Network Programming and Applications

- *Previously*, we looked at some of the services that a network provides
- *Now*, we look at computer networks from a application perspective
- A data network is a *passive* entity, it neither *generates* nor *understands* the data being sent
- Applications that use computer networks operate in *pairs;* the *client* and the *server.*
- Typically these applications run on hosts that are remote from each other.

Network Programming and Applications

- Between the host machines there is a network across which the data must travel.
- Server applications run on *server-class machines*:
  - Sometimes these are mistakenly referred to as *servers*,
  - *Servers* are applications and the machines on which they run are server-class machines.
- Client applications can run on any machine.

Network Programming and Applications

- Server applications play a vital role in modern networked applications.
- They are continuously running waiting for contact from Client applications.
- Some texts use the analogy of two people communicating over the telephone network to represent client-server communications:
  - However, the analogy falls short as there are some unique challenges when writing client and server applications which we will explore in the lab.

Making Contact - Client-Server Interaction

- How does an application know when a message has arrived?
- It's not so straight forward with applications:
  - Before any inter-application communication can take place an application must interact with its local protocol software to notify it to expect *messages* of a specific type,
  - The application then *waits passively* for contact from other applications

Client-Server Interaction

- The protocol software examines incoming messages and passes *matching* messages to the application.
- The application that is continuously running, passively waiting for contact from other applications is called a *server.*
- The application that actively initiates contact with a server is called a *client.*
- This is known as the *client-server* paradigm.

Characteristics Of Clients

- **In general**, client software has the following characteristics:
  - It provides general purpose computational functionality to a user but on occasion it becomes a *client application*
  - It is invoked directly by a user and executes for one session
  - It runs locally on a user's personal computer
  - It actively *initiates* contact with a server
  - It can access multiple *services* but can only contact one *server* at a time
  - It does not require specialised hardware or a sophisticated operating system

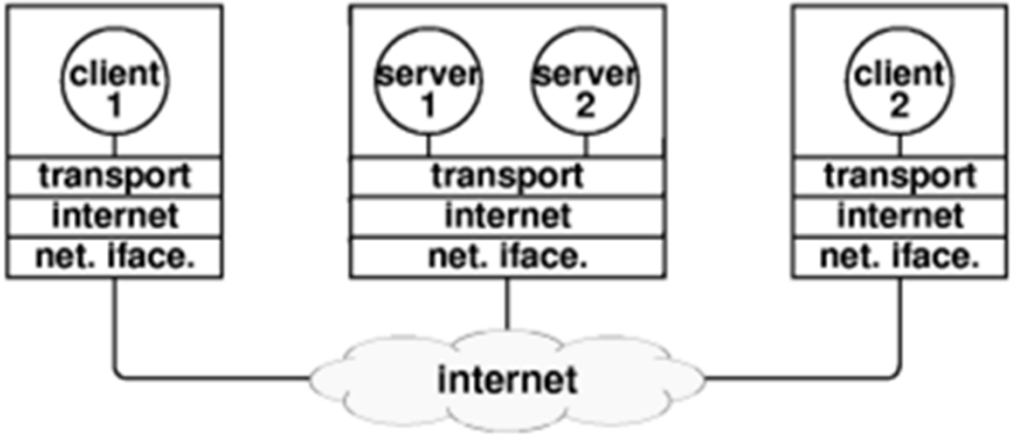
Characteristics Of Servers

- **In general**, server software has the following characteristics:
  - It is a special-purpose, *privileged* program dedicated to providing one *service*
  - It can handle multiple remote clients simultaneously
  - It invoked automatically upon boot-up, and executes through many sessions
  - It runs on a shared computer
  - It *passively* waits for and accepts contact from *arbitrary* remote clients
  - It requires powerful hardware and a sophisticated operating system

Multiple Services On One Computer

- Some server-class machines run *multiple* clients and servers simultaneously
- These computers have an operating system that allows multiple application programs to execute *concurrently* (e.g., UNIX or Windows).
- For each service offered there must be an associated server program executing
  - e.g. a single computer might run a *file server* and a *World Wide Web server*
- The following diagram illustrates this

Concurrent Server-class Machine



Multiple Services On One Computer

- With *multiple* servers running how can a client identify a particular server *unambiguously*?
- Some form of addressing is required:
  - Each server is assigned a unique *identifier*
  - *Clients* and *servers* use this identifier in all interactions
- The *communications paradigm* is as follows:
  - The *server* application starts execution first:
    - It registers its identifier with the local protocol software
    - It then waits for contact from clients
  - *Clients* contact *servers* by specifying the server's *location* and *unique identifier*
  - The client and server exchange messages and terminate communication