Distributed Systems

Abien

Distributed System

A collection of computing elements working together to appear as a single system to the end-user

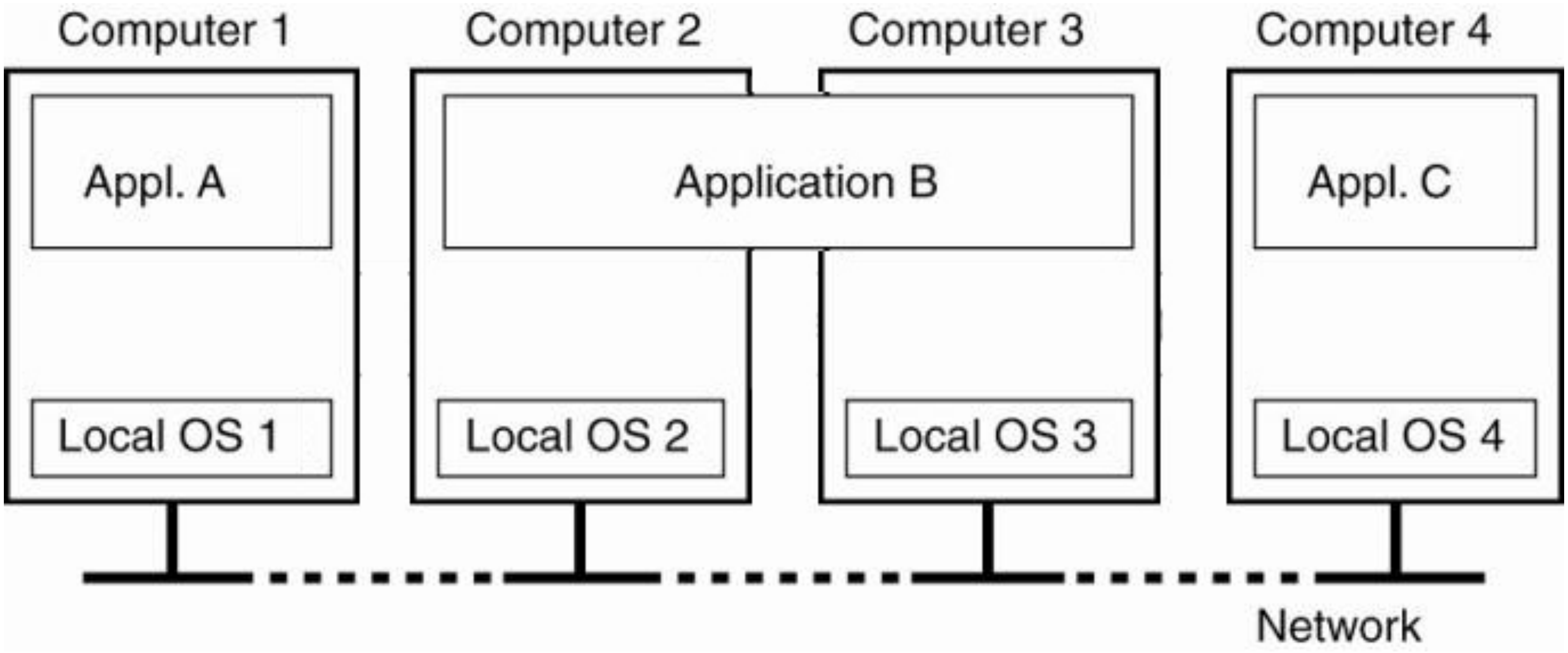
Single system

- Single computing device
- Fixed location or mobile
- Distribution of computing tasks is possible through concurrency or parallelization

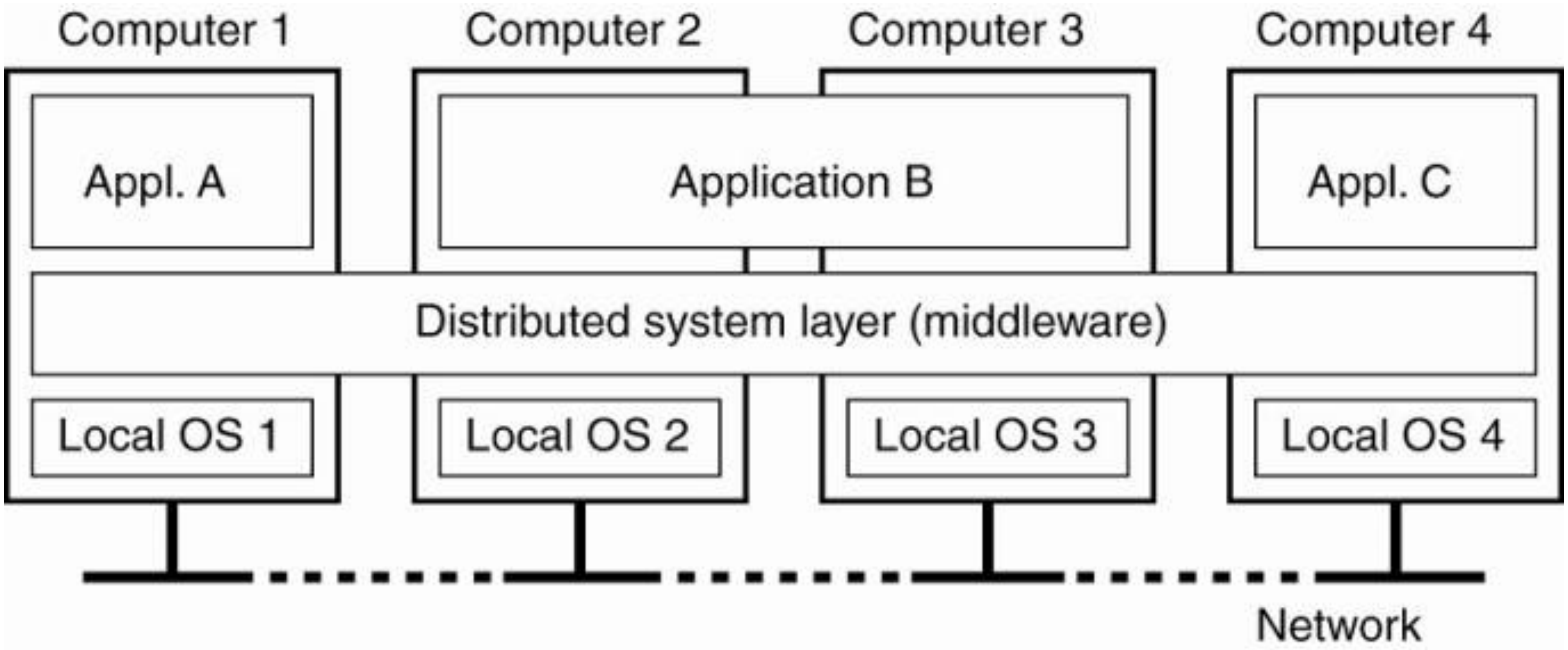
Distributed System

- Appears as a single system
- Multiple computing devices
- Often geographically dispersed
- Computing devices work to achieve a common goal
- Fault tolerance

Computer Network



Distributed System



Middleware

Middleware for distributed systems is similar an operating system for a single computer. Instead of managing resources within a single machine, it shares those resources with other machines over a network

- Web Servers
- Message Oriented Middleware
- ODBC/JDBC

Distributed Systems

1970s	ARPANET’s email application
1980s	Newsgroups and Bulletin Board Systems
1990s	The modern internet

Why build a Distributed System?

Stored data, processing, or users are in different physical locations.

- **Distributed Databases** – databases that are available on multiple machines. These are often found within organizations
- **Sensor Networks** – contains many small nodes that have one or more sensing devices. Such networks are used in environmental or system monitoring

Networked systems often provide more computing power than a single system.

- **Cluster Computing** – multiple identical computers on the same network are used for parallel high performance computing
- **Grid Computing** – computers from different administrative domains working on a task, such as those used in Distributed Computing Projects

Distributed Computing – Folding@home

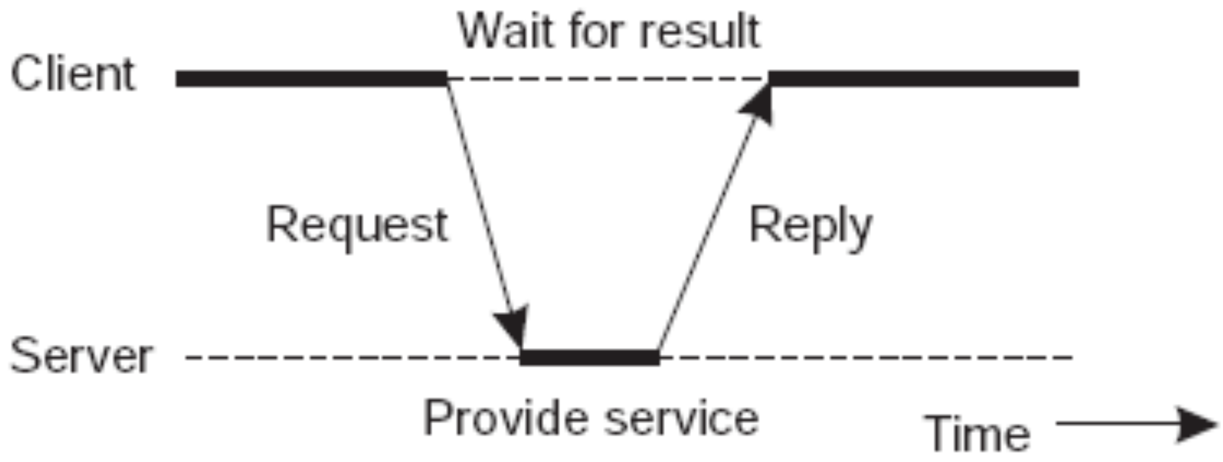
Grid Computing project that Simulates Protein Folding and movement of molecules.

Distributed Systems

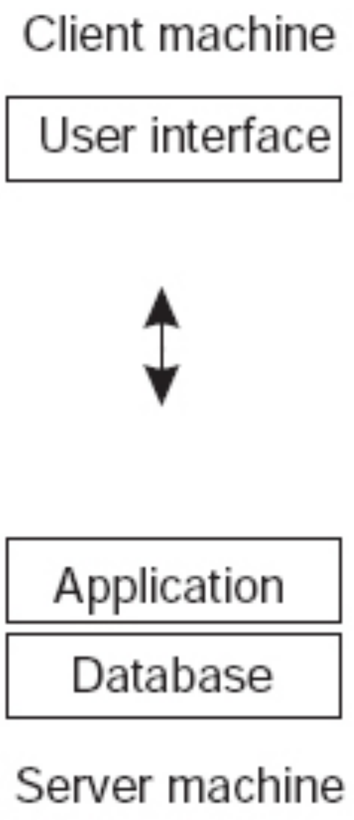
Architecture

System and Software Architecture for distributed systems

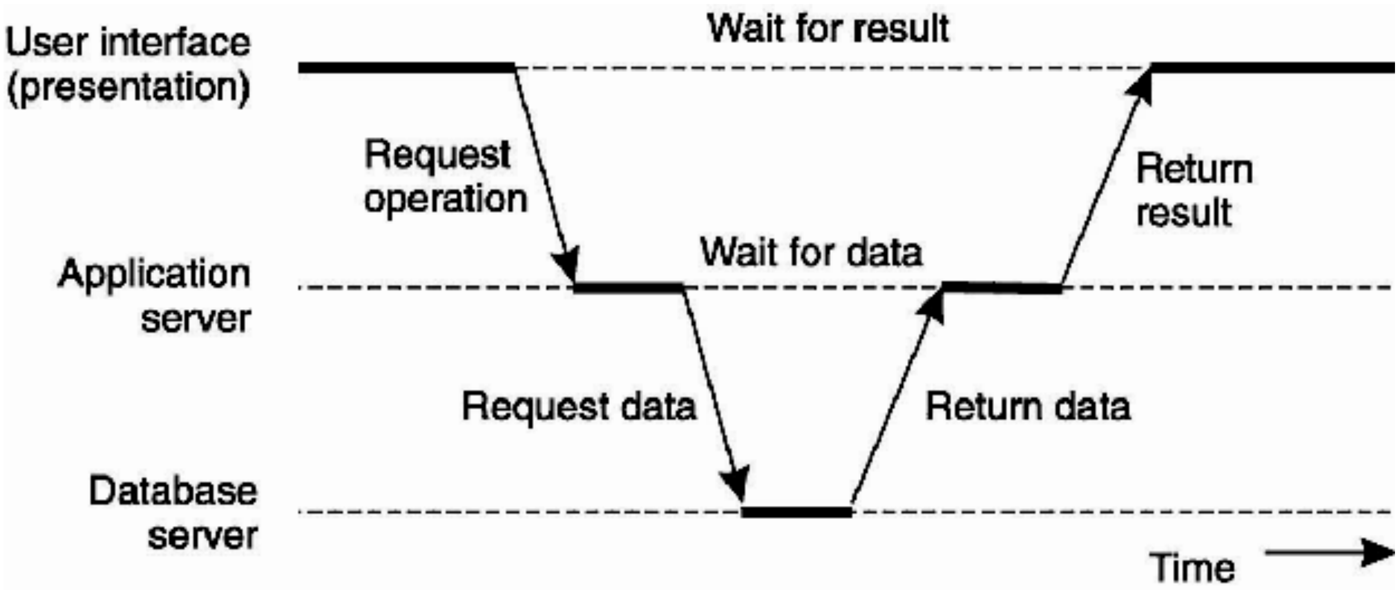
Client-Server Architecture



Two-tiered Architecture



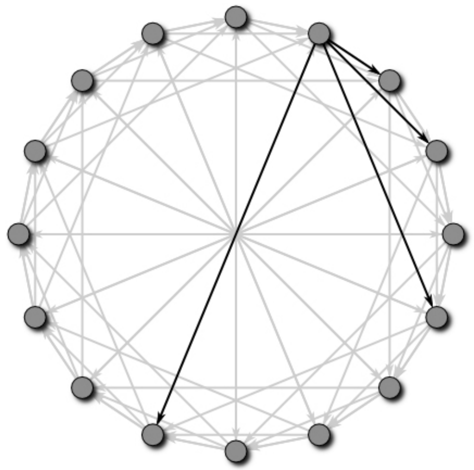
Three-tiered Architecture



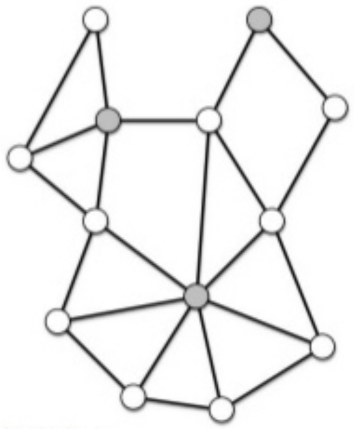
Peer to Peer systems

Each node of a peer to peer system acts as both client and server.

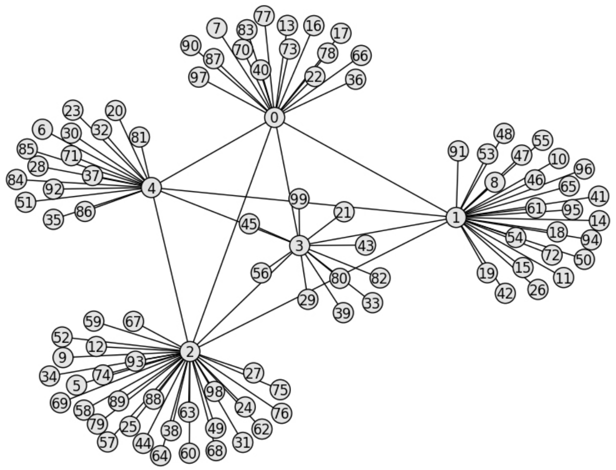
Structured



Unstructured



Hierarchically Organized



Middleware design

Wrapper

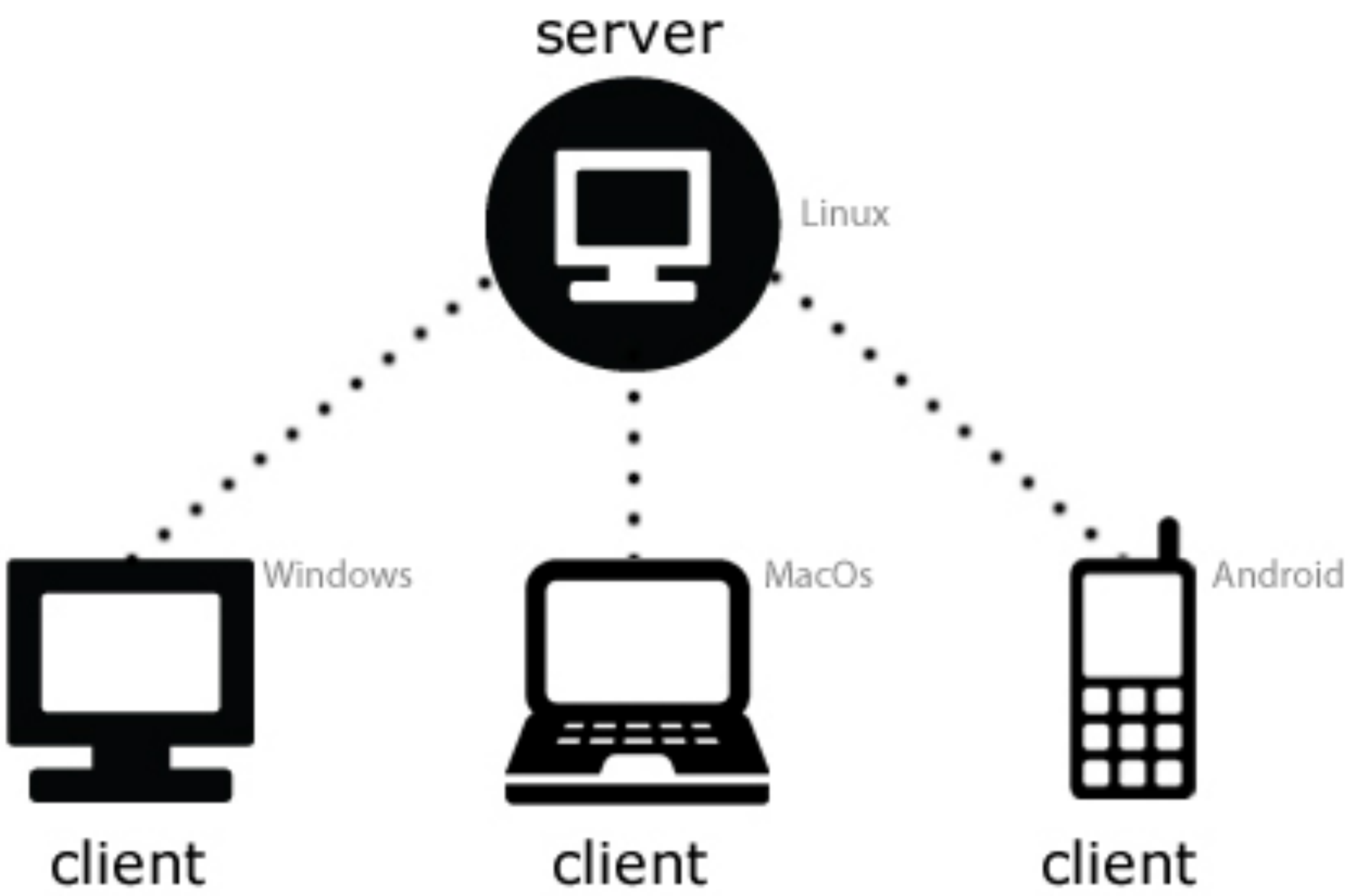
If applications on a distributed system have incompatible interfaces, a wrapper is created to enable compatibility between applications.

Interceptor

Allows a local object to make calls to a method belonging to an object on a different node

Folding@home - architecture

- Grid Computing
- Run parts of protein folding simulation across various operating systems and hardware
- Project servers will assign work units to teams and individual machines
- Client – Server architecture is used

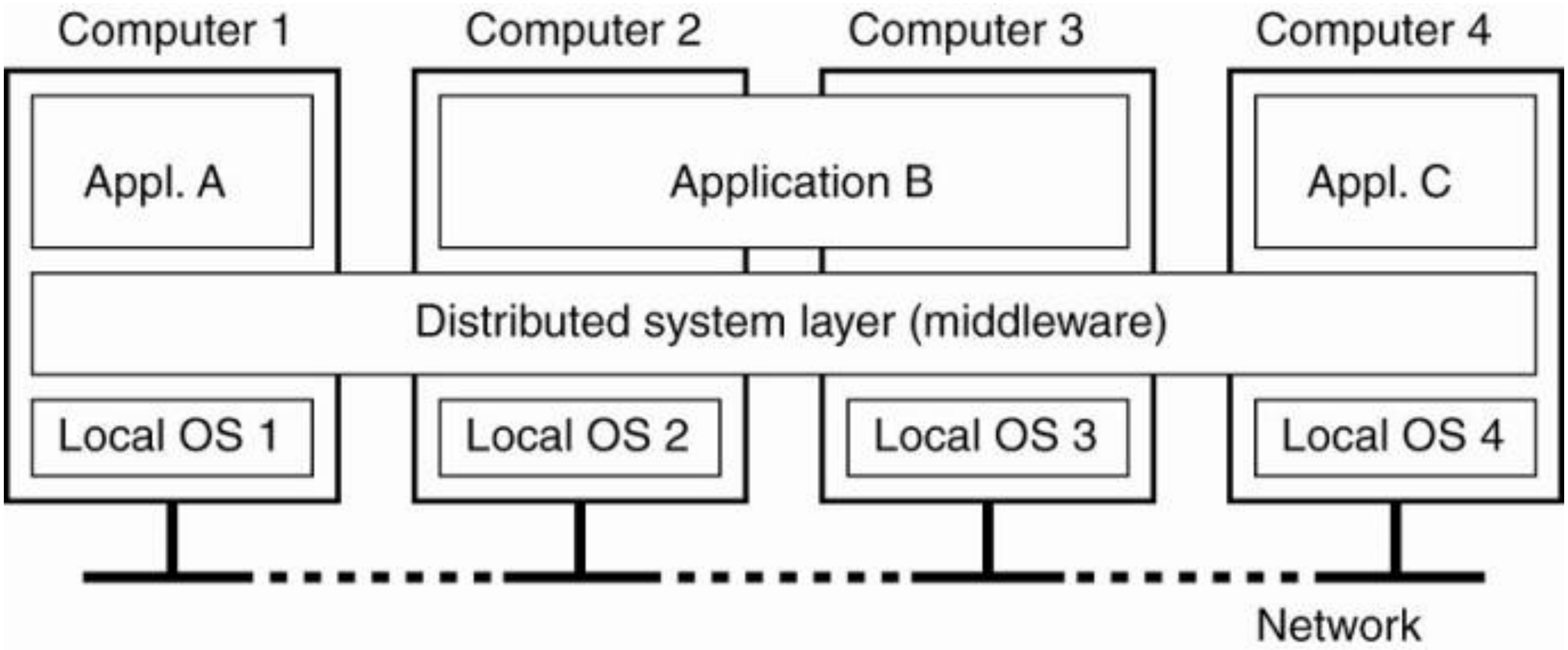


Interprocess Communication

IPC in distributed systems

Distributed System

Computer Network

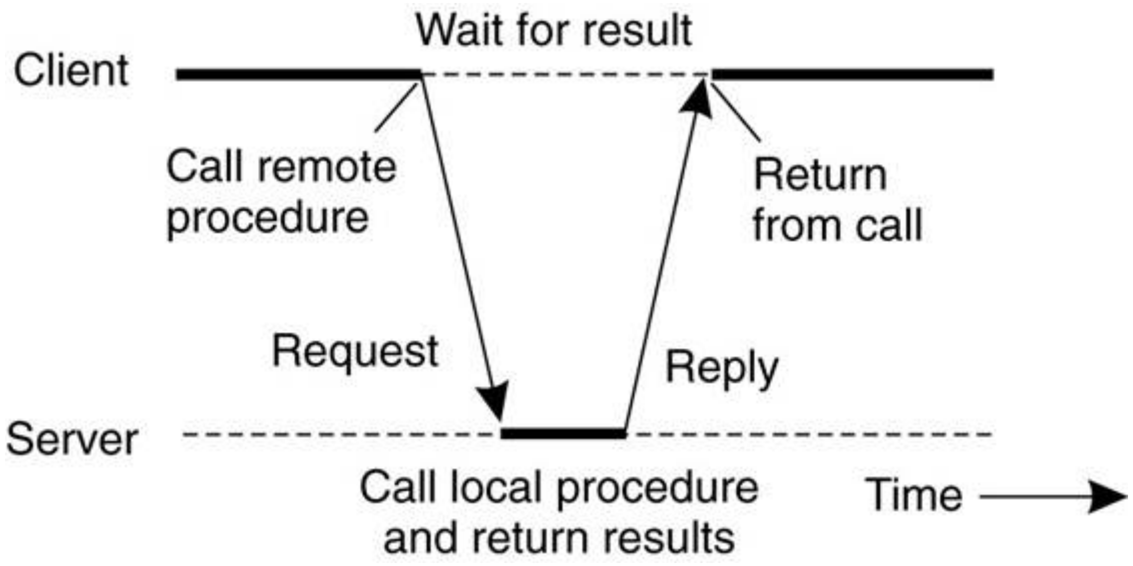


Middleware Protocols

- Remote Procedure Call
- Message Oriented Middleware
- Message Passing Interface

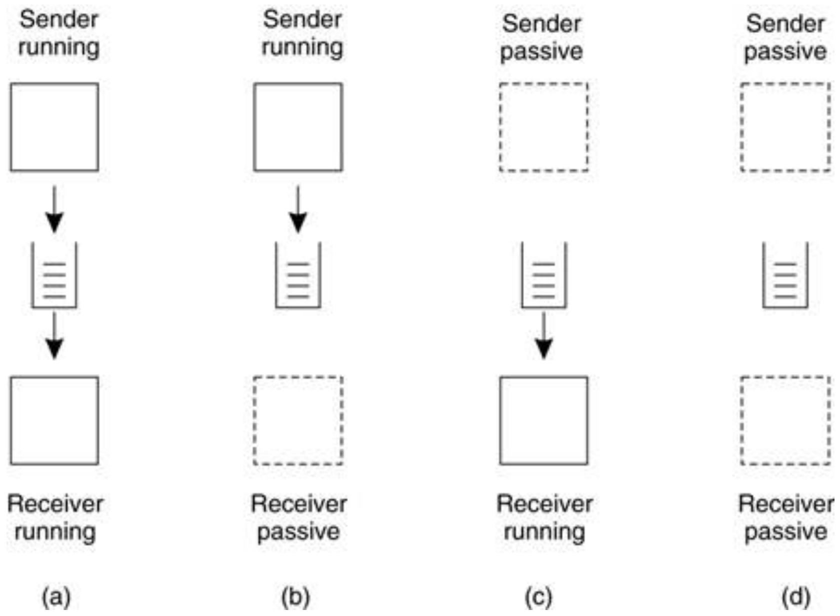
Remote Procedure Call

- IPC facility for calling a procedure on another machine
- Client stubs transform local procedure calls into network requests
- Server stubs transform requests from the network into calls



Message Oriented Middleware

- Also known as message queueing systems, applications send messages to one another via queues
- The sender or receiver doesn’t have to be active during transmission

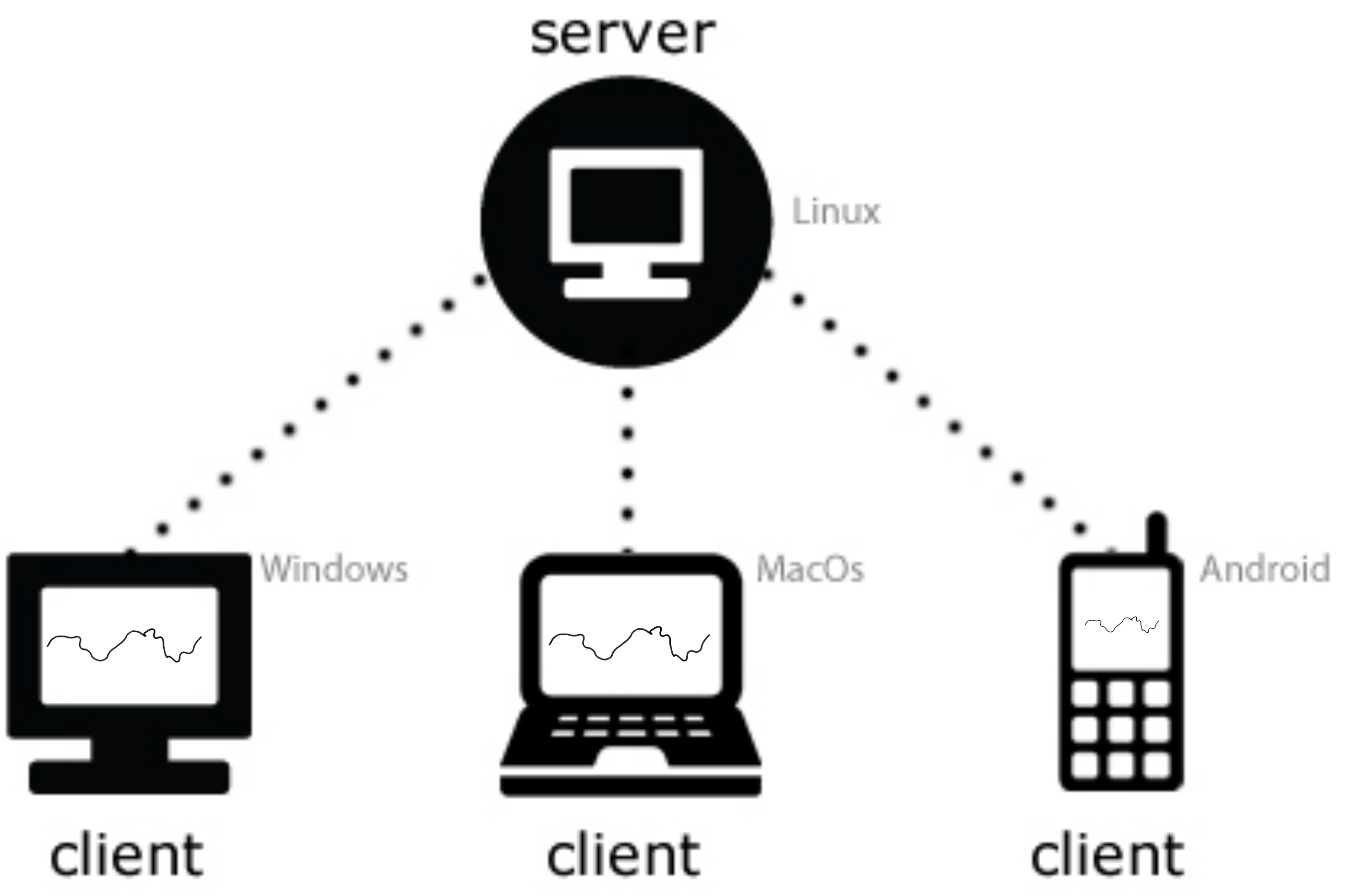


Message Passing Interface

Operation	Description
MPI_bsend	Append outgoing message to a local send buffer
MPI_send	Send a message and wait until copied to local or remote buffer
MPI_ssend	Send a message and wait until receipt starts
MPI_sendrecv	Send a message and wait for reply
MPI_isend	Pass reference to outgoing message, and continue
MPI_issend	Pass reference to outgoing message, and wait until receipt starts
MPI_recv	Receive a message; block if there is none
MPI_irecv	Check if there is an incoming message, but do not block

Folding@home – Communication

- Grid Computing
- Run parts of Protein folding simulation across various operating systems and hardware
- Client – Server architecture is used
- Message Passing Interface is used to optimize parallel computation



Coordinating Processes

Process Synchronization and Cooperation in distributed systems

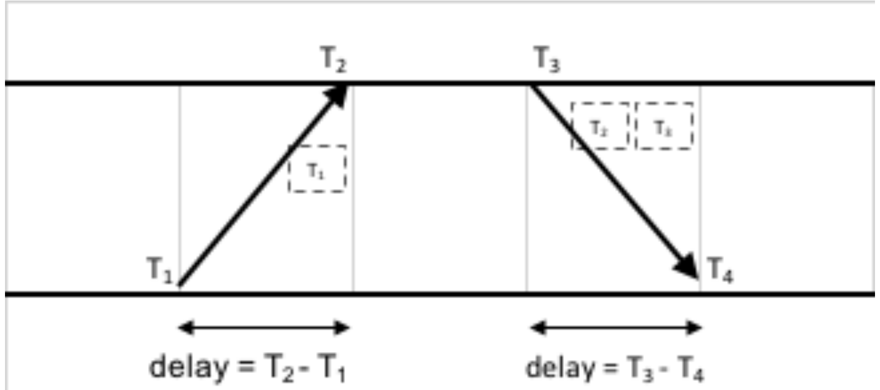
Coordinating Processes

Process synchronization ensures that sequential processes in a distributed system occur in the appropriate order

Coordination is the management of processes that depend upon the completion of other processes

Clock Synchronization

Network Time Protocol – Contact a time server, and use timestamps to estimate delay or relative offset.



Offset, $\theta = \frac{(T2-T1)+(T3-T4)}{2}$ Delay, $\delta = \frac{(T4-T1)-(T3-T2)}{2}$

Eight pairs of offset and delay are calculated. The minimum value is the best estimation of delay and offset between the servers. The server adjusts its clock.

Clock Synchronization

The Berkeley Algorithm – a server gets times from other machines and has them adjust their clocks to its time

Reference broadcast synchronization – two nodes, p and q, use linear regression of their delivery times to compute clock offset

Offset[p,q](t) = $\alpha t + \beta$, where α and β are computed from pairs of times from nodes p and q

Synchronizing Logical Clocks

Lamport's Logical Clocks – all processes agree on the order in which events occur, and adjust their clocks if necessary.

Vector Clocks – each process maintains a vector for the number of events that have occurred before it.

Distributed Algorithms for Collaboration and Coordination

Mutual Exclusion allows one node to access a resource at a given time

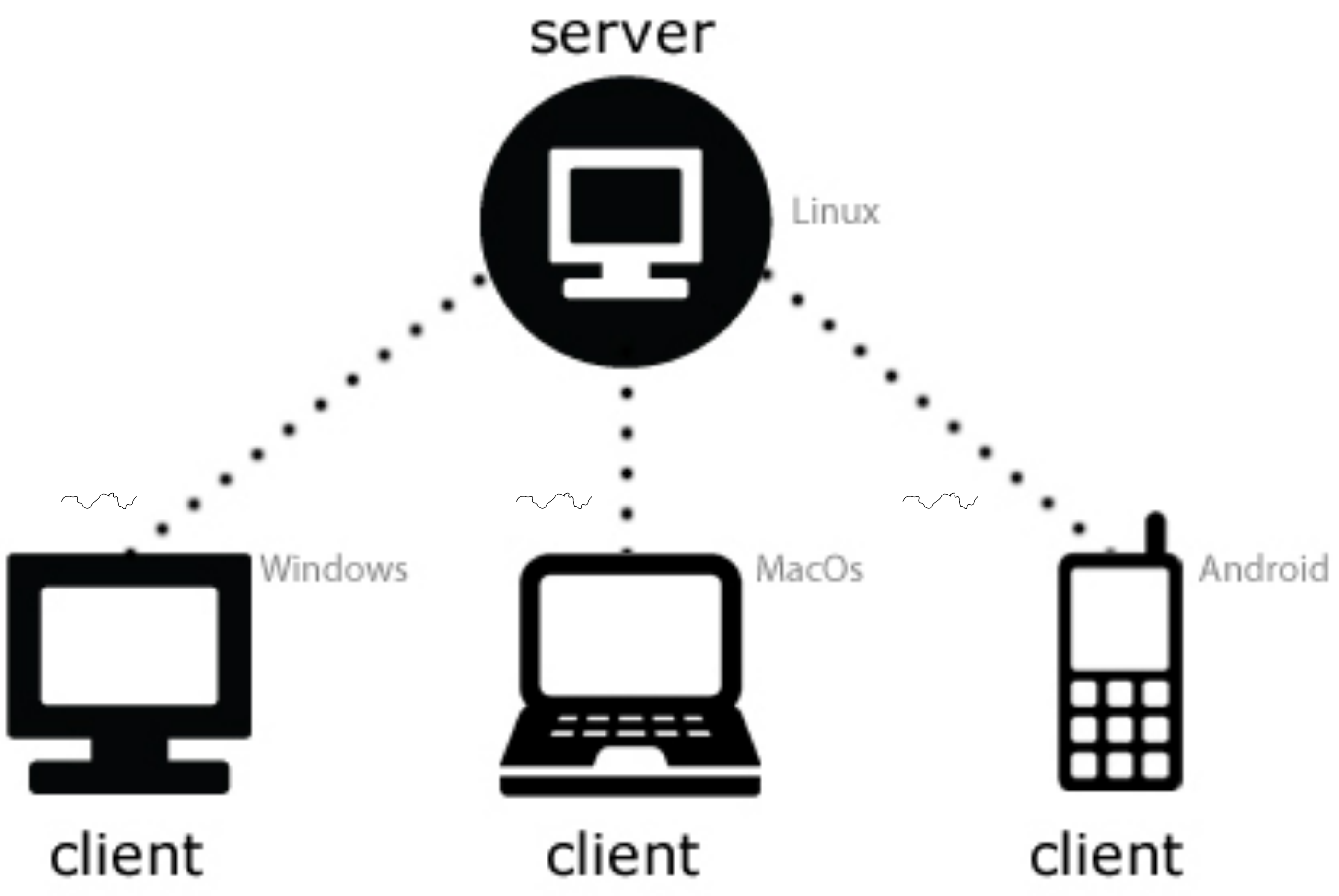
Election Algorithms are used to determine which process will be the coordinator or initiator

Distributed Event matching allows subscriber processes to specify events that they wish to receive. It also allows a process to publish notifications, which will be sent only to subscribers.

Differential GPS can be used to determine the geographic location of nodes, and optimize depending on latencies between nodes.

Folding@home - Coordination

- Grid Computing
- Support various Operating systems and hardware
- Client – Server architecture is used
- Message Passing Interface is used to optimize parallel computation
- Coordination is important because completed work units had to be compiled in a specific order



The impact of Folding@home

Molecular Biology

- Improved understanding of protein folding
- Improved understanding of Molecular dynamics
- Drug Discovery

Distributed Computing

Name	Users	PFLOPS	Type
SETI @ Home	5,000,000	1.09	Distributed System
Einstein @ Home	2,761,797	2.39	Distributed System
GIMPS	1,799,783	0.61	Distributed System
Folding @ Home	110,685	135.0	Distributed System
MilkyWay @ Home	27,408	0.85	Distributed System

Name	PFLOPS	Type
Summit	143.50	High Performance Computing
Folding@home	135.00	Distributed System
Sierra	94.64	High Performance Computing
Sunway TaihuLig	93.02	High Performance Computing
Tianhe-2A	61.45	High Performance Computing

Bibliography

“Distributed Computing”, https://en.wikipedia.org/wiki/Distributed_computing, accessed April 2019.

Day, John. *Patterns in Network Architecture: A return to Fundamentals.*Pearson Education, 2008.

Herlihy, Maurice. *“Blockchains from a Distributed Computing Perspective” Communications of the ACM,* Vol. 62, No. 2 (February 2019).

Van Steen, Maarten, and Andrew S. Tanenbaum. *Distributed Systems.* Maarten van Steen, 2017.

Tanenbaum, Andrew S. and David J. Wetherall. *Computer Networks*.Prentice Hall, 2011.