Slides for Chapter 2: Architectural Models

From **Coulouris, Dollimore, Kindberg and Blair**

Distributed Systems: Concepts and Design

Types of Models for Distributed Systems

Levels of models:

- Physical models: Describe hardware components and devices
- Architectural models: Describe computation and communication tasks
- Fundamental models: Describe an abstract perspective

Fundamental models can examine different aspects of a distributed system:

- Interaction models
- Failure models
- Security models
- etc.

Physical Models

Early distributed systems (1970s, 1980s):

- Use local networks (e.g. ethernet)
- Small number of nodes (10 – 100)
- Two-tier client server (printer servers, file servers)

Internet-scale distributed systems (1990s):

- Internet is a network of networks
- Nodes relatively static

More recent distributed systems (2000s):

- Mobile nodes, embedded nodes, pools of nodes

Figure 2.1
Generations of distributed systems

Distributed systems:	*Early*	*Internet-scale*	*Contemporary*
Scale	Small	Large	Ultra-large
Heterogeneity	Limited (typically relatively homogenous configurations)	Significant in terms of platforms, languages and middleware	Added dimensions introduced including radically different styles of architecture
Openness	Not a priority	Significant priority with range of standards introduced	Major research challenge with existing standards not yet able to embrace complex systems
Quality of service	In its infancy	Significant priority with range of services introduced	Major research challenge with existing services not yet able to embrace complex systems

Architectural Models: Elements

Communicating elements or entities:

- Processes communicating via inter-process communication
- May be implemented as threads
- Primitive nodes such as sensors

Programming perspective of communicating elements:

- Objects accessed via interfaces
- Components are similar to objects but explicitly describe dependencies (contracts) with other components
- Web services are intrinsically integrated into WWW using standards developed byW3C

Architectural Models: Communication Paradigms

Interprocess communication:

- Low level message-passing primitives (e.g. socket programming)

Remote invocation – several techniques for two-way exchange:

- Request-reply protocols
- Remote procedure calls (RPC)
- Remote method invocation (RMI)

Direct communication:

- Senders and receivers
- Both aware of one another

Architectural Models: Communication Paradigms

Indirect communication:

- Through a third party
- Group communication – set of recipients
- Publish-subscribe systems – producers (writers) and consumers (readers)
- Message queues – intermediate queues store messages
- Tuple spaces – intermediate persistent store keeps data
- Distributed shared memory (DSM)

Figure 2.2
Communicating entities and communication paradigms

Communicating entities (what is communicating)		*Communication paradigms (how they communicate)*		
System-oriented entities	*Problem-oriented entities*	*Interprocess communication*	*Remote invocation*	*Indirect communication*
Nodes	Objects	Message passing	Request-reply	Group communication
Processes	Components	Sockets	RPC	Publish-subscribe
	Web services	Multicast	RMI	Message queues
				Tuple spaces
				DSM

Architectural Models: Process Roles

Client-server:

- Different roles
- Client processes interact with server processes
- A server may be client to another server
- Examples of servers: DNS, search engines, e-commerce sites

Architectural Models: Process Roles

Peer-to-peer:

- Processes/nodes have similar roles
- No distinction between client and server
- Exploits all available resources to accomplish tasks – potential to scale better than client-server model
- Examples: Napster (music sharing), BitTorrent (file sharing)

Figure 2.4a
Peer-to-peer architecture

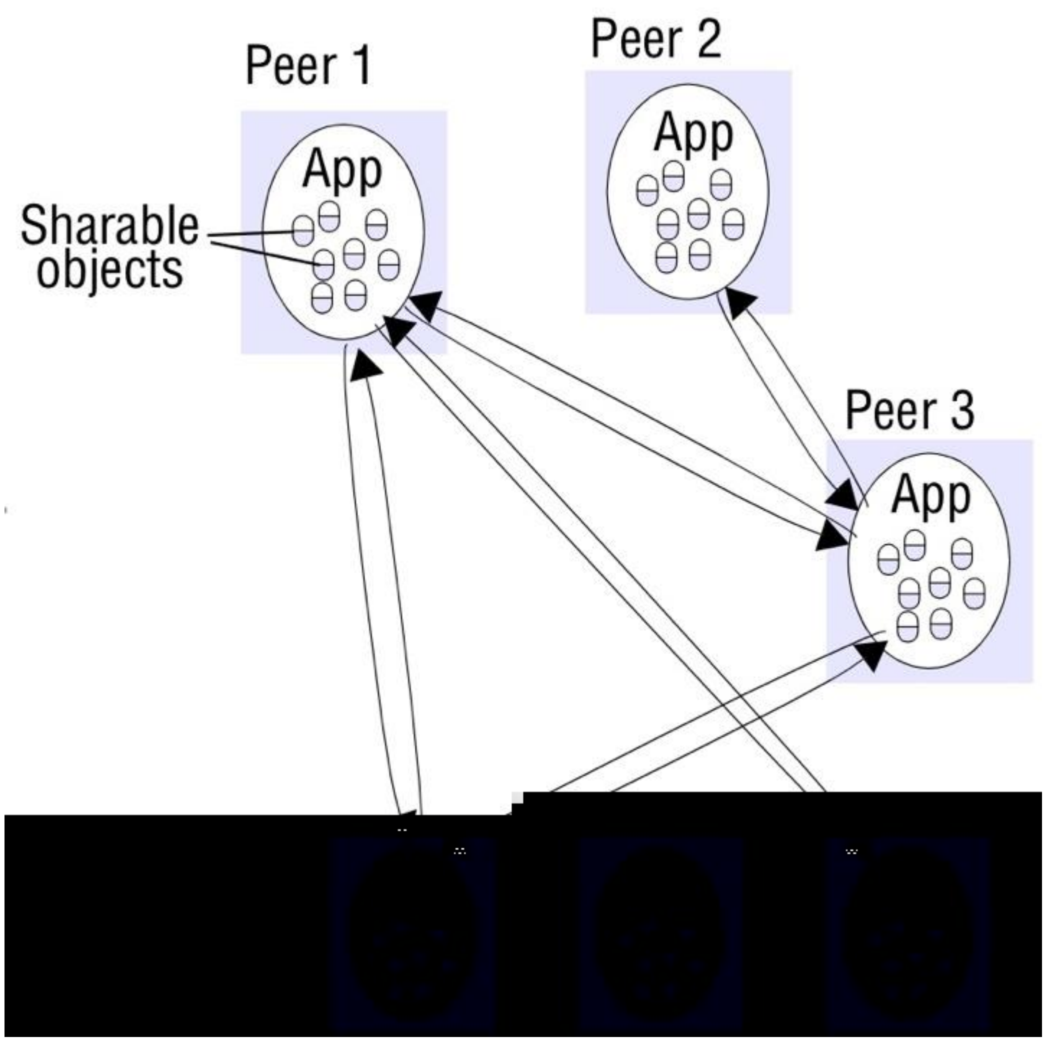
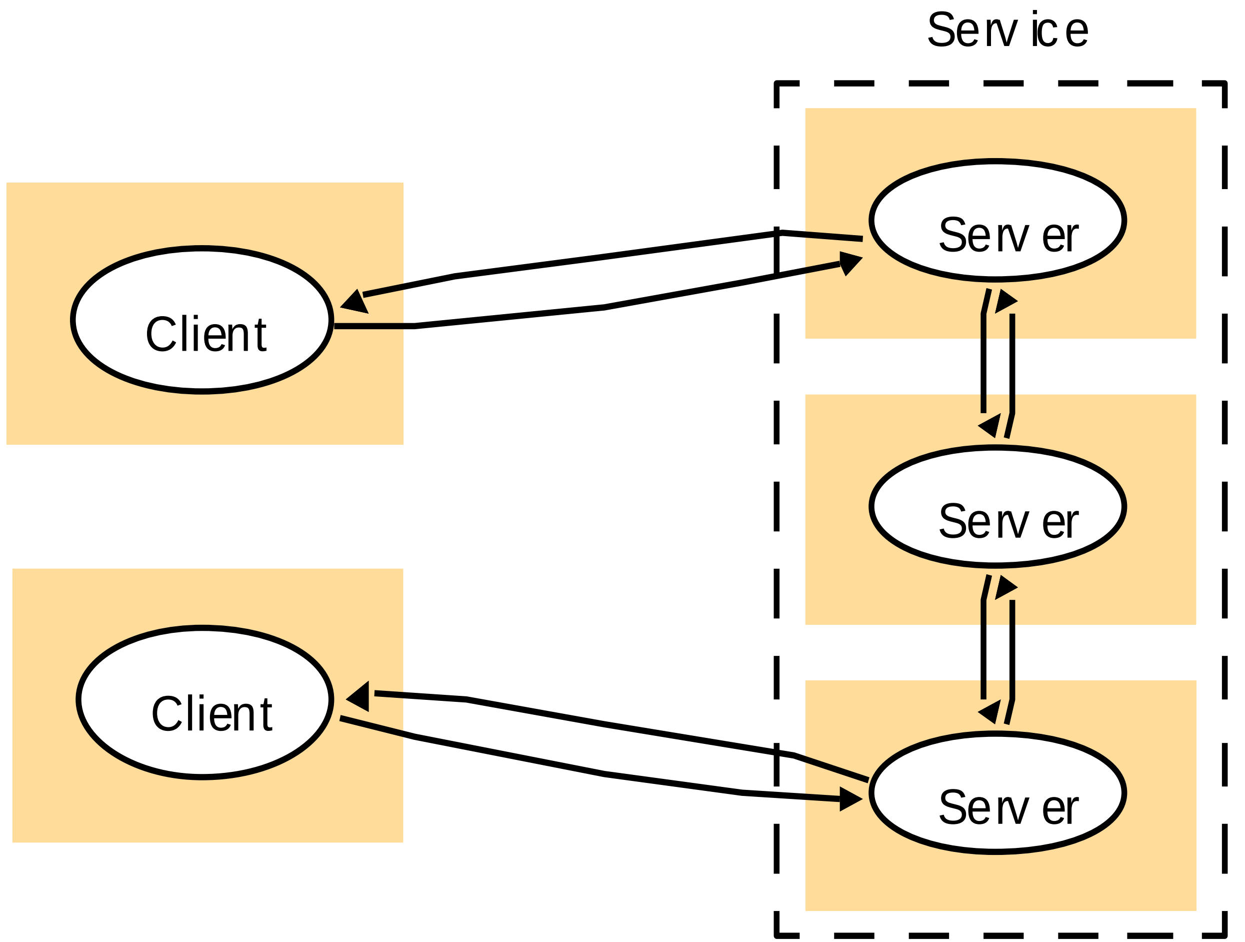


Figure 2.4b
A service provided by multiple servers



Architectural Models: Placement

- How to determine the nodes where client/server processes are placed
- Application dependent

Possible placement strategies

- Mapping of services to multiple servers
- Caching – web browsers
- Mobile code – applets downloaded from server
- Mobile agents – code and data move to various nodes
- Latter two pose security threats

Figure 2.5
Web proxy server

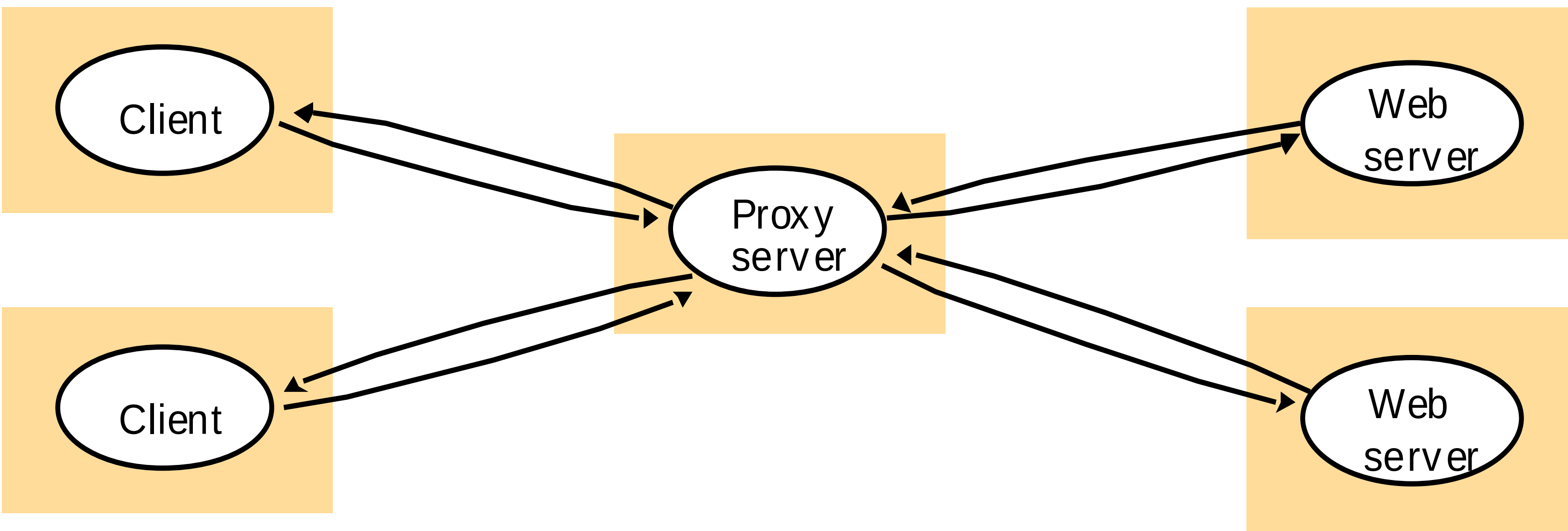
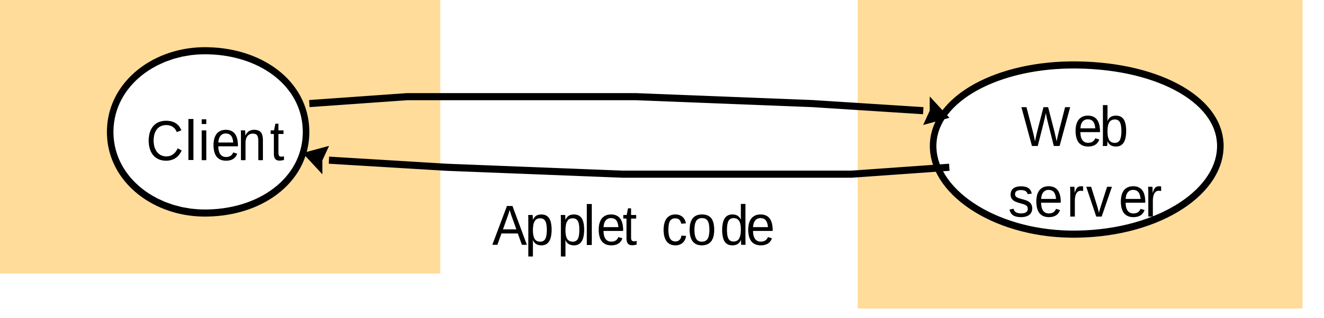


Figure 2.6
Web applets

a) client request results in the downloading of applet code



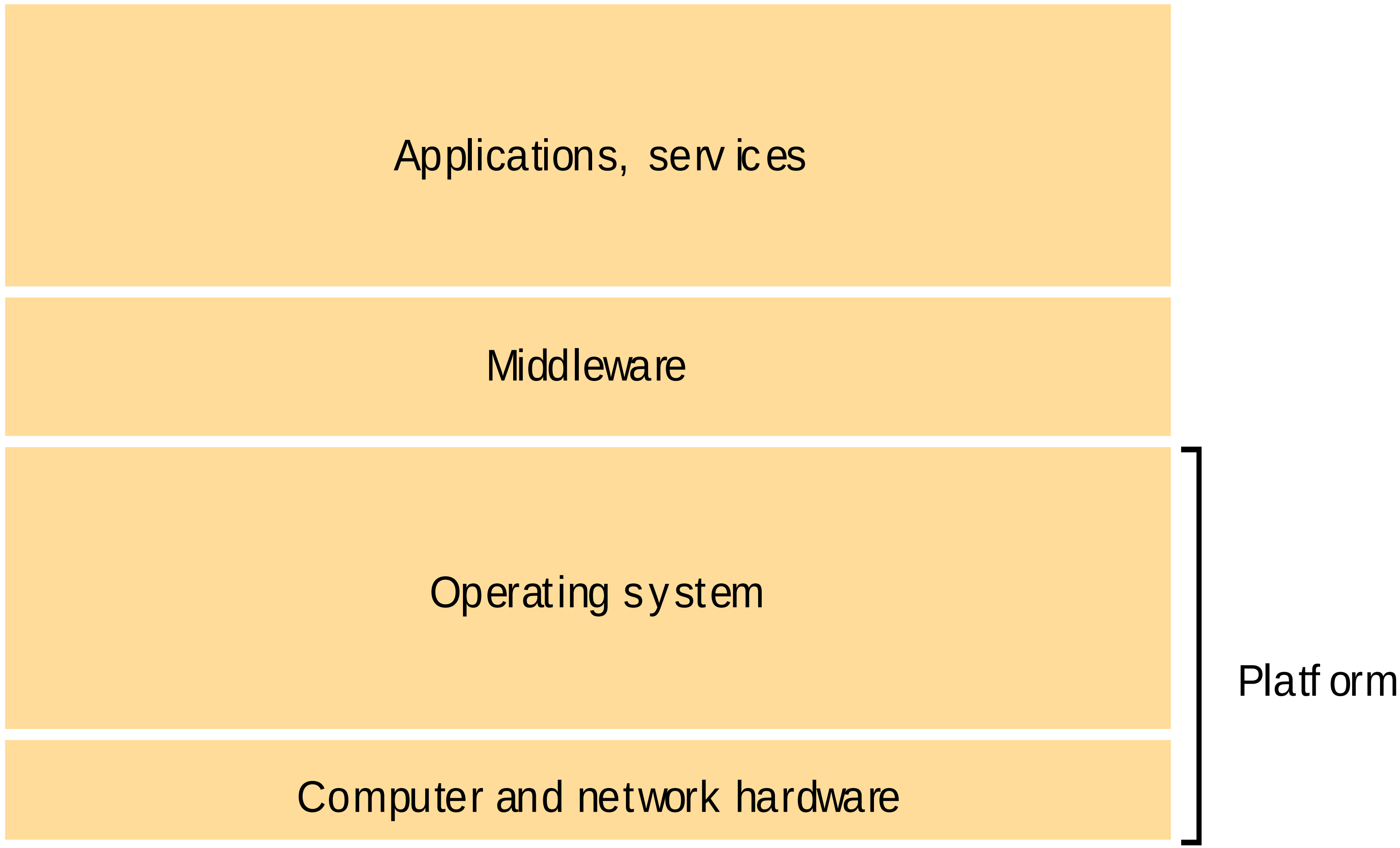
b) client interacts with the applet

Architectural Models: Architectural patterns

Layered Architectures:

- Each layer uses services provided by lower layer
- Vertical organization of services
- Complex services can be organized into layers
- Platform layers: OS, hardware, and network
- Middleware layer: software to mask heterogeneity in platform layer

Figure 2.7
Software and hardware service layers in distributed systems



Architectural Models: Architectural patterns

Tiered Architectures:

- Can utilize layering at a higher level
- Two-tier and Three-tier are common examples
- Two-tier: Client (PC, mobile device) and Server
- Three-tier: Presentation, application, database
- Can generalize to n-tiers
- Tiered architectures can use AJAX (Asynchronous Javascript and XML) to selectively update web pages from databases

Figure 2.8
Two-tier and three-tier architectures

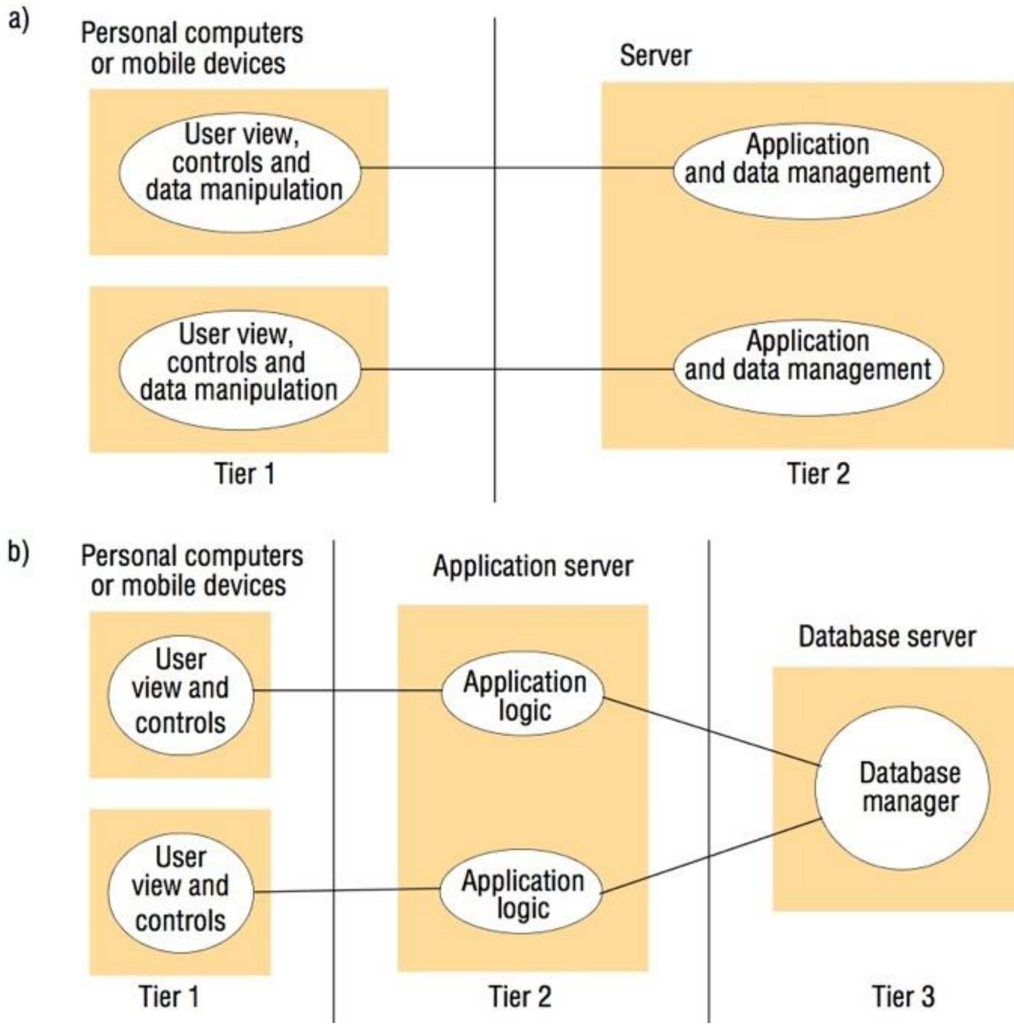


Figure 2.9
AJAX example: soccer score updates

```
new Ajax.Request('scores.php?
                game=Arsenal:Liverpool',
               {onSuccess: updateScore});
function updateScore(request) {
.....
```

(*request* contains the state of the Ajax request including the returned result.

The result is parsed to obtain some text giving the score, which is used to update the relevant portion of the current page.)

```
.....
}
```

Architectural Models: Architectural patterns

Thin Clients:

- Reduces demand on client devices
- Services executed on cloud instead of at client node
- Example: VNC (virtual network computing)

Other patterns:

- Proxy – located at local address to represent remote object
- Brokerage services – match requested service to a server that offers service)
- Reflection – supports introspection (dynamic discovery of system properties) and intercession (to dynamically modify structure or behavior)

Figure 2.10
Thin clients and compute servers

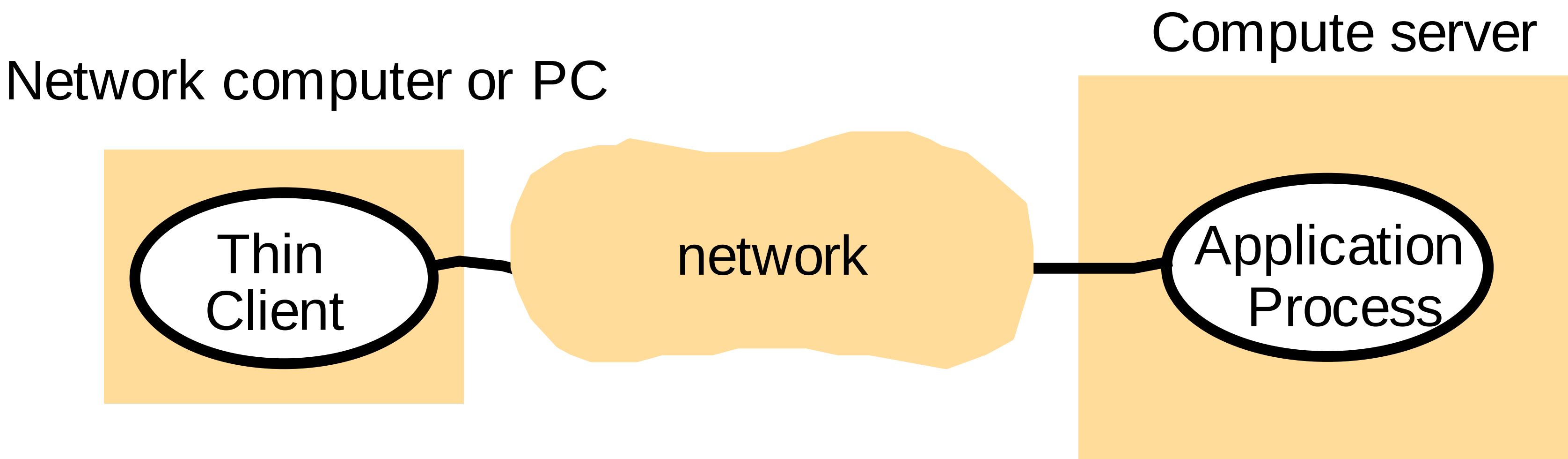
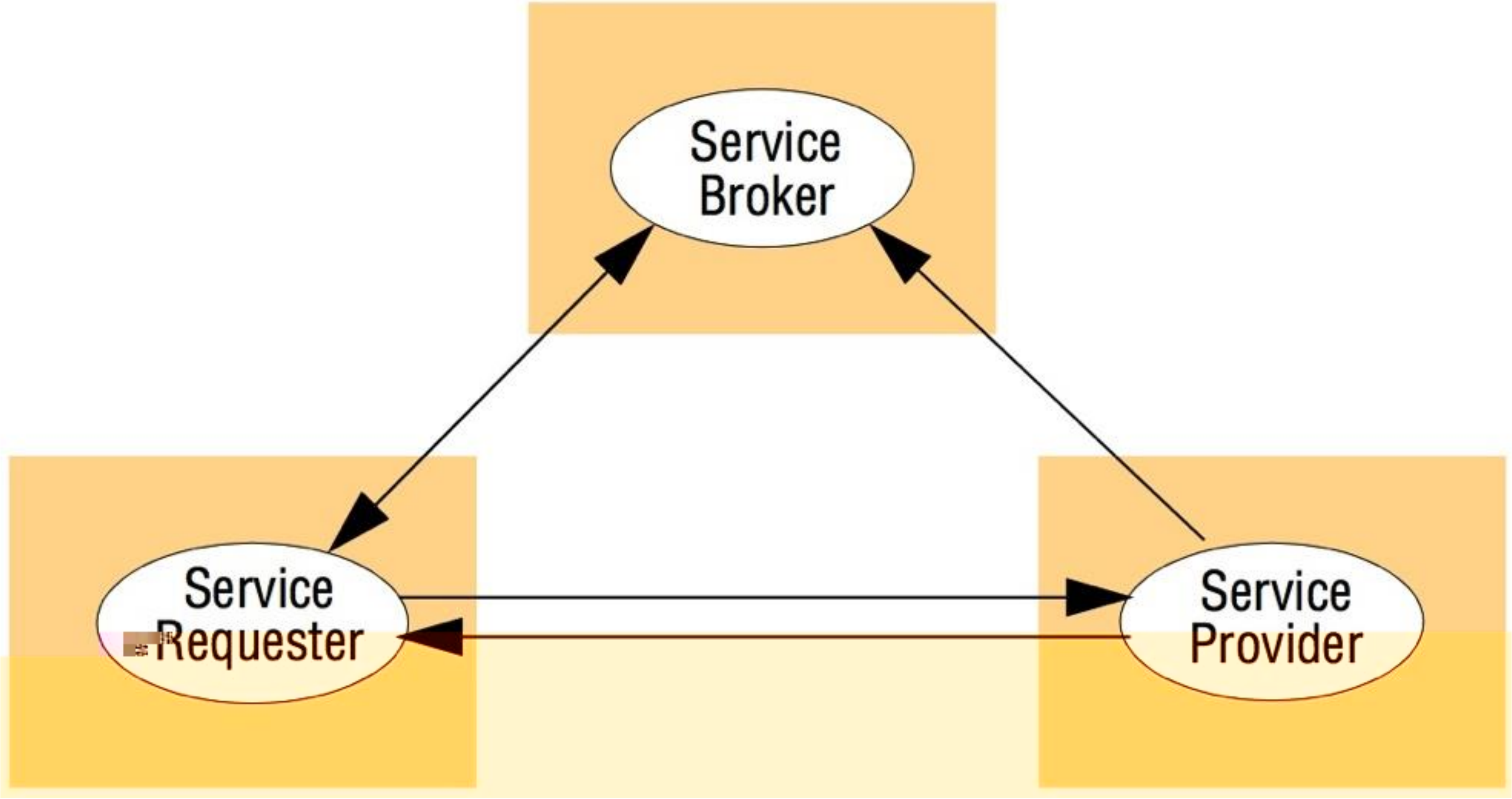


Figure 2.11
The web service architectural pattern



Middleware solutions

- Based on architectural models
- Major middleware categories on next slide

Figure 2.12
Categories of middleware

Major categories:	*Subcategory*	*Example systems*
Distributed objects (Chapters 5, 8)	Standard	RM-ODP
	Platform	CORBA
	Platform	Java RMI
Distributed components (Chapter 8)	Lightweight components	Fractal
	Lightweight components	OpenCOM
	Application servers	SUN EJB
	Application servers	CORBA Component Model
	Application servers	JBoss
Publish-subscribe systems (Chapter 6)	-	CORBA Event Service
	-	Scribe
	-	JMS
Message queues (Chapter 6)	-	Websphere MQ
	-	JMS
Web services (Chapter 9)	Web services	Apache Axis
	Grid services	The Globus Toolkit
Peer-to-peer (Chapter 10)	Routing overlays	Pastry
	Routing overlays	Tapestry
	Application-specific	Squirrel
	Application-specific	OceanStore
	Application-specific	Ivy
	Application-specific	Gnutella

Fundamental Models: Architectural patterns

Purpose:

- Make all relevant assumptions about system explicit
- Generalize what is possible and not possible given these assumptions
- Should capture the following aspects; interaction, failure, security

Fundamental Models: Interaction model

Focus:

- Effect of communication delay on distributed interactions (communication performance as a limiting characteristic)
- Impossibility of maintaining uniform time clocks on distributed nodes (cannot maintain global notion of time)

Performance of communication channels:

- Delay between sending and receiving of a message depends on: latency , network bandwidth, jitter

Fundamental Models: Interaction model

Computer clocks and timing events:

- Clocks on different nodes not synchronous
- Drift rate

Synchronous vs. asynchronous models:

Synchronous distributed system:

- Execution steps have lower/upper bounds
- Message transmission takes a bounded time
- Clock drift rates have known bounds

Asynchronous distributed system (e.g. Internet):

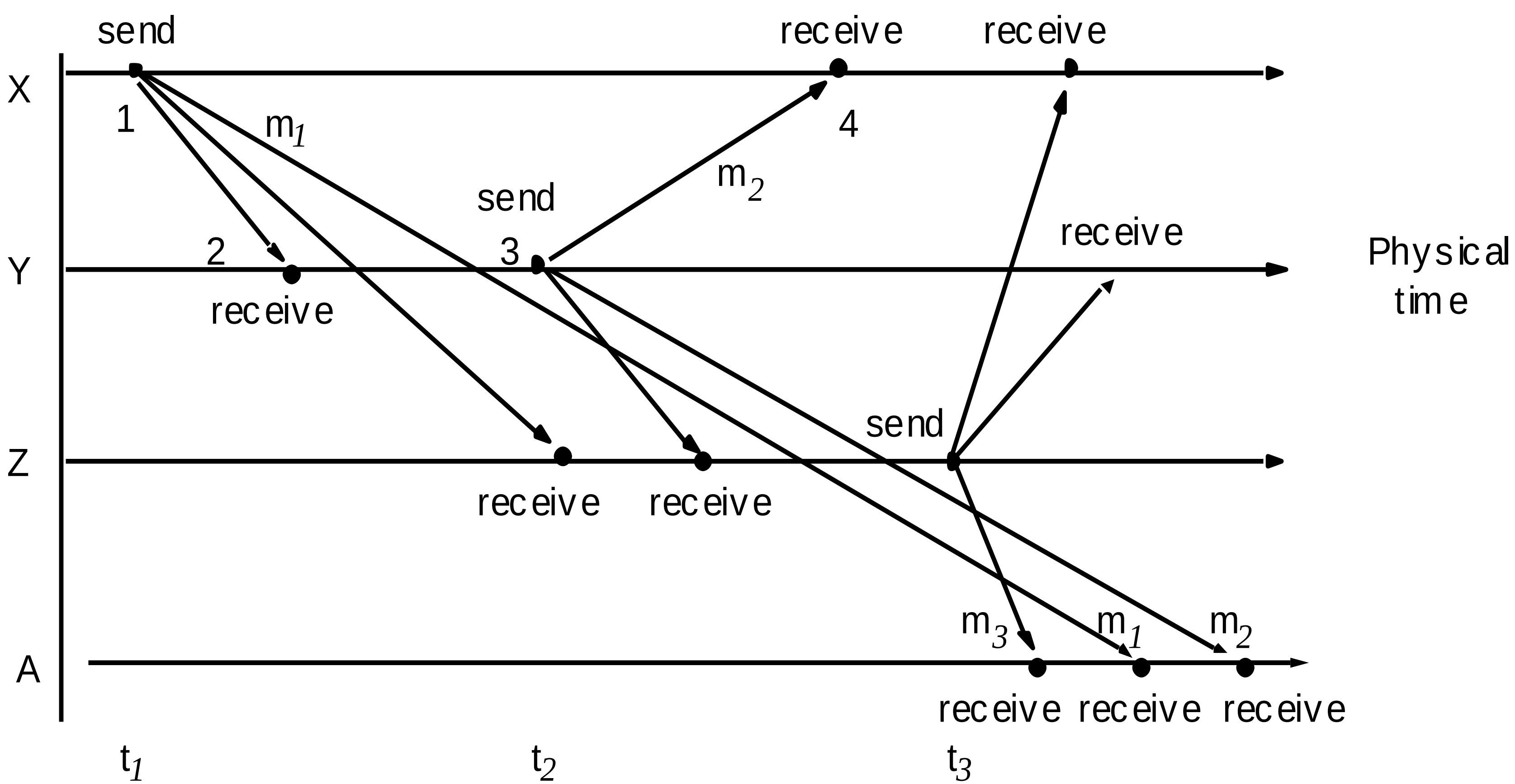
No bounds on: process execution speeds, message transmission delays, clock drift rates

Fundamental Models: Interaction model

Event ordering:

- Example on next slide
- Messages can be delivered in a different order than te order they were sent
- Logical clocks and logical ordering addresses this problem

Figure 2.13
Real-time ordering of events



Fundamental Models: Failure model

Omission failures:

- Process or communication channel fails to perform action
- Process crash – fail-stop if other processes can detect failure (e.g. through timeouts)
- Communication omission failure – network fails to deliver message

Arbitrary failures:

- Random failures – hard to detect

Timing failures:

- In synchronous distributed systems – violating upper bounds on message delivery time, process execution time, or clock drift rate

Figure 2.14
Processes and channels

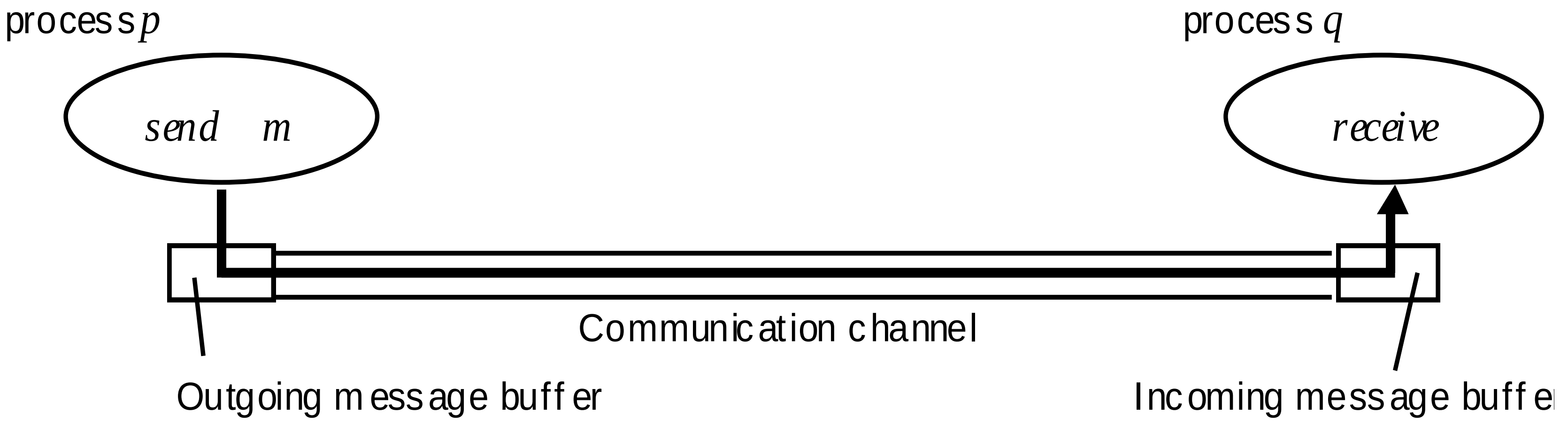


Figure 2.15
Omission and arbitrary failures

Class of failure	*Affects*	*Description*
		send,

Figure 2.11
Timing failures

Class of Failure	*Affects*	*Description*

Fundamental Models: Security model

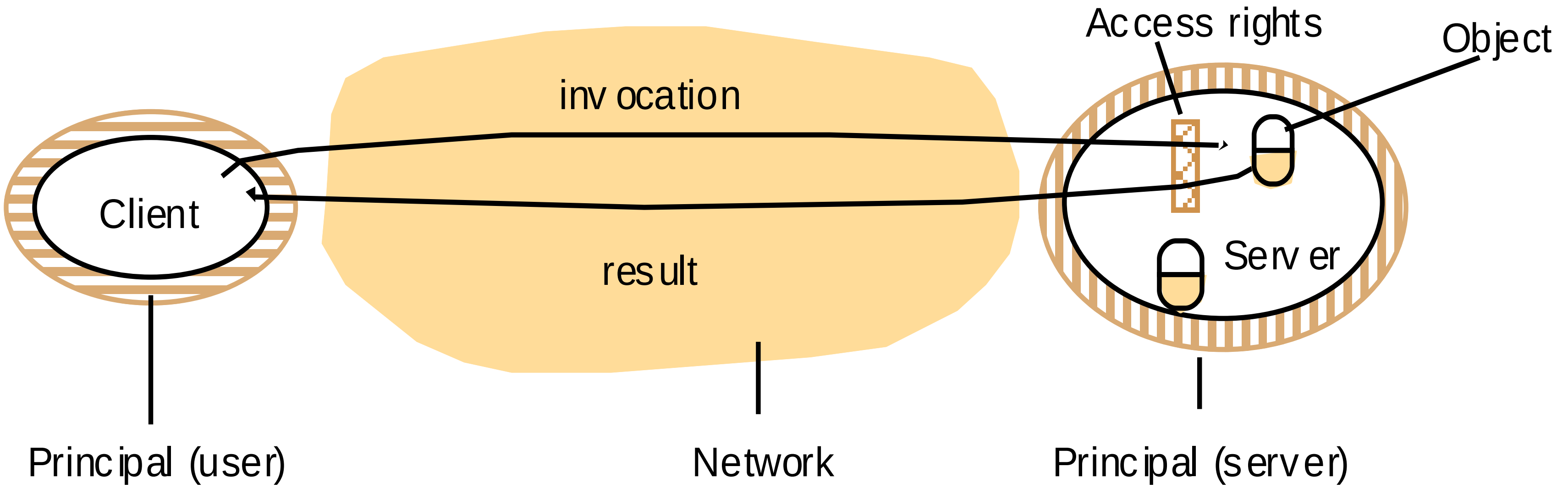
Security of a distributed system:

- Securing the processes and communication channels by protecting the objects that they encapsulate from unauthorized access

Object protection:

- Specify access rights to objects
- Principal: a user or process accessing an object or delivering a result
- Server needs to verify identity of principal and if principal is authorized to access object
- Client may need to check identity of principal behind the server to ensure result comes from required server

Figure 2.17
Objects and principals



Fundamental Models: Security model

Enemy or adversary:

- An entity capable of sending messages to processes or copying message sent between processes
- Threats to processes and threats to communication channels

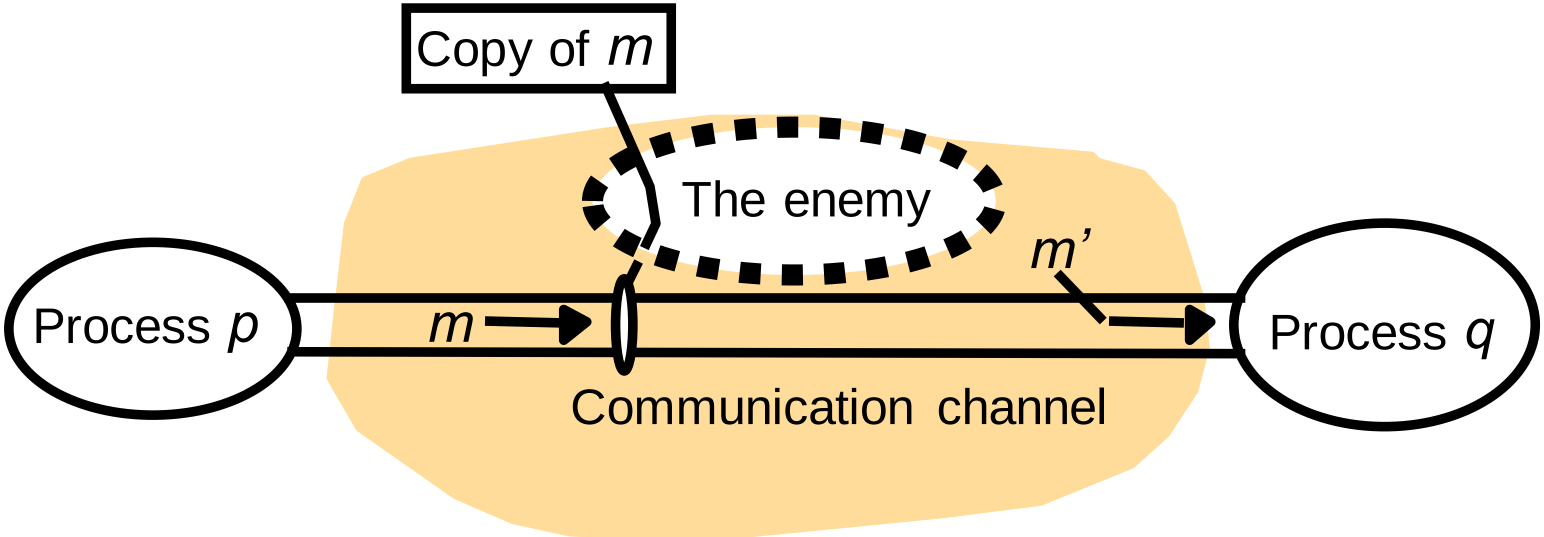
Threats to processes:

- Server processes
- Client processes

Threats to communication channels:

- Copying, altering, or injecting messages

Figure 2.18
The enemy



Fundamental Models: Security model

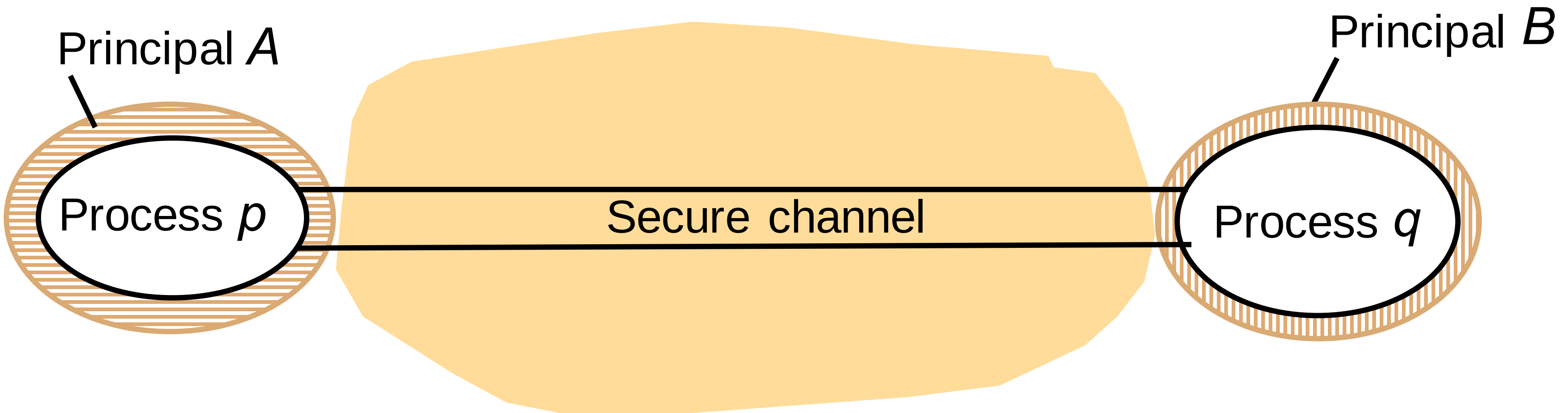
Dealing with security threats:

- Cryptography
- Authentication
- Secure channels

Other types of threats:

- Denial of service
- Mobile code

Figure 2.19
Secure channels



Summary

Models of distributed systems:

- Physical Models
- Architectural Models
- Fundamental Models