Define bandwidth.

1

Define throughput.

2

On a graph diagram of a distributed system, what is represented by the nodes on the graph?

3

Name some properties of edges that connect the nodes in a graph of a distributed system.

4

Why is it that even though connections between nodes can be implemented in different ways (such as wifi or Ethernet), they can be treated as the same?

5

Name the eight axioms of distributed systems.

6

Why is transparency desirable in a distributed system?

7

What is transparency of location? How can we achieve it?

8

Throughput measures the actual rate at which messages are communicated.

Bandwidth measures the maximum amount of data that can be communicated within a certain amount of time.

2

1

The type of connection (wired/wifi/mobile data etc).
The bandwidth.
The latency.

The physical nodes of the network (individual systems). Each can host multiple processes and resources.

4

3

·   *Latency is greater than zero.*
·   *Bandwidth is less than infinite.*
·   *Transport cost is greater than zero.*
·   *There is more than administrator.*
·   *The network topology can and will change.*
·   *The network is not homogeneous (the nodes and edges differ).*
·   *The network is not secure.*
·   *The network is not reliable.*

The implementation details of each connection is abstracted away by many layers of protocols.

6

5

An attempt to hide the need to know of where a specific resource is physically located.
Use DNS servers to map host names to IP addresses.

It allows us to design systems as though the distributed axioms were false.

8

7

What is transparency of migration? How can we achieve it?

9

What is transparency of relocation? How is it achieved?

10

What is transparency of replication. How is it achieved?

11

What is transparency of access? How is it achieved?

12

What is transparency of concurrency?

13

What is deadlock?

14

What is livelock?

15

What is transparency of failure? How is it achieved?

16

Transparency of relocation is when parts of the system move while they are being accessed. This is hard to mitigate, and is often a problem with mobile phone communications.

When a host moves location in the network, we shouldn't need to the details of the move.
The DNS architecture implements this, though if a resource keeps moving, then the route through the network and therefore the latency of the connection to the host is hard to predict.

10

9

Transparency of access is the ability to not care about how a node is implemented. This is often achieved using protocols and API's and middleware.

When there is more than one physical resource that does the same job, which one do we use? It is hard to achieve.

12

11

When two different processes are unable to progress since each is waiting for information from the other.

Different users shouldn't need to know that others are using the same resource and may be competing for its time. Atomic operations and enforcing consistency are ways to achieve this, but this can force users to wait on each other (deadlock, livelock etc).

14

13

Users should not know that a specific node has failed or has recently had downtime. Hard to ensure, since sometimes slow connections are indistinguishable from failed nodes.

When two processes change with respect to one another so that neither can make progress.

Define systems software.

17

Describe the start of the mainframe era.

18

What advances were made during the mainframe era?

19

What is the danger of multiprogramming?

20

What three things can help multiprogramming be effective?

21

How does the scheduler make decisions about programs?

22

In the beginning of the PC era, what was a typical PC like?

23

What happens to the price of PC's during the PC era?

24

- *Hardware was vastly more expensive than people.*
- *Only single user programs.*
- *Programs had to be written every time they were to be run.*

The underlying level of software which allows applications to perform their tasks efficiently. Examples include device drivers and the operating system.

18

17

One program could potentially have access to data in other programs, which could present security/compatibility concerns. Memory protection is used so that programs can only read and write stuff they own.

- *The ability to store programs was developed.*
- *Batch processing, where jobs would be written to a queue to be executed. Lost the ability to debug.*
- *Use interrupt handlers and buffers to allow the CPU and IO to work in parallel.*
- *Multiprogramming so that one program has the CPU and another has the IO to enable full resource utilisation.*

20

19

Give them priority levels.

- *Fairness policies that impose limits on how much resources jobs should use*
- *Schedulers that aim to minimise the time it takes to complete jobs, by reducing their response and turnaround time.*
- *Preemptive scheduling will temporarily stop jobs if they are hogging resources.*

22

21

Price falls due to commoditization and competition.

Everything was cut back with just a single user in mind; no timesharing, multiprogramming or protection in the OS, few resources such as memory and CPU power etc.

24

23

How does the usage of PC's change during the PC era?

25

What does WIMP stand for and when was it introduced?

26

What drove an increase in security focus during the PC era?

27

What are the four standard layers of the network protocol stack?

28

When did middleware start to become widely used?

29

In the device era, CPU cycles, storage space and power is all limited. How is this overcome?

30

Since there are millions of requests sent to data centres in the device era, how is the demand handled?

31

What is the aim of the operating system with regard to resource management?

32

Windows, Icons, Menus, Pointer

The WIMP system became common during the PC era.

They become used for more simple tasks such as word processing, rather than number crunching and enterprise processing.

Name	Example
Application Layer	`HTTP, POP, SSL`
Transport Layer	`TCP, UDP`
Internet Layer	`IP, ICMP, IGMP`
Link Layer	`ARP, Ethernet, DSL, ISDN`

The fact that sharing data over networks was becoming increasingly common, which exposed security risks. Authentication and authorisation methods were therefore developed.

Make use of connectivity to servers to offload computation somewhere else. This is called the cloud.

At the beginning of the web era. API's are used to provide a generic interface between the OS and programming languages so that system calls are no longer needed. The OS must support middleware rather than implementing functions itself.

To make the most efficient use of the available resources as possible.

Elastically scalable architectures made up of commodity hardware are developed along with middleware such as hadoop and map reduce in order to allow for massively parallel computation.

How is memory access controlled by the operating system?

33

Why is it important that a process only have access to isolated areas of memory?

34

Each process executes in a sequential manner, what is a disadvantage of this?

35

What is a **scheduling policy** with regard to operating systems?

36

Describe forking.

37

Describe threading.

38

When there are many independent, self-sufficient, autonomous, heterogeneous machines working together with spatial separation, what is required to facilitate effective working?

39

A significantly long interconnect (i.e. a ⬛⬛⬛⬛⬛⬛⬛) is not ⬛⬛⬛⬛⬛⬛.

40

So that processes cannot interfere with each others memory, because of a bug or malicious intent.

- The OS assigns a **unique identity** to each process.
- Each process is then assigned an **address space**
- The OS ensures each process P has a single address space A that is exclusive to that process.

34

33

A method of granting each process a time during which it is executed, enabling multiple processes to execute in a concurrent fashion, even though there may be only one physical set of resources.

If there was some task that took ages to do, say an API request over a slow network connection, then it might be sensible to execute other tasks (maybe preparing data for another API request) while the current task is executing (i.e. make the tasks concurrent). Due to the sequential nature of processes, this is not done.

36

35

When a new thread is created in a process:
- A new child process is created
- The address space is shared between the child and the parent.
- The parent and child processes are less isolated, and can therefore interact together easier, however, this can be dangerous and a source of bugs.
- Since the no memory is copied, threading is less expensive than forking.

When a process forks (through an OS system call) the following happens:
- A child process is created
- It is given a copy of the parent process's address space, though each address space is distinct.
- This causes two copies of the process to be active concurrently.

38

37

A significantly long interconnect (i.e. a full blown network) is not cost-neutral.

Message exchange over a network, best facilitated with a middleware API to abstract away some complexity.

40

39

Describe an SISD architecture.

41

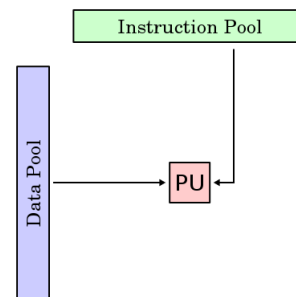*Describe an SIMD architecture.*

42

Describe a MISD architecture.

43

Describe a MIMD architecture.
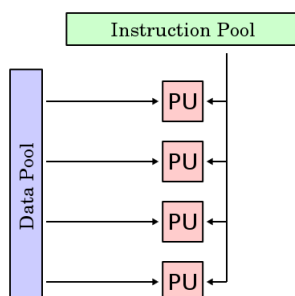
44

Describe the advantages of a MIMD architecture.

45
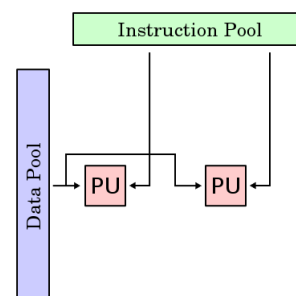
What architecture is this:

Instruction Pool

Data Pool

PU

46

What architecture is this:

Instruction Pool

Data Pool

PU

PU

PU

PU

47

What architecture is this:

Instruction Pool

Data Pool

PU

PU

48

A single instruction/program, multiple dataset architecture. This could be used to describe a computer with a GPU that is applying **the same** *transformation to many pixels at once. Access to data is* **not** *cost neutral since the data is being shared amongst different instances of a program/instruction.*

42

A single instruction/program, single dataset architecture. An example of this is an older, single core PC, where there is only data space (the main memory) and there is cost neutral access to it.

41

A multiple instruction/program, multiple dataset architecture. Here, a form of interconnect will bind together several self sufficient autonomous components. The interconnect isn't cost neutral, but the gain in functionality is great. This architecture is widely used in applications such as clouds, clusters, and even multi-core PC's.

44

A multiple program/instruction, single dataset architecture. Though not as useful as other architectures, a sample MISD use case is in safety critical systems where two independent processors could compute the same function on the same data and check their results against each other.

43

SISD

A MIMD architecture allows you to scale out instead of up. An architecture that lets you scale out is one where you can add more hardware (such as servers/hard drives) as opposed to upgrading existing hardware which is scaling up (i.e. increasing the RAM in a server).
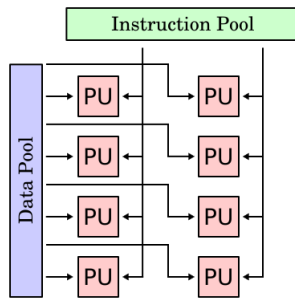
46

45

MISD

SIMD

48

47

What architecture is this:

49

MIMD architectures (aka �_____ architectures) often communicate over a _____ such as _____ that exists _____ to the architecture itself.

50

What does vBNS stand for?

51

What does OC stand for? What does OC-n mean?

52

What is the network protocol stack?

53

What are the five layers of the protocol stack?

54

Describe the job of the application layer.

55

Describe the job of the transport layer.

56

MIMD architectures (aka shared-nothing architectures) often communicate over a network such as the Internet that exists separately to the architecture itself.

MIMD

50

49

OC stands for Optical Carrier.

$$OC\text{-}n = n \times 54.84 Mbit/s$$

Very high performance Backbone Network Service.

52

51

$Application \leftrightarrow Transport$
$\leftrightarrow Internet$
$\leftrightarrow Link$
$\leftrightarrow Physical$

Layers of protocols that talk to each other to facilitate network connectivity.

54

53

To make sure that there is a logical channel for messages to flow through. E.g. TCP

To take care of specific application needs such as sending emails or HTTP requests.

56

55

Describe the job of the Internet layer.

57

What is the job of the link layer?

58

What is the job of the physical layer?

59

What is the hourglass model?

60

What is a packet switched network?

61

What is the job of a router in a packet switched network?

62

What are some differences between TCP and UDP?

63

What is packet encapsulation?

64

Making sure that there is a physical channel to pass data through. E.g. MAC

To make sure packets sent over the network reach their intended destination. E.g. IP

Very varied application layers at the top of the network stack (emails, streaming video etc), as simple protocols as possible in the middle of the stack, and very varied physical devices at the bottom (bluetooth, wifi, optical fibre etc).

To physically pass messages between networked components. Deals with all the physics etc. E.g. Wifi

To examine the header data in each packet and send the packet on to another router in the network that is closer to the packet's destination, while accounting for how saturated datapaths are in order to find an efficient route.

- *A network of networks formed by interconnecting routers.*
- *The message is divided into packets before transmission.*
- *The packets are reconstituted into the original message at reception.*
- *Packets may take varied routes while in transit, and be grouped with unrelated packages from other messages.*

Each layer of the networking stack wraps messages in a packet that contains information about the packet. By the time the packet has descended to the physical layer, then it may have many layers of headers and metadata.

- *TCP is connection oriented, UDP isn't.*
- *TCP is reliable and ensures that messages arrive at their destination, intact and in the right order. If the messages do not arrive, then it will be known. UDP only provides a corruption guarantee.*
- *UDP is more lightweight than TCP and is better for quick communication where the occasional loss or out of order packet isn't a problem.*

Why does packet encapsulation enable flexibility?

65

What does IPC stand for?

66

A successful communication between processes must consist of one process being _____ *coinciding with another process being* _____ *. This is non trivial with* _____ *.*

67

Describe Direct Message Exchange as a Distributed Architecture.

68

What is asymmetric DME?

69

What is a symmetric DME?

70

What is a MME architecture?

71

If a message in a MME architecture goes to more than one destination it is described as being a _____ *, if messages only go to one destination, then the system is described as being* _____ *.*

72

Inter-Process Communication

It doesn't matter about what information the packet is carrying, at each layer of the network stack, the callee layer has multiple choices of what protocol to use for the next level on the stack, and simply chooses the most convenient/suitable one.

66

65

In order to communicate, the sending process must be in a position to send and the receiving process must be in a position to receive messages. Upon receiving a message, a process may interpret it and then respond.

A successful communication between processes must consist of one process being ready to send coinciding with another process being ready to receive. This is non trivial with independent timelines.

68

67

Direct Message Exchange where no special roles are assigned to processes, also known as a peer-to-peer architecture. Event synchronisation is less simple with a symmetric DME than with an asymmetric one.

Direct Message Exchange where processes are assigned special roles such as a client or server in order to simplify the process. All the server has to do is wait for messages and all the client has to do is send messages.

70

69

If a message in a MME architecture goes to more than one destination it is described as being a public subscribe system, if messages only go to one destination, then the system is described as being point to point.

A Mediated Message Exchange architecture is one where inter process communication goes through a central middleware component (often referred to as message orientated middleware). Allows for event synchronisation to be decoupled in time since the middleware is always avalible.

72

71

What is RPC?

73

Why is middleware such as MME, RPC and RMI beneficial?

74

What is the function of the HEAD *HTTP verb?*

75

What is the function of the GET *HTTP verb?*

76

What is the function of the POST *HTTP verb?*

77

What is the function of the OPTIONS *HTTP verb?*

78

What does the 'store and forward' model mean?

79

What is the SMTP protocol?

80

They raise the level of abstraction at which we can build and engineer distributed systems.

A remote procedure call is similar to a message, though the data is passed as parameters to a method. It is often facilitated by middleware such as CORBA. *When OOP principles are involved, RPC is called* **Remote Method Invocation (RMI)***.*

74

73

Gets a resource from the server.

Provides metadata about a resource. Just like a GET *request, except the body of the request is not returned. Used to let clients know if cached data has changed.*

76

75

Returns the list of commands supported by the server.

Submits data to the server. The state of the server may change as a result of a POST *request.*

78

77

The Simple Mail Transfer Protocol is a method for sending email. It is a stateful, connection based protocol, so a connection to the server must be opened before the message can be transferred, and then closed again after.

When a message needs to be transmitted along a network, but not all the nodes along the path may be active at the same time. Consequently, the message is stored by each node until it has been confirmed to have been received by the next node.

80

79

Describe 'reduction to absurdity'.

81

With a Lamport clock, how is the clock of A affected if B sends A a message?

82

Describe the purpose of a mutex.

83

What is a critical section?

84

Describe the two-phase commit.

85

Describe the bully algorithm.

86

Describe Cristian's algorithm?

87

Describe the Berkly algorithm.

88

```
if LCy < (LCx + 1):
    LCy = LCx + 1
```

Manchester University: a place where you get a score of 29/30 and yet still 78.7This is a type of argument where an idea is shown to be false by showing that an absurd result can be obtained by the acceptance of the idea.

A critical section is a sequence of actions that should execute in an atomic matter. That is to say that if the critical section does not succeed, then the state of the system should be exactly the same as when the critical section started.

A mutex ensures that two processes do not enter into a critical section of their execution at the same time and therefore prevents race conditions and enables the safe sharing of resources.

1 *A node wants to be a coordinator, so it sends a message to all the other nodes with it's number.*
2 *All the other nodes will only reply if they have a higher number. If they do, they will start another election.*
3 *The process continues until a node receives no replies, and is therefore the coordinator, at which point, it will inform all the other nodes of this.*

1 *A coordinator node wishes all of the nodes to perform an action.*
2 *It sends a request to all of the other nodes.*
3 *The other nodes attempt to perform the action, and reply with* commit *or* abort *depending on whether they to performed the action.*
4a *If any one of the nodes replied* abort, *then the coordinator node will reply* global_abort *and all the nodes will roll back their state to that of the start of the transaction.*
4b *If all the nodes replied* commit, *then the coordinator node will send* global_commit *and the nodes will retain their current state.*

1 *A master node is chosen.*
2 *The master polls the other nodes for the time.*
3 *The master observes the round trip time for each state, and estimates the time of each client.*
4 *The master then averages out the clock times and sends each node a message containing the difference that it is from the average.*

In order to synchronise the clocks of two systems, one system S will send it's time to another system C. After receiving the request from S, C will then reply with it's own clock time. When S receives this, then it will set its clock to be the time that C sent added to the round trip time (discernible from C's reply) divided by two.