System Models

Vaidė Narváez

Computer Information Systems

June 14th, 2010

How systems are structured and organized

PAYAP UNIVERSITY

Architectural models

The architecture of a system is its structure in terms of separately specified components.

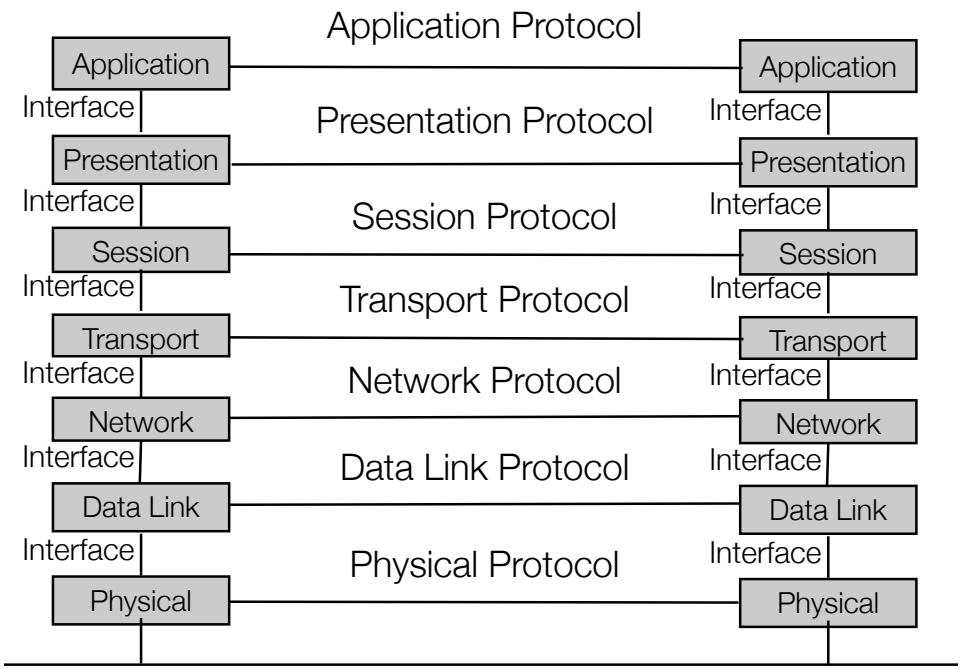
PAYAP UNIVERSITY

Architectural models

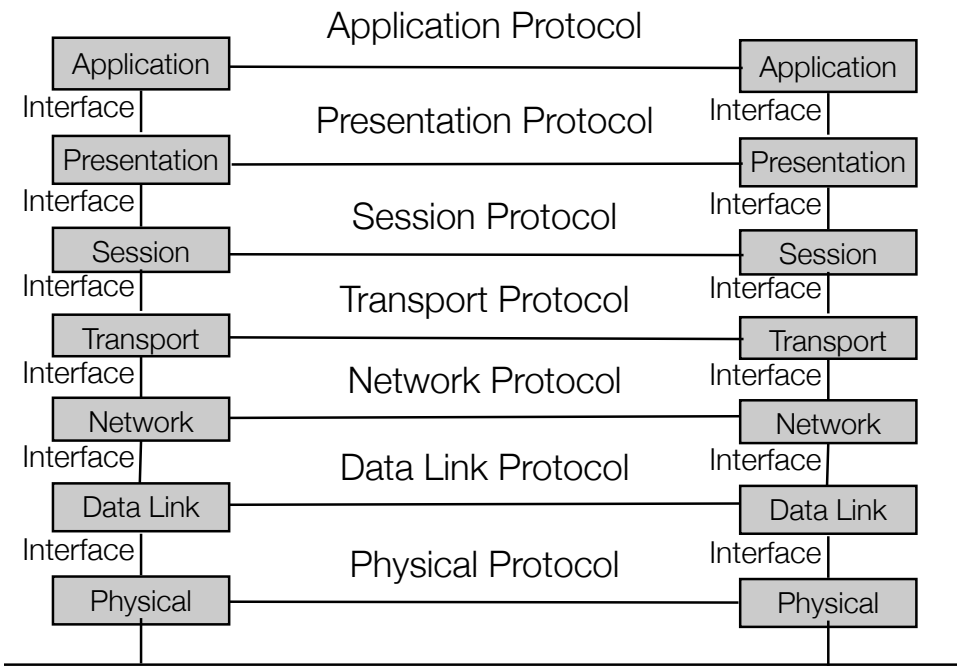
- placement of components across a network of computers
  - define useful patterns for the distribution of data and workload
- interrelationships between components
  - functional roles, patterns of communication
- abstraction: server processes, client processes, peer processes
- variation of models, e.g. for client-server architecture
  - define mobile code to have some part of the application run on the client
  - enable computers and other mobile devices to be added or removed seamlessly

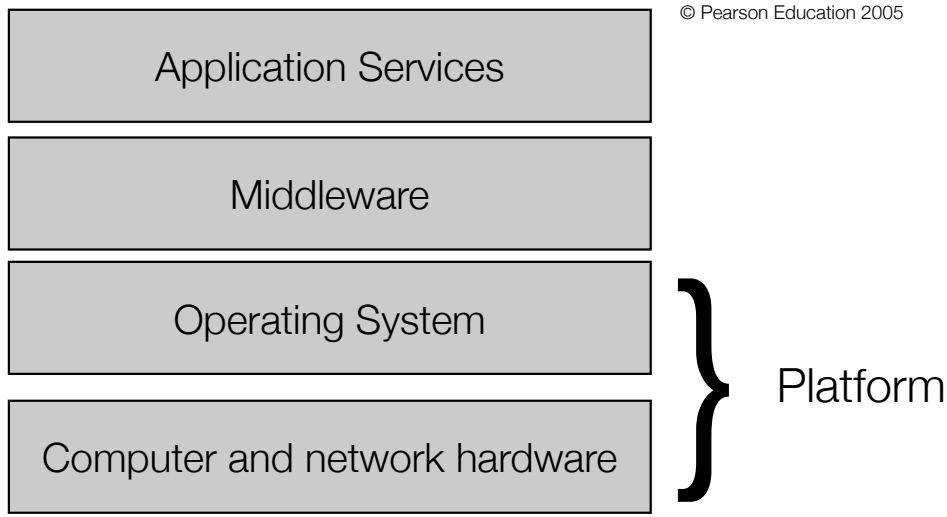
PAYAP UNIVERSITY

Layers



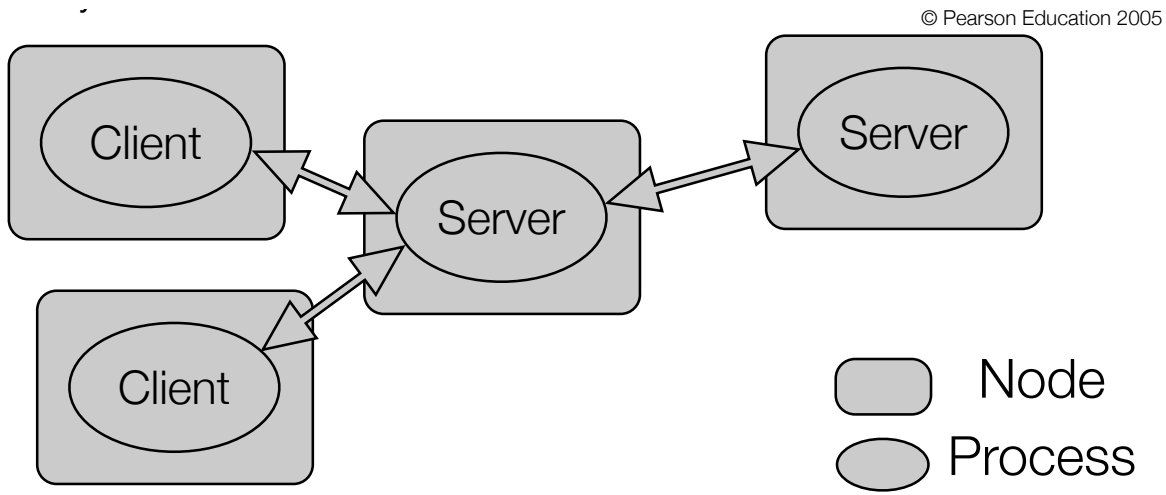
Layers





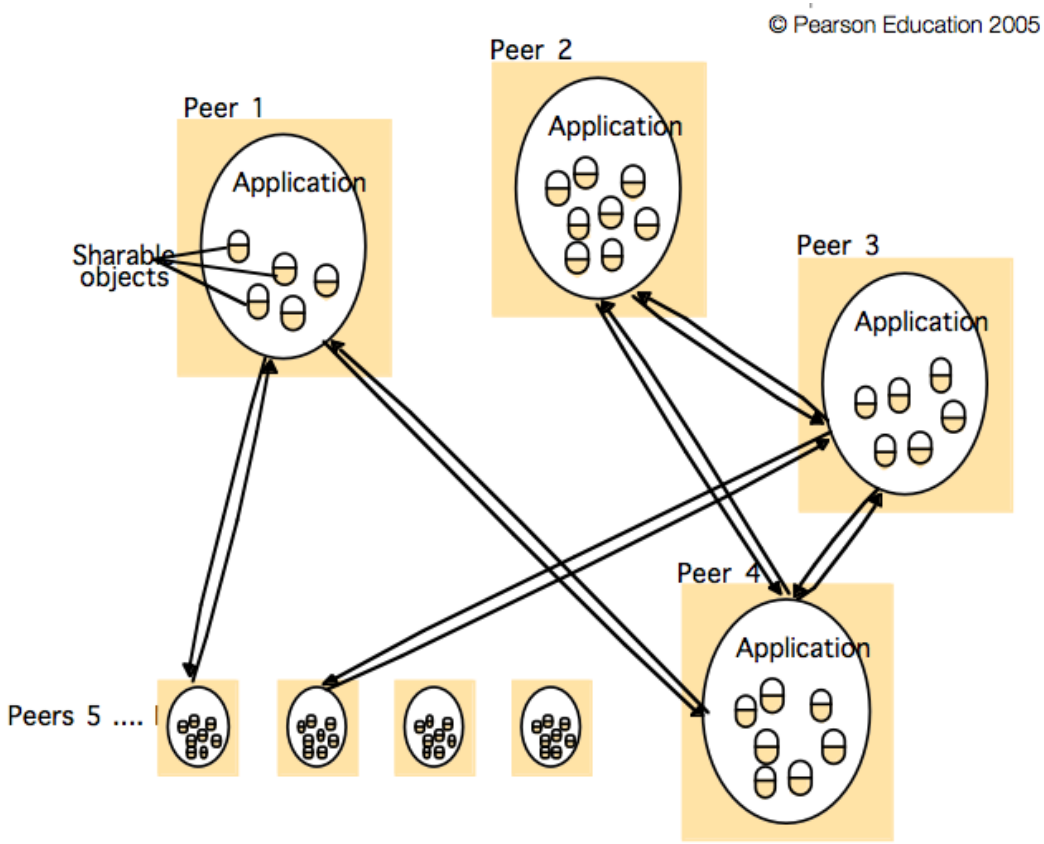
System Architectures:Client/Server

- Client/Server model
  Client: consumer process, uses remote services / information
  Server: provider process, offers a service / information



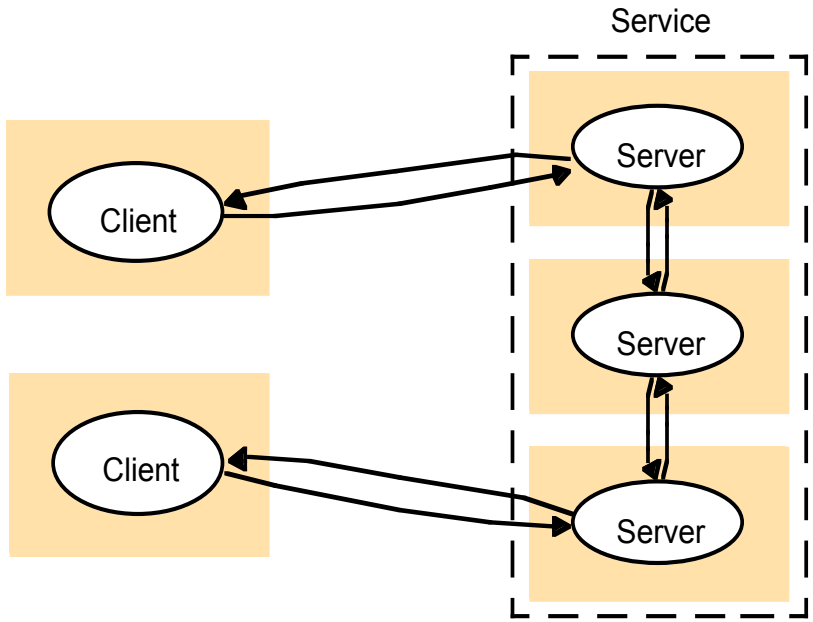
System Architectures: P2P

- Peer-to-Peer model
  - potentially large number of participants (often, home users)
  - communication pattern varies over time
  - replication necessary to provide resilience in the event of disconnection



System Architectures:Variations

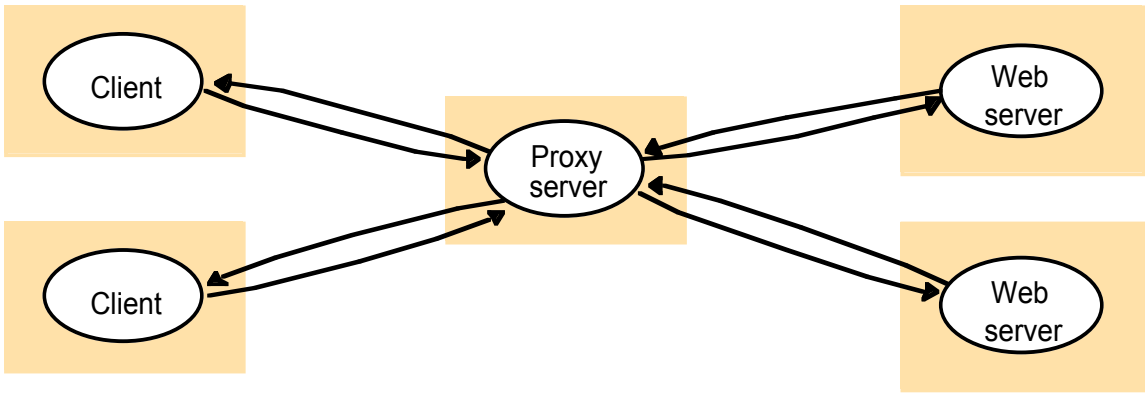
- Services provided by multiple servers



PAYAP UNIVERSITY

System Architectures:Variations

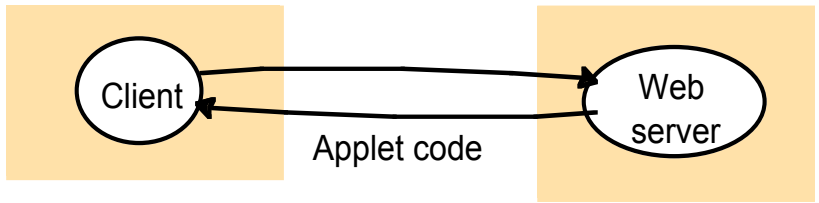
- Proxy servers and caches



System Architectures:Variations

- Mobile code

a) client request results in the downloading of applet code



b) client interacts with the applet

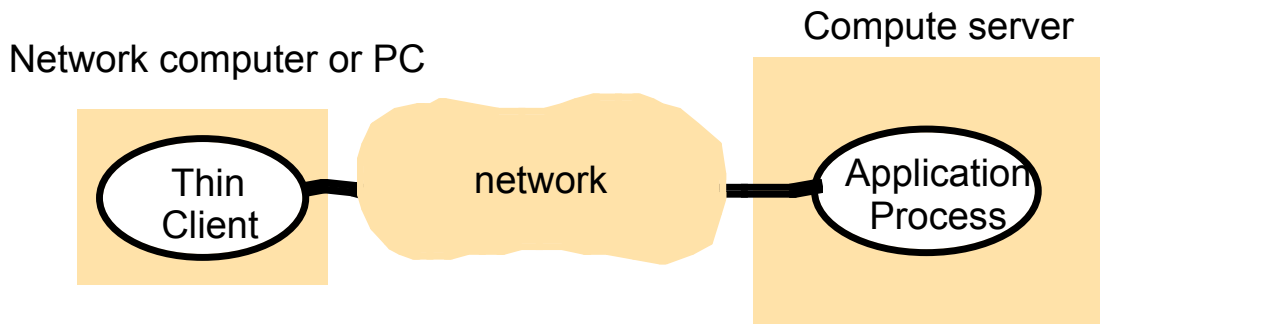
PAYAP UNIVERSITY

System Architectures:Variations

- Mobile agent
  - running program is moved from node to node (with both code and data)
  - security challenge to the server: client gets hold of the entire state of the agent
    -only have non-secret data in the agent
  - security challenge to the client: client node runs arbitrary server-defined code
    - need to establish trust in agent code
    - need to restrict agent's access to local resources

System Architectures:Variations

- Thin Clients:
  - Client performs just I/O, no computation



PAYAP UNIVERSITY

Design requirements for DS

- Various objectives for creation of distributed systems
  - sharing of computational resources (e.g. cluster computing)
  - sharing of data
  - sharing of services
- Performance issues: responsiveness, throughput, load balancing
- Quality of Service (QoS): reliability, security, performance, adaptability, time- critical data
- Dependability: correctness, security, fault tolerance, (maintainability)
- Caching and Replication

PAYAP UNIVERSITY

Fundamental models

Models of distributed systems:

- Interaction model manages performance and time limits (e.g. communication involves delays)
- Failure model specifies faults and defines reliable communication (node and network failure threatens correct operation of system)
- Security model describes threats to processes and communication channels (consider attacks by both internal and external agents)

PAYAP UNIVERSITY

Interaction model

Two factors affecting interacting processes

- communication performance (latency, bandwidth, jitter)

- no global notion of time

PAYAP UNIVERSITY

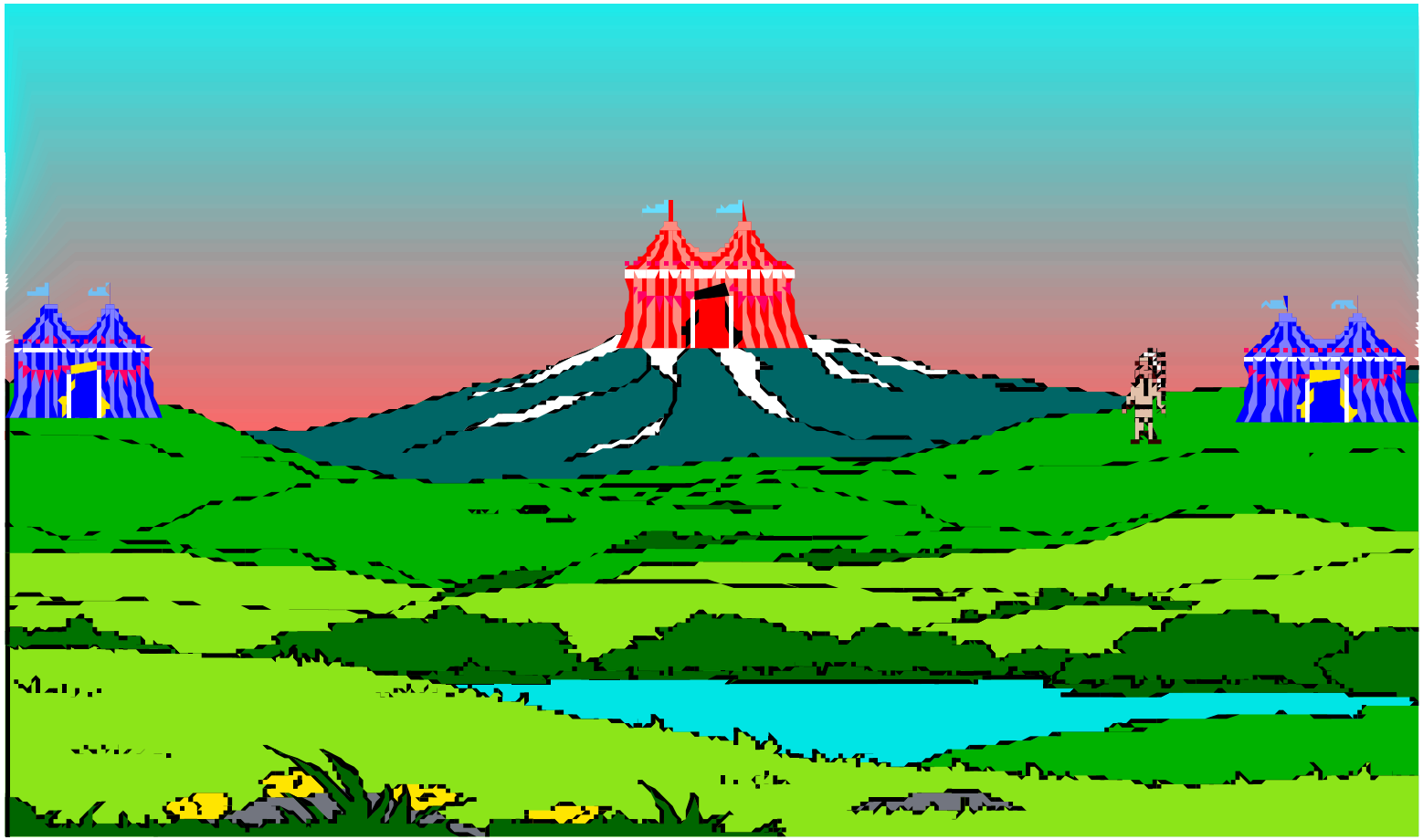
Synchronous vs. asynchronous

- Synchronous Systems
  - known lower and upper time bound for each execution step, for each message transmission, and for the clock drift
  - practically difficult to build, may help in simplifying analysis

- Asynchronous Systems
  - no time bounds on process execution speed
  - message transmission delays are not known
  - the drift rate of a clock is arbitrary

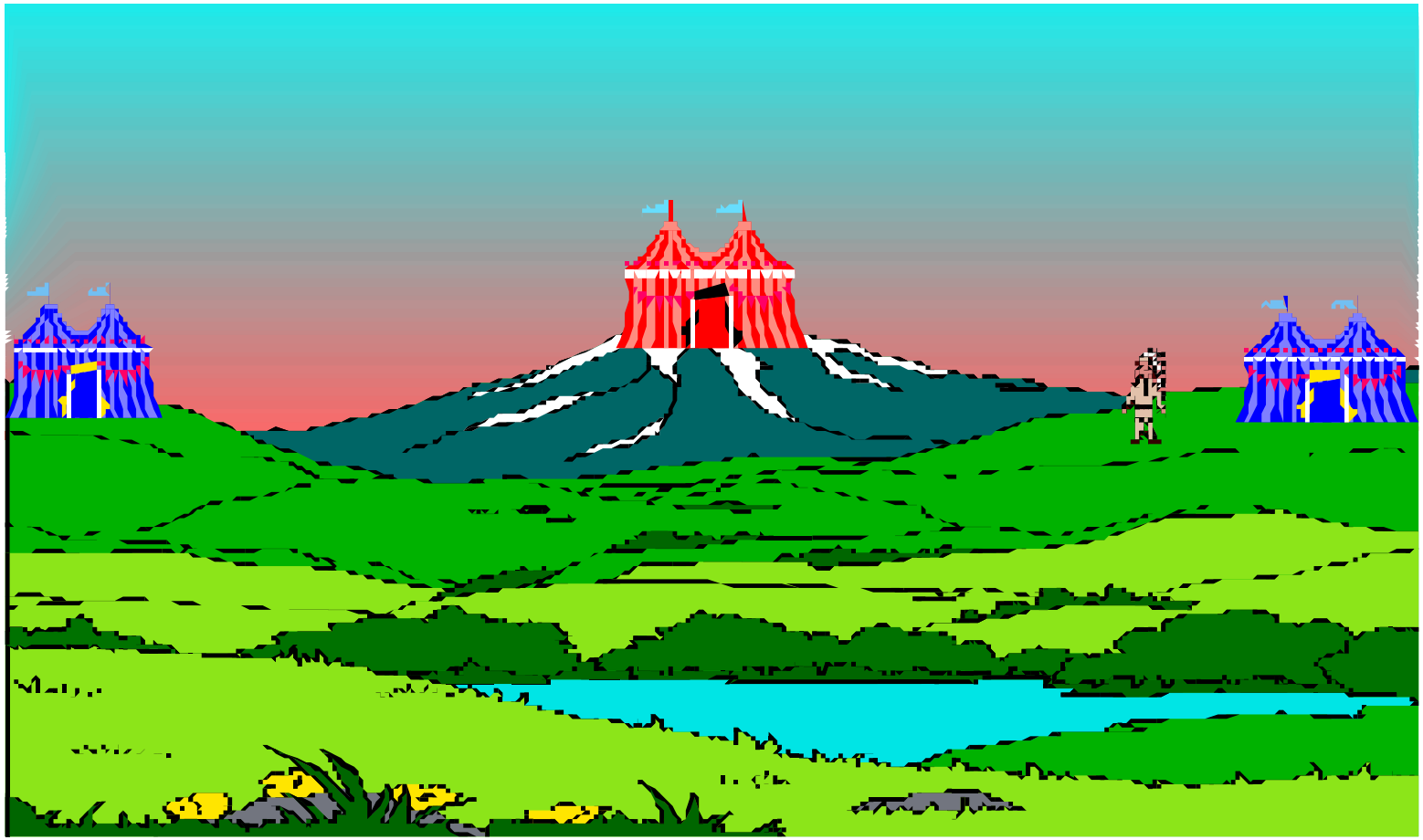
PAYAP UNIVERSITY

Agreement



PAYAP UNIVERSITY

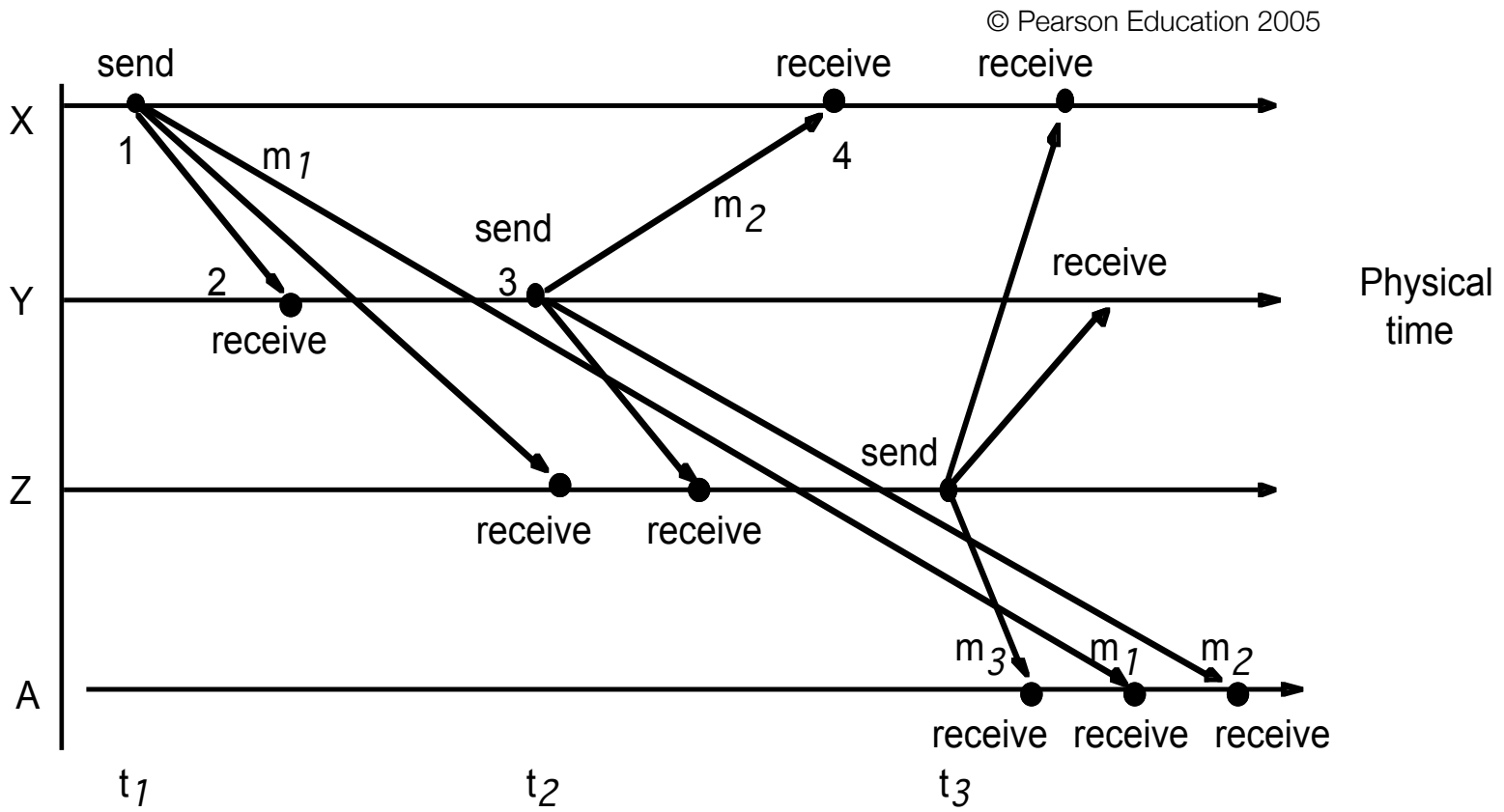
Agreement



It is impossible to synchronize the clocks in an asynchronous system!

PAYAP UNIVERSITY

Event Ordering



PAYAP UNIVERSITY

Logical Time

Happens-before relation: e_1 happens before e_2, iff

- e_1 is executed by the same process before e_2, or
- e_1 is a send operation, and e_2 is the corresponding receive operation, or
- there is an e_3 such that e_1 happens before e_3, and e_3 happens before e_2

Logical time: Assign a number L to each event, such that $L(e_1) < L(e_2)$ if e_1 happens before e_2

PAYAP UNIVERSITY

Synchronous vs. asynchronous

How to choose?

- Since any system is asynchronous, a protocol designed for such a system will work in any other system

PAYAP UNIVERSITY

Synchronous vs. asynchronous

How to choose?

- Since any system is asynchronous, a protocol designed for such a system will work in any other system

- So, why not develop all protocols for asynchronous systems?

PAYAP UNIVERSITY

Synchronous vs. asynchronous

How to choose?

- Since any system is asynchronous, a protocol designed for such a system will work in any other system
- So, why not develop all protocols for asynchronous systems?
- So, why not develop all protocols for synchronous systems?

PAYAP UNIVERSITY

Failure Model

- Omission Failures: process or channels fails to perform an operation
  - process omission failure (e.g. crash failure)
    fail-stop: other processes can detect crash (requires guaranteed delivery of messages)
  - communication omission failures (message drop)
- Arbitrary Failures (Byzantine failure): anything may happen
- Timing Failures:
  - processes: clock drift, execution bounds
  - channels

PAYAP UNIVERSITY

Failures

Class	Affects	Description
Fail-stop	Process	Process halts and remains halted. Other processes detect this state
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state
Omission	Channel	A msg inserted in an outgoing buffer never arrives at the other end's incoming buffer
Send-omission	Process	A msg is not put in outgoing buffer of the process
Receive-omission	Process	A msg is put in incoming buffer, but the process doesn't receive it
Arbitrary	Either	Process or channel exhibits arbitrary behaviour

PAYAP UNIVERSITY

Failure Model

- Masking failures: reconstruct reliable services on top of unreliable ones
  - through retries, error correction, ...

- Reliability of one-to-one communication:
  - validity (messages are eventually delivered to the receiver)
  - integrity (received message identical to sent one, and no message is delivered twice)

PAYAP UNIVERSITY

Security Model

- securing processes and channels against unauthorized access
- protecting objects: access rights given to a principal
- assumption of an enemy (aka adversary), capable of (threat model)
  - sending messages to any process
  - reading and copying any message between a pair of processes
- enemy may operate either legitimitely-connected node, or illegal node

PAYAP UNIVERSITY

Security Model

- threat to processes: may receive messages sent by enemy
  - may not be able to reliably determine identity of sender
  - server: may not be able to identify principal
  - client: may fall to ”spoofing”
- Threats to communication channels: enemy may
  - copy, alter, inject, or delete messages
  - gain information only intended for the communication partner
- Other threats: denial of service, trojan horses, ...
- Defeating security threats: cryptography, authentication, secure channelse