The X Windows system
Brief X history

- 1984: X was developed by the Athena project at MIT
- ‘X Consortium’ formed
  - Handles development and distribution of X
  - X specification made available
- XFree86 formed
  - [www.Xfree.org](http://www.Xfree.org)
    - A global volunteer organisation which produces XFree86®, a freely redistributable open-source implementation of X.
    - It lost much popularity and most of its developers when it adopted an unpopular license for version 4.4.0.
- 2004 X.org
  - Perceived problems with the XFree licence, X.org implementation ([www.x.org](http://www.x.org)) is now favoured
- X consists of client libraries used to write X applications ("clients"), and an X server responsible for the display.
- Client and server communicate via the X protocol, which allows to run clients and server on different computers.
Architecture /1

• The X Protocol defines a client-server relationship between an application and its display
• The X server maintains exclusive control of the display
• Applications only have to know how to communicate with X not with the actual graphics chips.
  – But a programming library would give us the same result
  – Yes but X lets remote clients ask for stuff to be drawn on the local screen
User at home wants to run an mathematically complex application which is not on his (old) home PC. Using a N/W connection he connects to the powerful running the application and tells it to redirect X requests, made by the application, to be sent across the N/W to his PC. Thus the remote machine does the number crunching and his machines X server accepts the requests and displays the GUI locally. The remote app. is a (remote) X client to the (local) X server.
Architecture /2

• The X Server
  – Runs on the local machine.
  – Accepts and demultiplexes network (or local IPC) based X client requests and acts upon them.

The X server has seamless access to distributed applications.
An X server communicates with various client programs.

The server accepts requests for graphical output (windows) and sends back user input (keyboard, mouse).
Confusion

• This client-server terminology — your terminal is the "server", the remote applications are the "clients" — often confuses new X users, because the terms appear reversed.

• X takes the perspective of the program, rather than the end-user or the hardware:
  – the remote programs connect to the X server display running on the local machine, and thus act as clients;
  – the local X display accepts incoming traffic, and thus acts as a server.
Architecture /3

- The X server therefore:
  - displays drawing requests on the screen.
  - replies to information requests.
  - reports an error in a request.
  - Manages the keyboard, mouse and display device.
    - Multiplexes keyboard and mouse input onto the network (or via local IPC) to the respective X clients. (X events)
  - creates, maps and destroys windows.
    - writes and draws in windows.
  - So the X server is responsible for visual output and data input from the user.
The X Client

- Essentially an application written with the aid of libraries (i.e. Xlib, Xt) that take advantage of the X Protocol.
- sends requests to the server.
- receives events from server.
- receives errors from the server.
The client

- A provides a library, called Xlib which handles all low-level client-server communication tasks. The client has to invoke functions within X to get work done.
Protocol Messages

– Requests
  • X clients make requests to the X server for a certain action to take place. i.e.: Create Window
  • To enhance performance, the X client normally does not expect nor wait for a response. The request is typically left to the reliable network layer to deliver.
  • X requests are any multiple of 4 bytes.

– Replies
  • The X server will respond to certain X client requests that require a reply. As noted, not all requests require a reply.
  • X replies are any multiple of 4 bytes with a minimum of 32 bytes.
Protocol Messages /2

• **Events**
  – The X server will forward to the X client an event that the application is expecting. This could include keyboard or mouse input. To minimize network traffic, only expected events are sent to X clients.
  – X events are 32 bytes

• **Errors**
  – The X server will report errors in requests to the X client. Errors are like an event but are handled differently.
  – X errors are the same size as events to simplify their handling. They are sent to the error handling routine of the X client. (32 bytes)
Design of an X server

• Depends greatly upon the platform (hardware) and operating system on which it is implemented.
  – As the capabilities of the underlying technologies increases, the power and capability of the X server also increases.

• Device Dependent Layer
  – It is this layer that is responsible for localizing the X server to the native environment, be it Windows NT or Solaris.
  – This layer swaps bytes of data from machines with differing byte ordering. Byte ordering (MSB and LSB) is noted in each X request.
  – This layer hides the architectural differences in hardware and operating systems.
  – Maintains device driver dependencies for keyboard, mouse and video.
X in service

- The X server communicates with the host operating system's kernel to drive input and output devices, with the exception of graphics cards.
  - These are generally managed directly by XFree86, so it includes its own drivers for all graphic cards a user might have.
  - Some cards are supported by vendors themselves via binary-only drivers.
  - X has supported (some) accelerated 3D graphics cards
  - It is also possible to use XFree86 in a framebuffer device, which in turn uses a kernel graphics card driver.
Difference with Windows

• X provides for the display and management of graphical information, much in the same manner as Microsoft's Windows

• The key difference is in the structure of the X Protocol.
  – Whereas Windows simply displays graphical applications local to the PC,
  – the X Protocol specifies a client-server relationship
X is basic

• X deliberately contains no specification as to application user interface, such as buttons, menus, window title bars and so on.

• These are provided by user software, such as window managers, GUI widget toolkits and desktop environments, or application-specific GUIs, such as point of sale.

• As such, the "typical" X interface has varied tremendously over the years.
Window Managers

- X doesn’t do everything
- It's another program's responsibility to manage onscreen space
  - The window manager

The window manager is just another (special) client to the X server with special privileges. The window manager manages other clients on screen look/location etc as well as providing some mechanism to start applications.
Window Managers

Window Managers

- FVWM
- FVWM95
- AfterStep
- Sawfish
- AMI
- Metacity
- icewm
- Enlightenment
- WINDOW MAKER
- FLUXBOX
- Others
- OLVWM
- MWM
- WM2
- blackbox
Toolkits

• Using the raw Xlib is ugly stuff.
• Higher level libraries are available – called toolkits or widgets libraries
  – Widgets include menus, picture areas, scrollbars etc
• There are lots of different widget libraries
  – Gtk (The GIMP toolkit or GNU toolkit)
    • Used with GNOME desktop
    • Feature rich, lightweight, free
  – Qt is another with some licencing issues
    • Used extensively in the KDE desktop
Desktop Environments

Desktops help to standardize the look/feel of the apps – something the users of Windows expect. KDE mandates a particular window manage (KWM) but GNOME simply has preference (Sawfish)
KDE
GNOME
Additional Reading

• Slides complied from material in
  – “X Window System architecture Overview HOWTO”, by Daniel Manrique
  – www.X.org
  – www.wikipedia.org