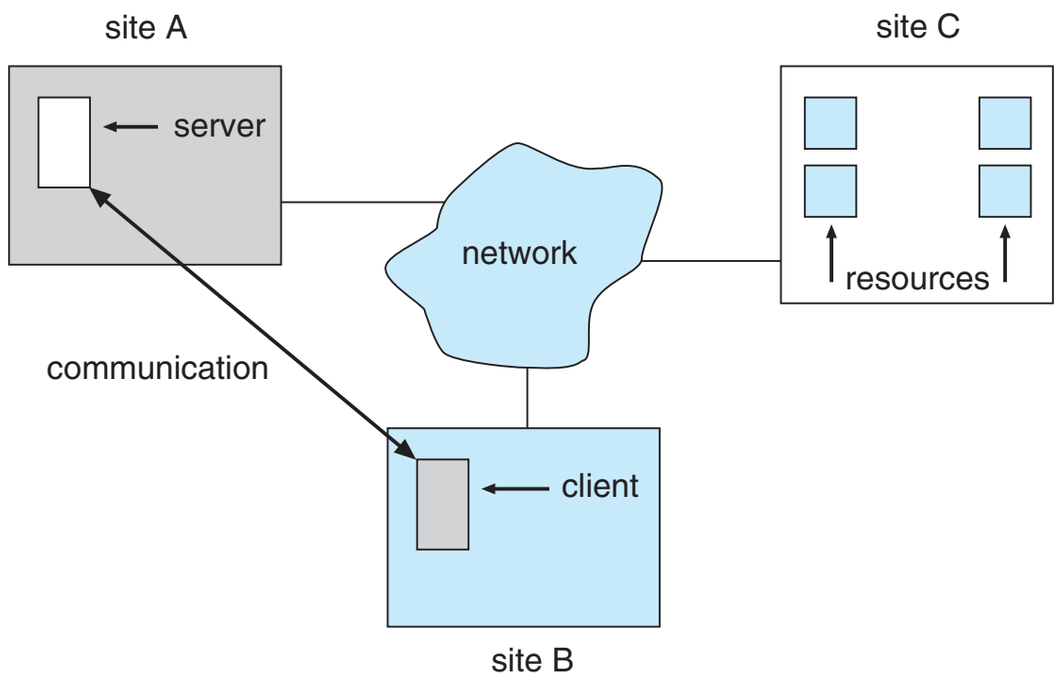
Networks and Distributed Systems

Networks and Distributed Systems

- Ch19.1 Advantages of Distributed Systems
- Ch19.4 Network and Distributed Operating Systems
- Ch21.6.2.7 Remote Procedure Calls
- B.6.2.7 Remote Procedure Calls
- Wikipedia – Two-Phase Commit Protocol

Distributed Systems

- A **distributed system** is a collection of loosely coupled nodes interconnected by a communications network
- Nodes variously called processors, computers, machines, hosts
  - **Site** is location of the machine, **node** refers to specific system
  - Generally a **server** has a resource a **client** node at a different site wants to use



Distributed Systems

A distributed system is ...

"one on which I cannot get any work done because some machine I have never heard of has crashed".

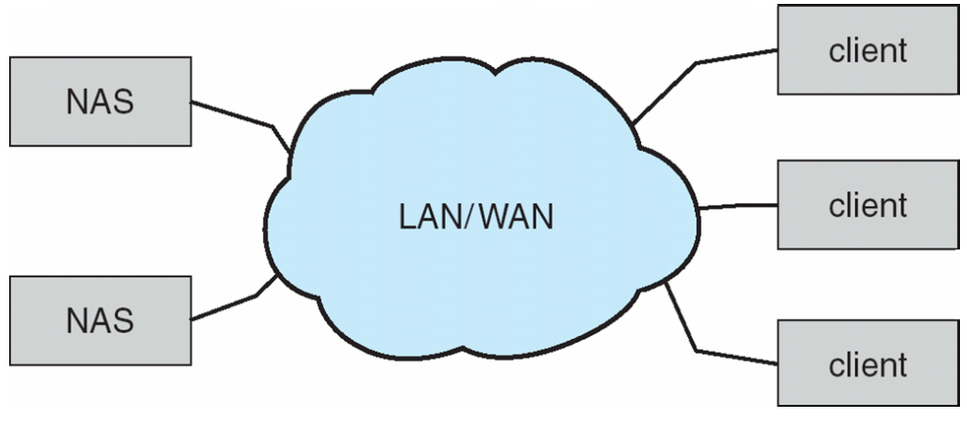
- Loosely-coupled
- network connection
- could be different OSs, or different parts of the OS
- processes must communicate via messages

Advantages:

- More work can get done
- Ability to share devices, programs and data
- Greater reliability
- Easier to expand

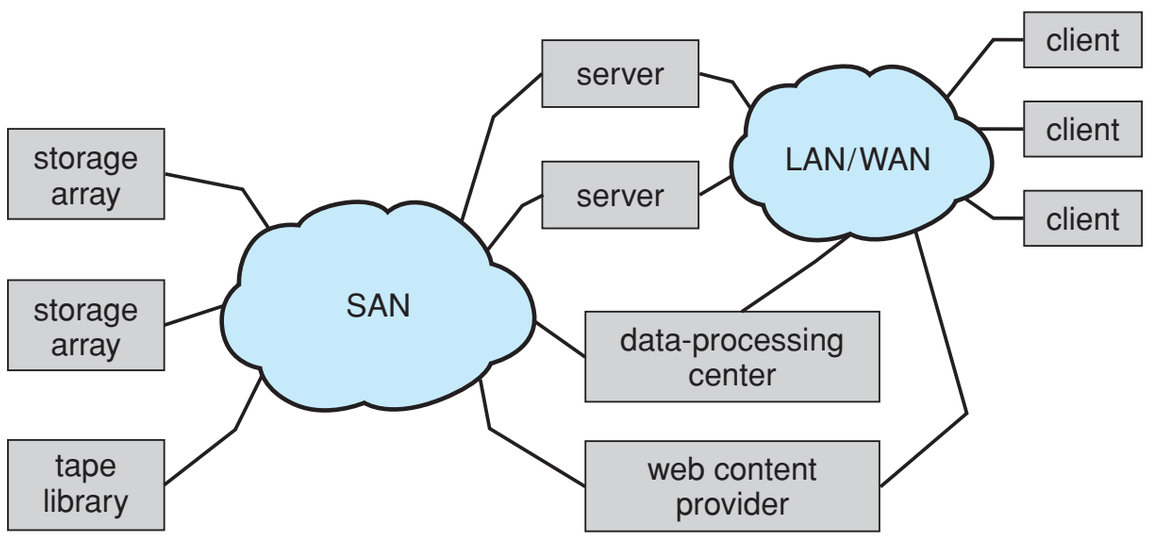
NAS and SAN

- Many distributed file systems are provided by Network-Attached Storage or Storage Area Networks.
- Network-Attached Storage (Ch11.7.2)
  - File storage made available over a network rather than a local connection
  - Can be as simple as a device which deals with a protocol such as NFS over the network.
  - Implemented via remote procedure calls (RPCs) between host and storage over typically TCP or UDP on IP network
- NAS devices just act as servers but they are designed to do the job efficiently and safely. The actual storage may be in a RAID setup.
- They are controlled and configured over the network.
- You can think of them as servers.



NAS and SAN

- Storage Area Networks (Ch11.7.4)
  - Use a different type of protocol, such as Fibre Channel. These protocols provide a specialised high speed network specifically for accessing storage.
  - The main difference is that SANs deal with blocks rather than file systems. The client side deals with the file system, the SAN provides the block storage.
  - Common in large storage environments Multiple hosts attached to multiple storage arrays – flexible
  - You can think of them as attached to servers, sort of like a really really large, very flexible disk devices.
- There are also SAN-NAS hybrids.



Question

- Which of the following would be slower in a distributed file system when compared to a direct attached file system?
  1. Opening files
  2. Reading files
  3. Writing files

Two Phase Commit Protocol

With distributed systems we want to ensure that if something goes wrong at one site we don't end up with inconsistent data.

A "transaction" is some event that has to be completely successful or not done at all (atomic).

We need stable storage – usually replicated on several devices – can be done with two copies.

- make the change to one (check for success)
- make the change to the other (check for success)
- if ever the copies disagree copy the original data from the second back to the first

2PC – transaction coordinator and all sites involved in a transaction have stable storage logs.

All transactions can be undone and redone safely.

Log entries and messages

Commit request phase (started at the end of the transaction)

<prepare> - started the protocol, sent to all sites (query)

<ready> - recorded and returned if ok, <abort> if not ok (voting)

Commit phase

<commit> - if all reply in time, sent to all sites

<abort> - sent by coordinator to all sites if something went wrong (they must rollback)

Network-oriented Operating Systems

Network OS

- Communications layer on top of a normal OS.
- Possibly different OS.
- User is aware of different machines.
- Some can copy files across the network but not share them. e.g. ftp
  - In this case the file location is explicitly known.
- Others can share files but the location is still part of the name.

Distributed OS

- Aim to have the system look like one machine.
- There is no difference (except speed) between accessing local and remote resources (location transparency).
- The Distributed OS can move resources and processes (migration transparency).

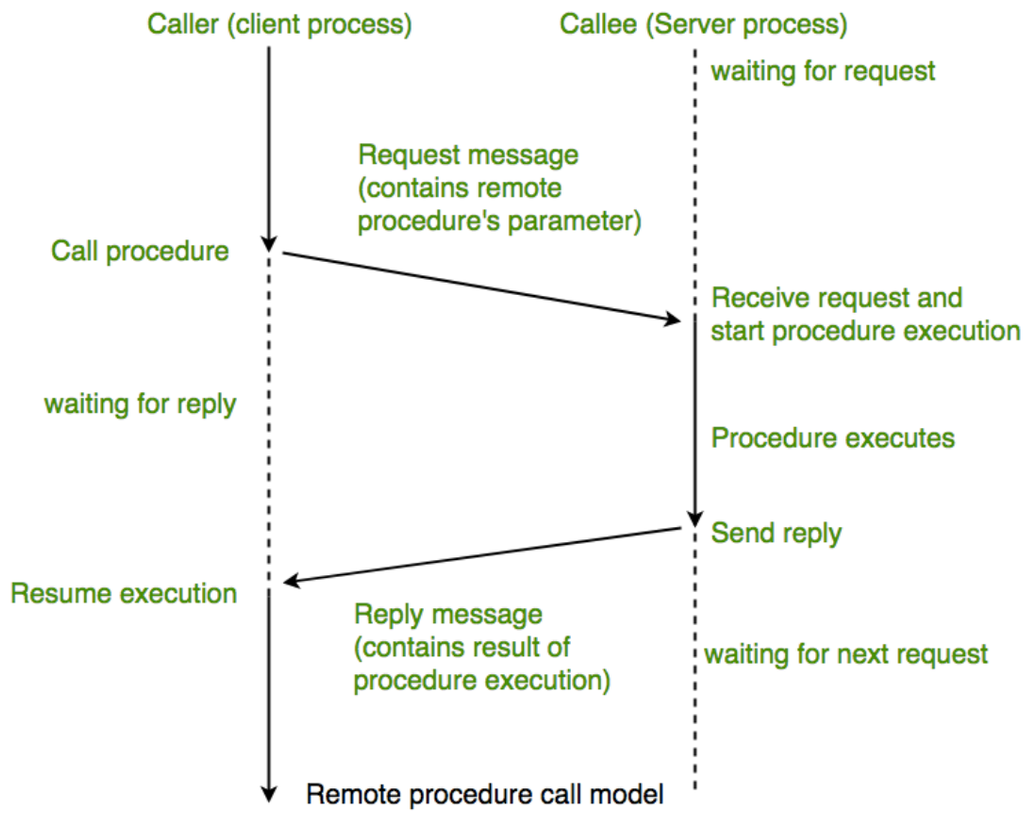
Distributed OS

- **Data Migration** – transfer data by transferring entire file, or transferring only those portions of the file necessary for the immediate task
- **Computation Migration** – transfer the computation, rather than the data, across the system
    - Via remote procedure calls (**RPCs**)
- **Process Migration** – execute an entire process, or parts of it, at different sites
    - Load balancing – distribute processes across network to even the workload
    - Computation speedup – subprocesses can run concurrently on different sites
- Consider the World Wide Web

Remote Procedure Calls (RPC)

Birrell & Nelson 1984

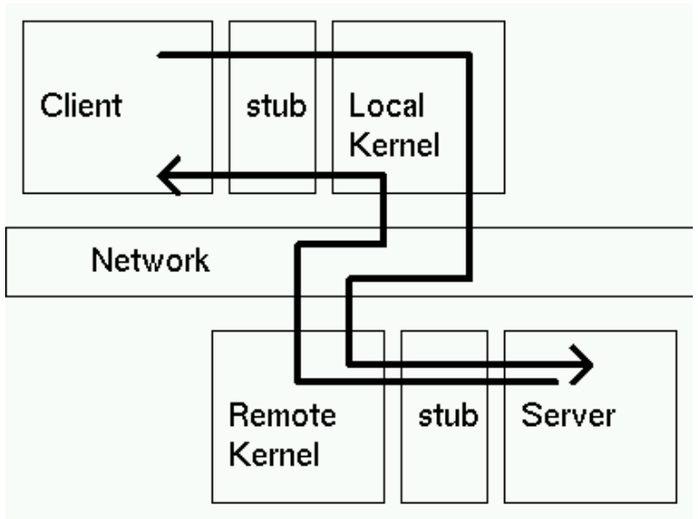
- Hide the message passing system so that it looks like a series of procedure calls.
- Most requests for service wait until the request is fulfilled – semantically just like a procedure call.
- Programmer doesn't have to package and unpackage data in the messages.

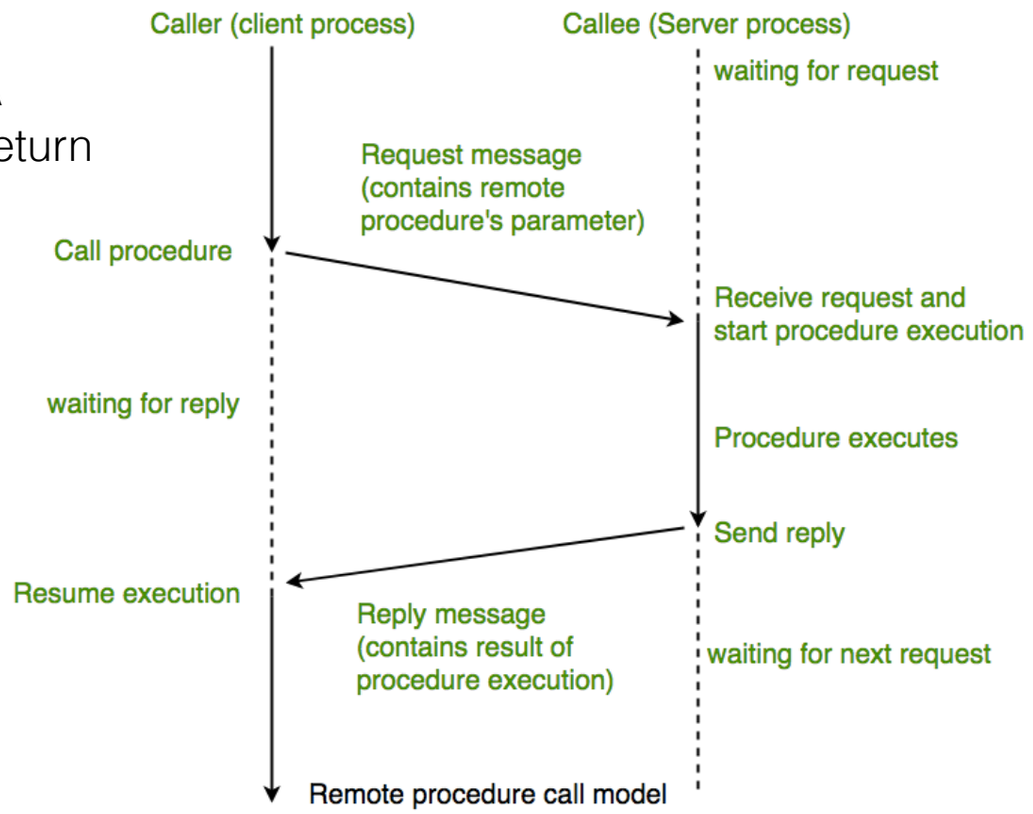


Client and Server Stubs

1. Client makes ordinary procedure call to the local stub.
2. Stub marshals parameters (may need to locate server as well).
3. Stub sends request via local kernel.
4. Remote kernel passes request to Server stub.
5. Server stub unpacks(demarshal) message request and parameters.
6. Makes ordinary procedure call to Server.
7. And then vice-versa.

- The stubs at both ends need to be constructed from the same interface specification - to ensure consistency.
- Care has to be taken about different versions of the service. A different version may take slightly different parameters or return different types etc.





RPC Messages

Messages are highly structured

- What procedure to execute
- Parameters
- Version number (service may survive a long time and code may be written to different versions)
- Timestamp (could be used for synchronization purposes)
- Source address
- Where to send the results
- Possibly the type of machine the request comes from

Finding the server

- Include port numbers at compile time or
- Have a binder or rendezvous/matchmaker service
- Server usually registers with a binder or name server
- Client end sends the request to find the server
- Multiple servers – the binder can spread the load
- Binder can periodically check on servers - deregistering those that don't respond
- Binder could do the security work
- Otherwise servers have to check each call

Marshalling

- Heterogeneous networks.
  Data formats in the different machines may be different.
- ASCII, Unicode, EBCDIC for strings
- Big or little endian for integers
- Different floating point formats
- Stubs need to know about this.

<u>Different solutions</u>

- client stub converts to the server format
- server stub converts from the client format
- convert to and from a canonical format
  - no one needs to know other machines formats
  - e.g. NFS's XDR - external data representation

Process Migration

- Moving a process from one site to another while it is running.
- Why would we like to do process migration?
  - To enable us to do proper load balancing.
    - If the process can be subdivided we can increase performance by having different parts running simultaneously on different machines.
    - To move the process closer to the resources it is currently accessing.
    - To move the process closer to the user.
    - To enable us to keep a process going when the site it is executing on has to be taken down.

What do we need?

We need location and migration transparency of

- processes
- resources used by the process

What defines a process?

- PCB
  - Resources
    - files
    - communication channels
    - memory
    - devices - including windows, keyboard, mouse
  - Threads
- Need some compatible machine (or virtual machine) architecture.
- Internal and external reference problems
- References to resources within the program.
- References to the process from outside e.g. other processes communicating with it.

How can that be done?

- Need a way of referring to all resources indirectly via global tables (like we did with our distributed file systems).
- We can extend the ideas of a distributed file system to refer to other objects, including processes.
- All process identifiers have no host information in the identifier.

e.g.

- A process table keeps track of which site each process is running on.
- When the process is moved the table is updated.
- Caching of information can be used for efficiency but we need ways to recover when the cache data is out of date.
- Not all processes need to be stored in this table.
  - Processes specific to a site which are not visible away from the site.

Doing the Migration

Minimise the amount of down time

- Process must be stopped at some stage
- Stop, copy, notify
- How much do we copy?
  - Only the working set
    - get the remaining pages by demand paging
    - Can’t be used if the host is going down.
  - Everything, but don't stop the process
    - then copy pages which were dirtied during the copy
- Both approaches only stop the process while the working set is moving.

Current Uses

- In reality process migration is not used for load balancing.
- It is too expensive.
  - Most processes only run for a few seconds.
  - Transferring a process can easily take a few seconds.
- It is still useful when a machine needs to be closed down for maintenance and it has running processes which we don't want to kill.

Another use - idle workstations

- Move processes when no longer idle.
- Generally load balancing is only done when a process starts
  - or when it has to move.
- Where is the best place to run this process?
- The textbook also talks about Computation Migration – this means sending messages (or RPCs) to get work done on another site.

Before Next Time

Read from the textbook

- Ch9.1 Background
- Ch9.2 Contiguous Memory Allocation
- Ch9.3 Paging
- Ch9.6.1.1 Segmentation

Interesting read on Windows 8 memory management

- http://arstechnica.com/information-technology/2011/10/how-windows-8s-memory-management-modifications-make-for-a-better-user-experience/