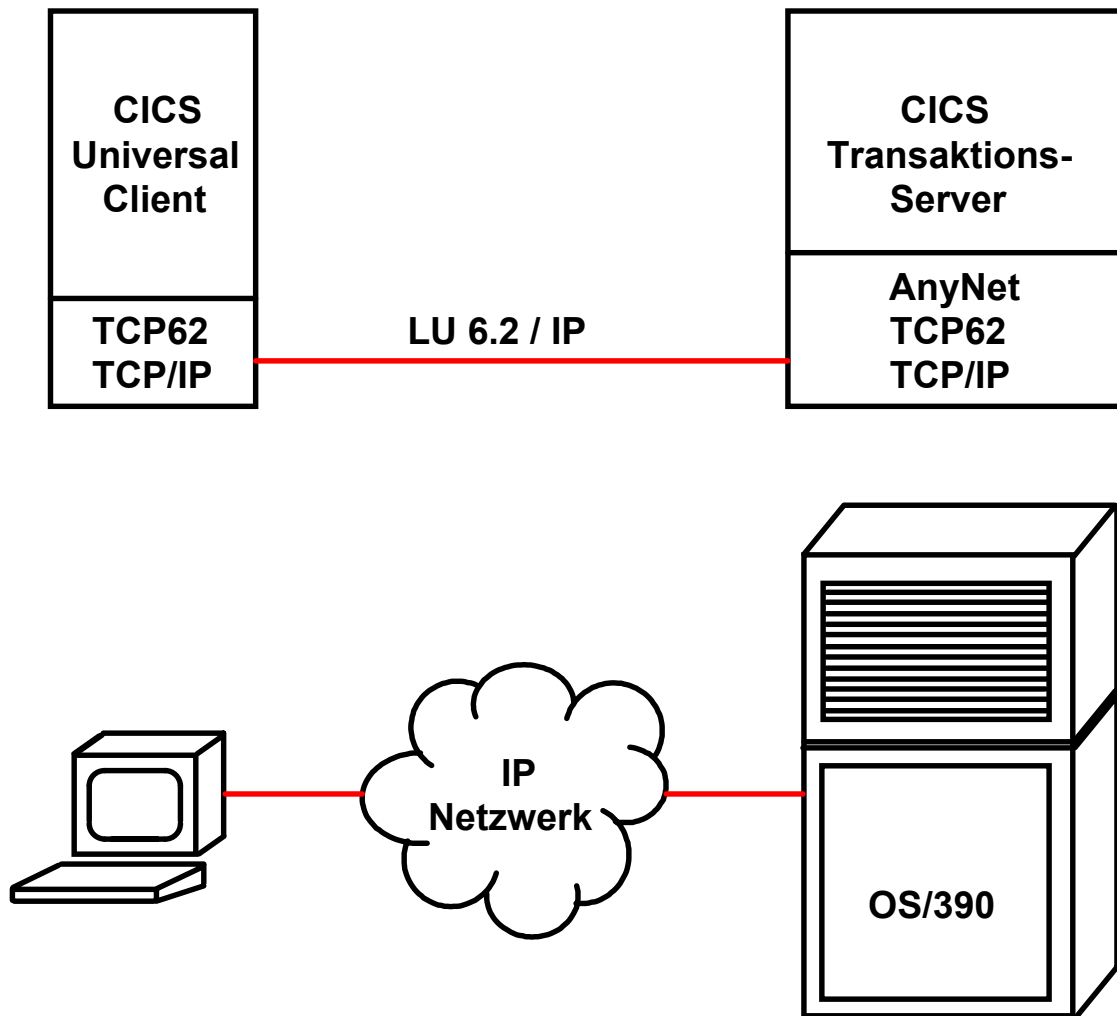
**CICS Klienten und Server kommunizieren miteinander über das SNA Netzwerkprotokoll.**

**Viele Wirtschaftsunternehmen benutzen intern nach wie vor SNA, stellen (sehr) langsam von SNA auf TCP/IP um. Für Internet Verbindungen ist TCP/IP erforderlich.**

**Lösung: SNA Protokolle über TCP/IP transportieren. Zwei unterschiedliche Implementierungen.**

**3270 Terminals (und 3270 PC-Emulatoren) verwenden SNA LU2. Die TCP/IP Implementierung erfolgt über TN3270 (LU 2 über TCP/IP)..**

**Verteilte CICS Systeme kommunizieren miteinander über „Distributed Program Link“ (häufig über die EPI Schnittstelle implementiert). DPL verwendet SNA LU6.2, erhöhter Funktionsumfang gegenüber LU2. Die TCP/IP Implementierung erfolgt über TCP62 (LU 6.2 über TCP/IP).**



## CICS Universal Client

1. The CICS Universal Client, using your CICSCLI.INI definitions, passes data to the AnyNet component on the workstation.
2. The AnyNet component on the client workstation uses the domain name suffix with the partner LU name to generate an Internet Protocol (IP) name. The IP address for this name is then determined either from the local IP hosts file or from a domain name server (DNS). Using standard TCP/IP flows, the data is shipped from the TCP/IP component on the workstation to TCP/IP on OS/390.
3. TCP/IP on OS/390 routes the data received from the workstation to the AnyNet component on OS/390.
4. The SNA over TCP/IP feature of AnyNet translates the inbound IP routing information to SNA routing information. The data is passed to VTAM and, in turn, to the CICS Transaction Server for OS/390.