Distributed Systems Architectures

Topics covered

- Multiprocessor architectures
- Client-server architectures
- Distributed object architectures
- Inter-organisational computing

Distributed systems

- Virtueel zijn alle grote computer-based systemen gekend als distributed systems.
- Informatie verwerking wordt verdeeld over verschillende computers ipv op één enkle machine.
- Distributed software engineering is daarom zeer belangrijk voor computer systemen op bedrijfsniveau.

Systeem types

- Personal systems die niet distributed zijn en die werden ontwikkeld om te werken op een personal computer of workstation.
- Embedded systems die lopen op één enkele processor of op een geïntegreerde groep van processors.
- Distributed systems waar de systeem software loopt op een geïntegreerde groep van samenwerkende processoren die gelinkt zijn via een netwerk.

Distributed system characteristics

- Resource sharing
- Openheid
  - Gebruik van hard- en software van verschillende fabricanten.
- Concurrency
  - Concurrente processing om de prestaties te verhogen.

Distributed system nadelen

- Complexiteit
  - Complexer dan gecentraliseerde systemen.
- Security
  - Gevoeliger voor indringers van buitenaf.
- Manageability
  - Het systeembeheer is complexer.
- Onvoorspelbaarheid
  - Onvoorstelbare responses afhankelijk van de systeem organisatie en het netwerk.

Distributed systems architectures

- Client-server architectures
  - Distributed services die worden opgeroepen door clients. Clients gebruiken services geleverd door servers.
- Distributed object architectures
  - Geen onderscheid tussen clients en servers. Om het even welk object op het systeem kan services leveren aan- of gebruiken van andere objecten.

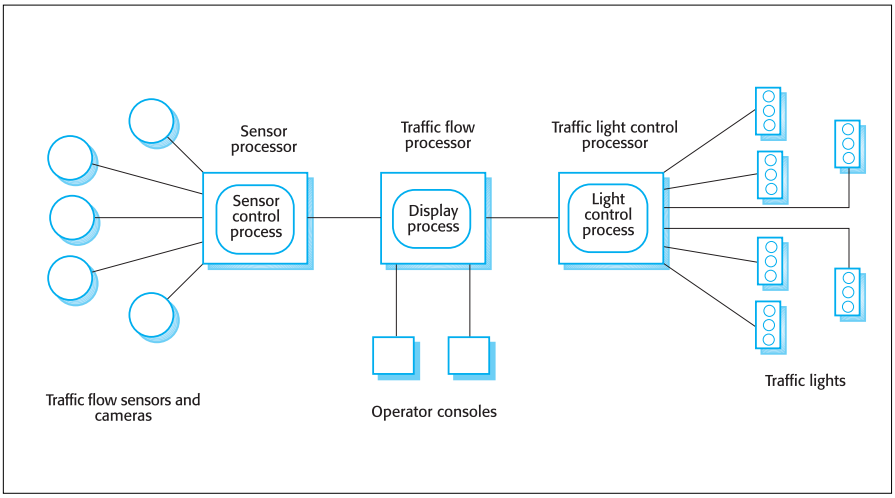
Middleware

- Software die het beheer en het onderhoud doet van de verschillende componenten van een distributed system. Het zit essentieel middenin het systeem.
- Voorbeelden:
  - Transaction processing monitors;
  - Data converters;
  - Communication controllers.

Multiprocessor architectures

- Het eenvoudigste distributed systeem model.
- Een systeem opgebouwd uit verschillende processen die al of niet worden uitgevoerd op verschillende processoren.
- Het Architectuur model van vele grote real-time systemen.
- De procesverdeling kan vooraf zijn vastgesteld of onder de controle van een dispatcher.

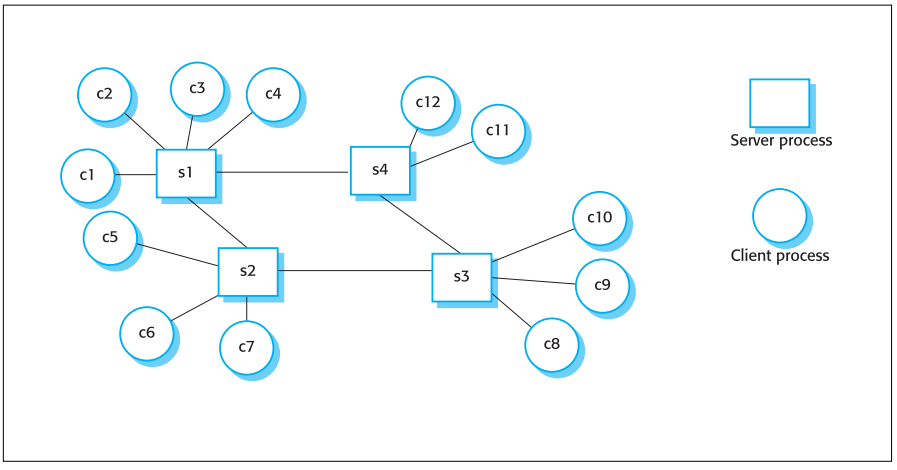
A multiprocessor traffic control system



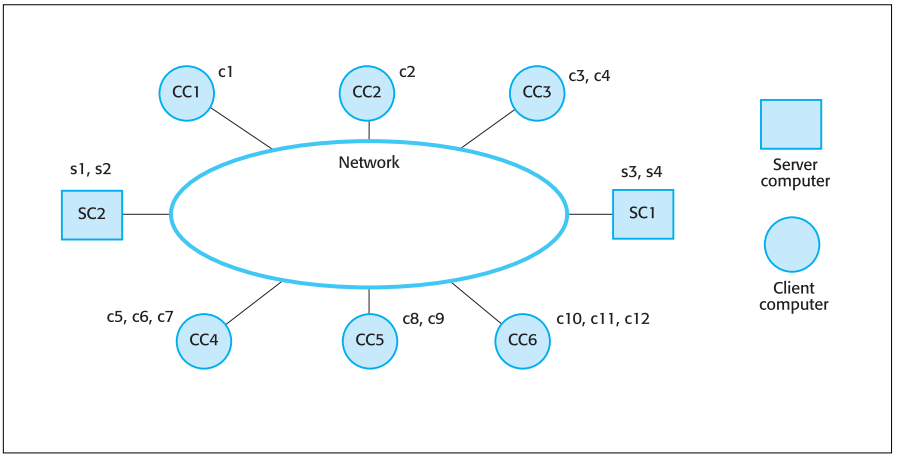
Client-server architecturen

- De toepassing is gemodelleerd als een set van services die worden geleverd door servers en een set van clients die deze services gebruiken.
- Clients kennen de servers mar de server hoeven de clients niet te kennen.
- Clients en servers zijn logische processen
- De mapping van processors naar processen is niet noodzakelijk 1 : 1.

A client-server system



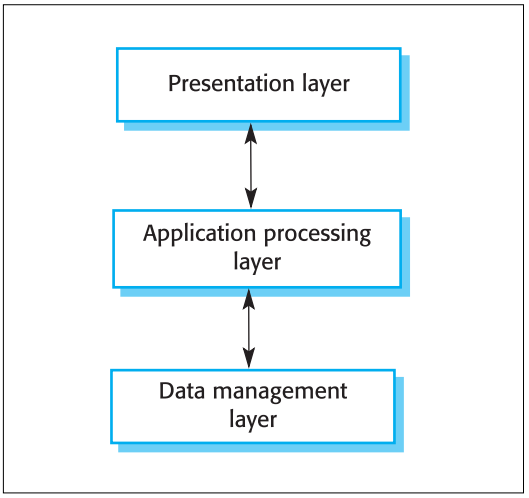
Computers in a C/S network



Layered application architecture

- Presentation layer
  - Betrokken bij het voorstellen van de resultaten van een bewerking aan system users en met het verzamelen van user inputs.
- Application processing layer
  - Voorziet de toepassing van de specifieke functionaliteit b.v., in een banksysteem, functies zoals het openen en sluiten van rekeningen, etc.
- Data management layer
  - Betrokken bij het beheer van de systeem databases.

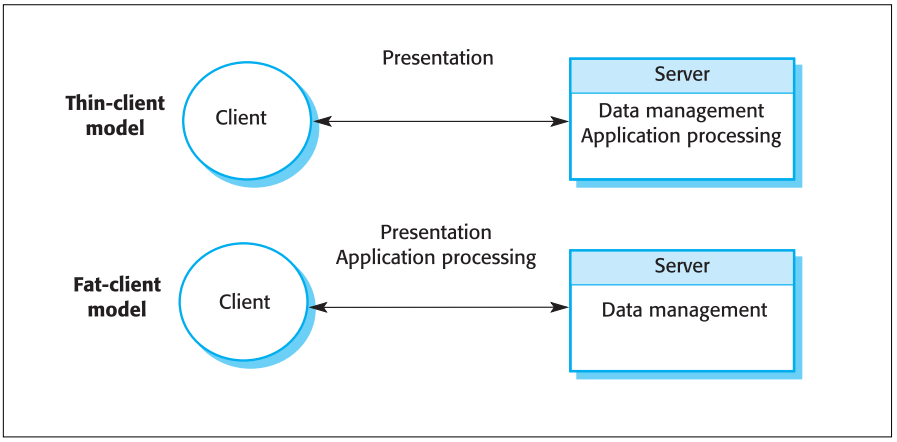
Application layers



Thin and fat clients

- Thin-client model
  - In een thin-client model, wordt de volledige verwerking van de toepassing en alle data management uitgevoerd op de server. De client is alleen verantwoordelijk voor het uitvoeren van de presentation software.
- Fat-client model
  - In dit model, is de server alleen verantwoordelijk voor het data management. De software op de client implementeert de applicatie logica en de interacties met de systeemgebruiker.

Thin and fat clients



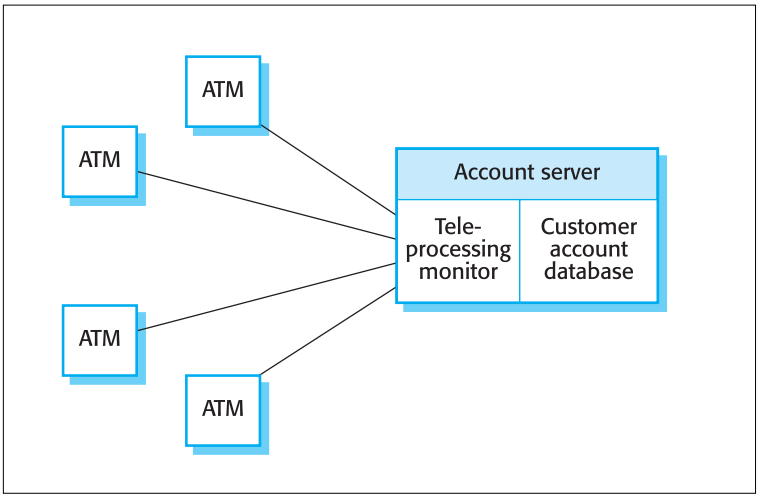
Thin client model

- Worden gebruikt wanneer bestaande systemen worden gemigreerd naar client server architecturen.
  - Het bestaande systeem handelt als een server met een grafische interface geïmplementeerd op een client.
- Het grote nadeel is een zware druk op de server en het netwerk.

Fat client model

- Meer verwerking is gedelegeerd naar de client omdat de verwerking lokaal gebeurt.
- Meest topasbaar voor nieuwe C/S systemen waar de mogelijkheden van het client systeem vooraf zijn gekend.
- Complexer dan het thin client model zeker voor het beheer. Nieuwe versies van de toepassing moeten op de clients geïnstalleerd worden.

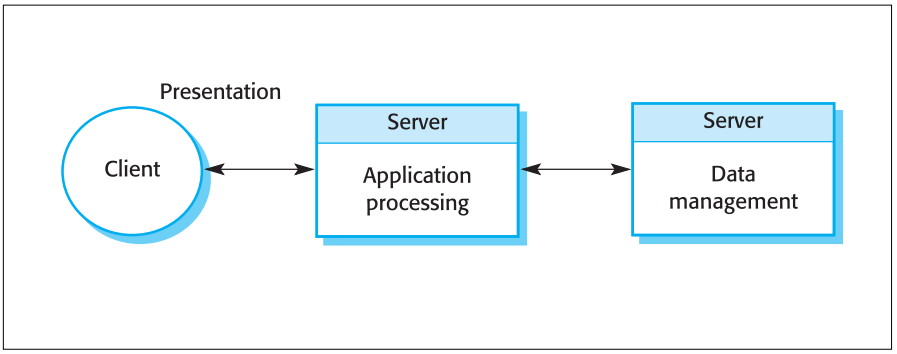
A client-server ATM system



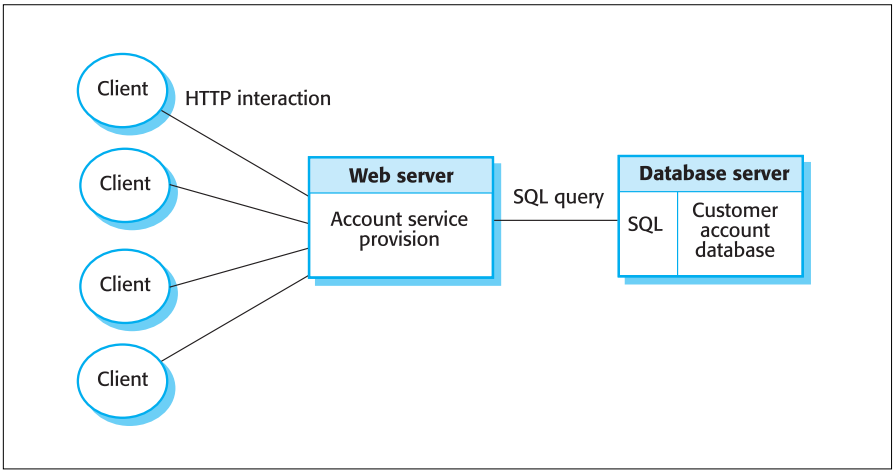
Three-tier architectures

- In een three-tier architectuur, kan elk van de application architecture layers uitgevoerd worden op een afzonderlijke processor.
- Presteert beter dan een thin-client en is gemakkelijker te beheren dan een fat-client benadering.

A 3-tier C/S architecture



An internet banking system



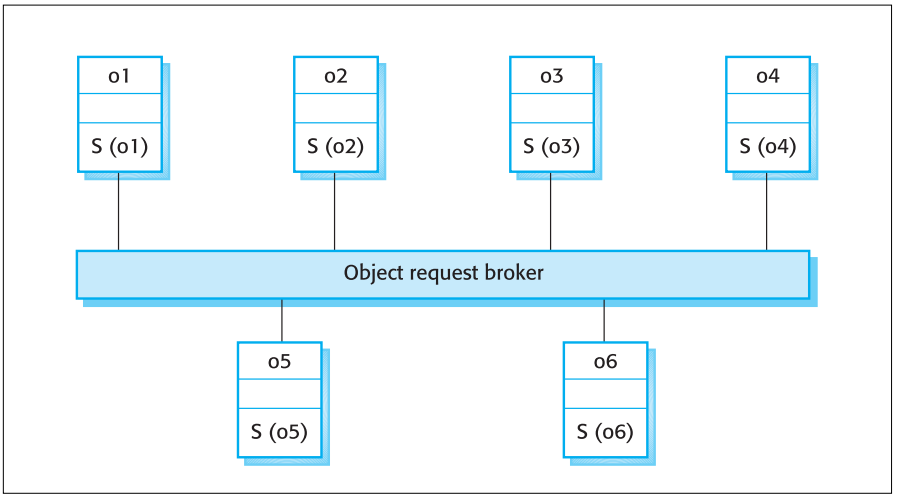
Use of C/S architectures

Architecture	Applications
Two-tier C/S architecture with thin clients	Legacy system applications where separating application processing and data management is impractical. Computationally-intensive applications such as compilers with little or no data management. Data-intensive applications (browsing and querying) with little or no application processing.
Two-tier C/S architecture with fat clients	Applications where application processing is provided by off-the-shelf software (e.g. Microsoft Excel) on the client. Applications where computationally-intensive processing of data (e.g. data visualisation) is required. Applications with relatively stable end-user functionality used in an environment with well-established system management.
Three-tier or multi-tier C/S architecture	Large scale applications with hundreds or thousands of clients Applications where both the data and the application are volatile. Applications where data from multiple sources are integrated.

Distributed object architectures

- Geen onderscheid tussen clients en servers.
- Elke gedistribuerde entiteit is een object dat services voorziet voor andere objecten en services ontvangt van andere objecten.
- Object communicatie is over een middleware system een object request broker gnoemd.
- Gedistribueerde object architecturen zijn complexer om te ontwikkelen dan C/S systemen.

Distributed object architecture



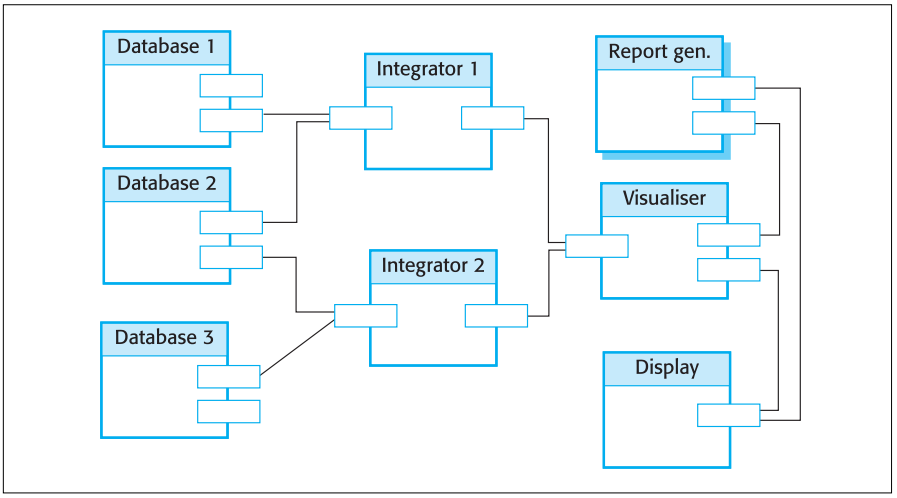
Voordelen van de distributed object architecture

- Het laat de systeem designer toe om beslissingen van waar en hoe services moeten worden voorzien, uit te stellen.
- Een open systeem, zodat nieuwe resources kunnen toegevoegd woden waar nodig.
- Het is een flexibel en uitbreidbaar systeem.
- Kan dynamisch geherconfigureerd worden met objecten die migreren over het netwerk, waar nodig.

Uses of distributed object architecture

- Het is een logisch model dat ons toelaat om het systeem te structureren en te organiseren In dit geval denken we voor het leveren van functionaliteit van de toepassing enkel in termen van services en combinaties van services.
- Als een flexiebele benadering van de implementatie van client-server systemen. Het logisch systeemmodel is een client-server model maar zowel clients als servers worden gerealiseerd als distributed objects die communiceren over een gemeenschappelijk communication framework.

A data mining system



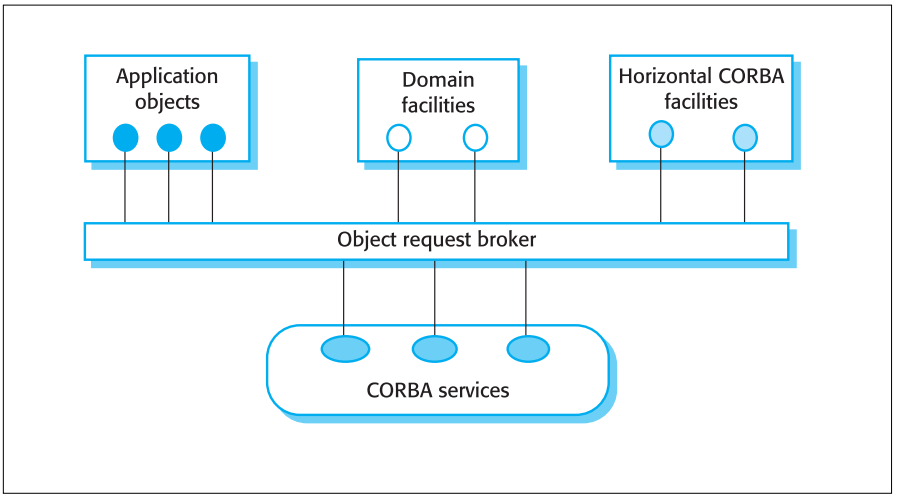
Data mining system

- Het logisch model van het system is er niet een van service provision omdat er afzonderlijke data management services zijn.
- Het aantal databases kan worden verhoogd zonder onderbreking van het systeem.
- Nieuwe types van relaties kunnen worden omschreven door het toevoegen van nieuwe integrator objects.

CORBA

- CORBA is een internationale standaard voor een Object Request Broker - middleware voor het beheer van communicaties btussen distributed objects.
- Middleware voor distributed computing is vereist op 2 levels:
  - Op logisch communication level: de middleware laat objecten op verschillende computers toe om data uit te wisselen en control informatie;
  - Op component level: de middleware voorziet een basis voor het ontwikkelen van compatibele componenten. CORBA component standaarden zijn gedefinieerd.

CORBA application structure



Application structure

- Application objecten.
- Standaard objecten, gedefinieerd door de OMG, voor een specifiek domein b.v. insurance.
- Fundamentele CORBA services zoals directory en security management.
- Horizontaal faciliteiten zoals user interface faciliteiten.

CORBA standaards

- Een object model voor applicatie objecten
  - Een CORBA object is een encapsulation van properties en methods met een goed gedefinieerde, language-neutral interface gedefinieerd in een IDL (interface definition language).
- Een object request broker die aanvragen voor object services beheert.
- Een set van algemene object services bruikbaar voor vele distributed applications.
- Een set van gemeenschappelijke componenten gebouwd bovenop deze services.

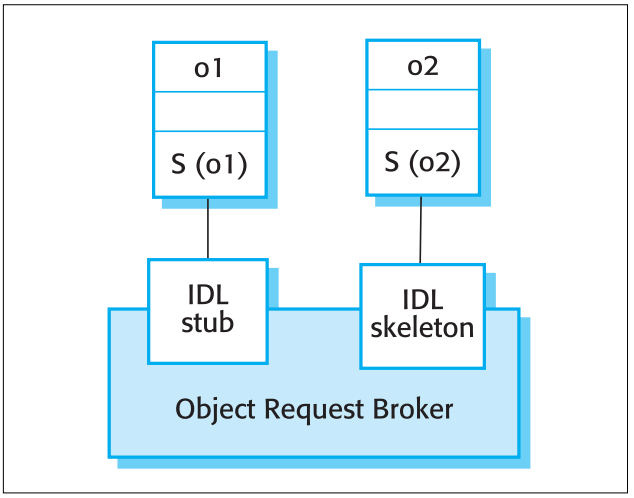
CORBA objects

- CORBA objecten zijn, in principe, vergelijkbaar met objecten in C++ en Java.
- Ze MOETEN een afzonderlijke interface definitie hebben die wordt uitgedrukt gebruikmakend van een gemeenschappelijke taal (IDL) gelijkend op C++.
- Er is een mapping van deze IDL naar programmeeertalen (C++, Java, etc.).
- Daarom kunnen objecten, geschreven in verschillende talen communiceren met mekaar.

Object request broker (ORB)

- De ORB verwerkt object communications. Het kent alle objecten in het systeem en hun interfaces.
- Gebruikmakend van een ORB, bindt het calling object zich met een IDL stub die de interface definieert van het opgeroepen object.
- Het oproepen van deze stub resulteert in calls naar de ORB die vervolgens het vereiste object oproept via een gekend IDL skeleton die de interface linkt aan de service implementation.

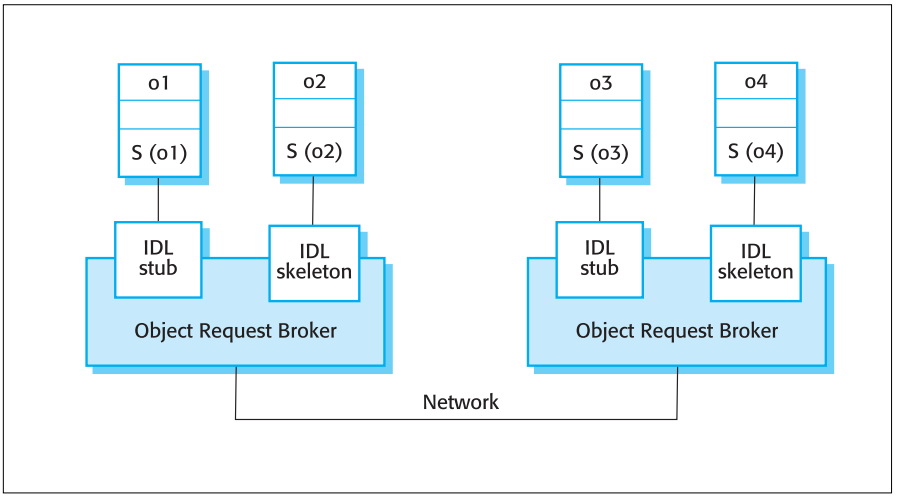
ORB-based object communications



Inter-ORB communications

- ORBs zijn gewoonlijk geen afzonderlijke programma's, maar behoren tot een set van objecten in een library die worden gelinkt aan een toepassing wanneer ze wordt ontwikkeld.
- ORBs verwerken communications tussen objects die worden uitgevoerd op dezelfde mahine.
- Meerdere ORBS kunnen beschikbaar zijn en iedere computer in een distributed system heeft zijn eigen ORB.
- Inter-ORB communications worden gebruikt voor distributed object calls.

Inter-ORB communications



CORBA services

- Naming en trading services
  - Laten toe aan objecten om te refereren naar andere objectn op het network.
- Notification services
  - Deze laten toe aan objecten om andere objecten op de hoogte te brengen van een optredend event.
- Transaction services
  - Deze ondersteunen atomaire transacties en rollback bij een fout.

Inter-organisational computing

- Om praktische- en om veiligheidsredenen, zijn de meeste gedistribueerde systemen geïmplementeerd op enterprise level.
- Nieuwere modellen van distributed computing werden ontworpen om informatiesystemen tussen bedrijven te ondersteunen. In dit geval zijn verschillende nodes gelokaliseerd in verchillende organisaties.

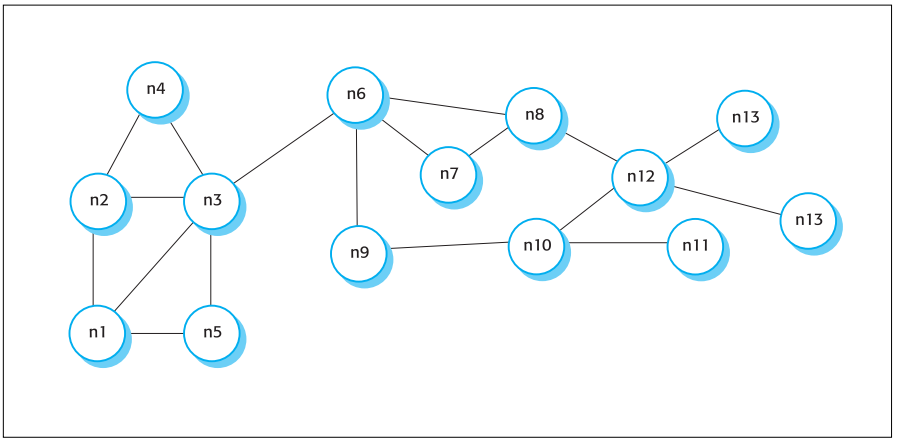
Peer-to-peer architectures

- Peer to peer (p2p) systemen zijn gedecentraliseerde systemen waar bewerkingen kunnen uitgevoerd worden door elke node in het netwerk.
- Het totale systeem is ontworpn om voordeel te halen uit de werkkracht en opslag van een groot aantal computers over een netwerk.
- De meeste p2p systemen waren ooit personal systems maar het gebruik van deze technologie stijgt voortdurend.

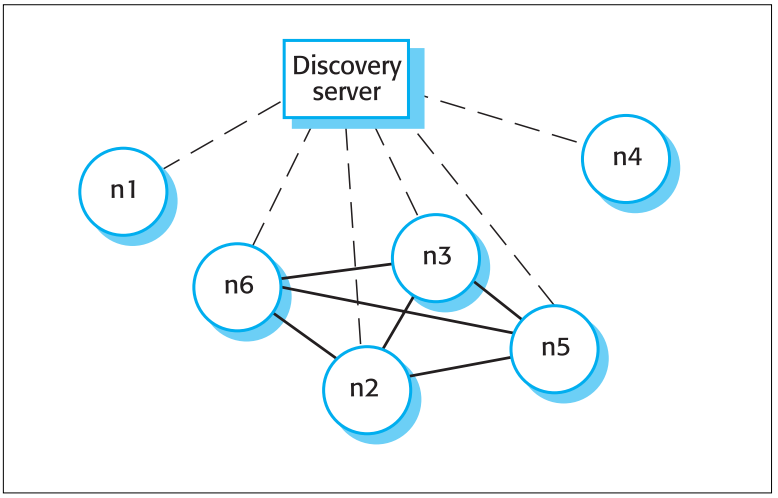
P2p architectural models

- De logische netwerk architectuur
  - Decentralised architectures;
  - Semi-centralised architectures.
- Applicatie architectuur
  - De generieke organisatie van componenten waaruit een p2p applicatie bestaat.
- De nadruk ligt op netwerk architecturen.

Decentralised p2p architecture



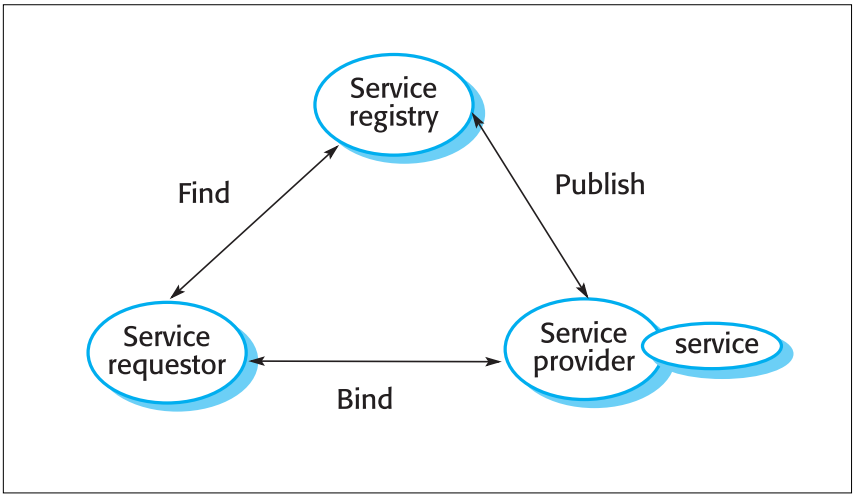
Semi-centralised p2p architecture



Service-oriented architectures

- Gesteund op extern aangeboden services (web services).
- Een web service is een standaard benadering om een herbruikbare component beschikbaar te stellen en toegangkelijk over het WWW

Web services



Services standards

- Services zijn gebaseerd op aanvaarde, XML-based standaarden en kunnen bijgevolg ook voor elk platform worden gebruikt en geschreven in elke prorammeertaal
- Key standards
  - SOAP - Simple Object Access Protocol;
  - WSDL - Web Services Description Language;
  - UDDI - Universal Description, Discovery and Integration.