Announcements

- **No class on Tuesday May 12**
- Final is May 20, 1996 1:30-3:30 PM
- Reading: none
- Project #5
  - due Wed. May 13 (in section)

Display and Window Management

- The screen is a resource in a workstation system
  - multiple processes desire to access the device and control it
  - OS needs to provide abstractions to permit the interaction
- Services
  - protection
  - windows
  - multiplex keyboard and mouse
  - configuration and placement
- Issues
  - how to get good performance and remain device independent
  - how much policy to dictate to users

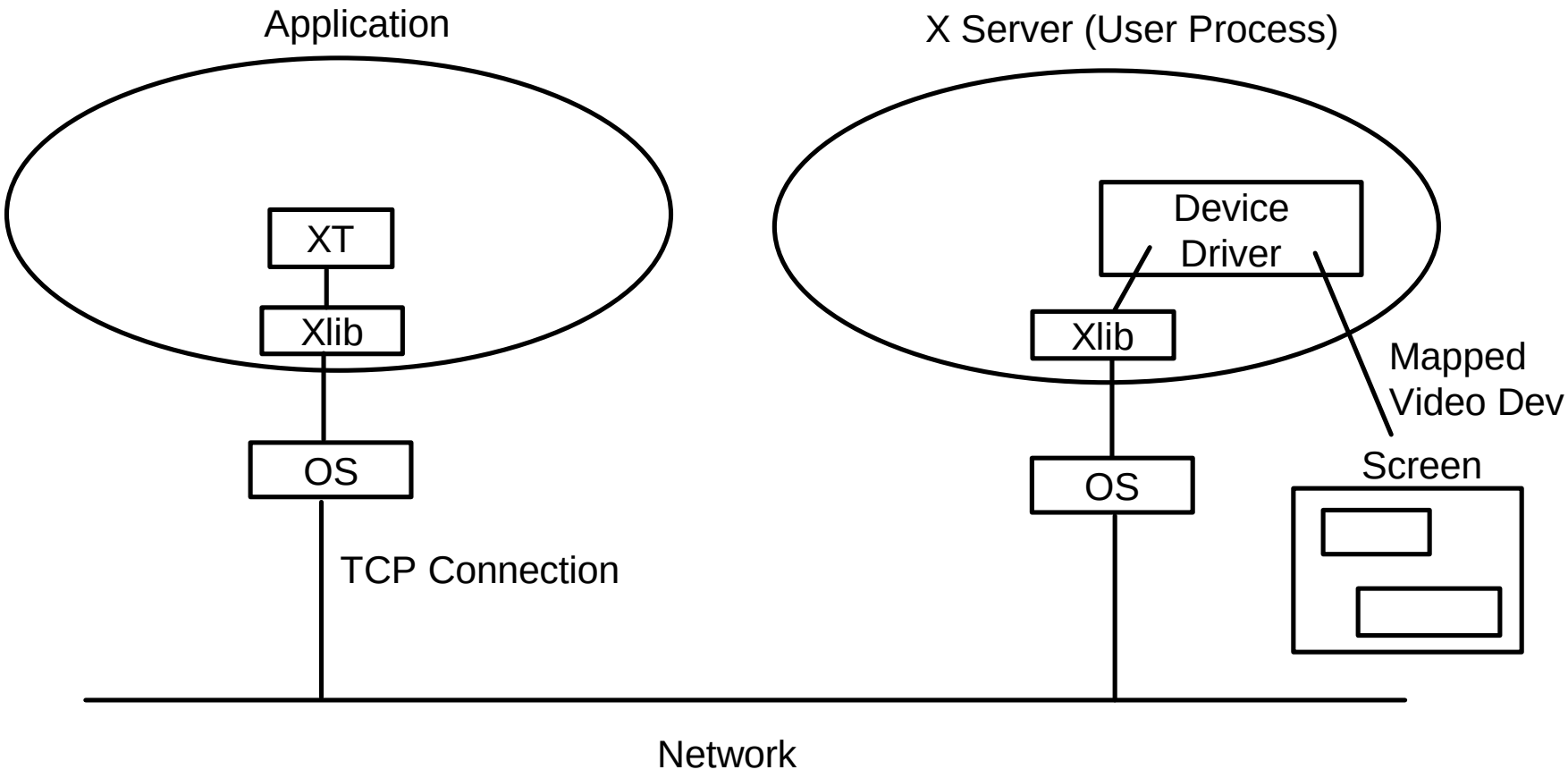
X Window System

- Designed for mid range workstations
  - very little policy
  - supports network display services
    - applications can run one place and display another
    - server is the display
    - clients are programs that contact to the server
  - basic protocol called X11r6
  - event based programming model
    - next event loop in application
  - typical requests
    - create window, draw line, draw circle, display text
  - typical events
    - key pressed, mouse moved, window (or part) now visible

X Libraries

- Programming raw X is tedious
- Many libraries exist to make it easier
  - libraries are linked into applications
  - X toolkit
    - object oriented interface plus widget library
    - widgets: buttons, menus, text, lists, etc.
    - provides main message loop
  - Motif
    - like X tool kit but “standardized”
    - more stylish look and feel
  - Tcl/Tk
    - Tk is the X interface (sort of Motif like)
    - TCL is a language for describing applications

X Windows



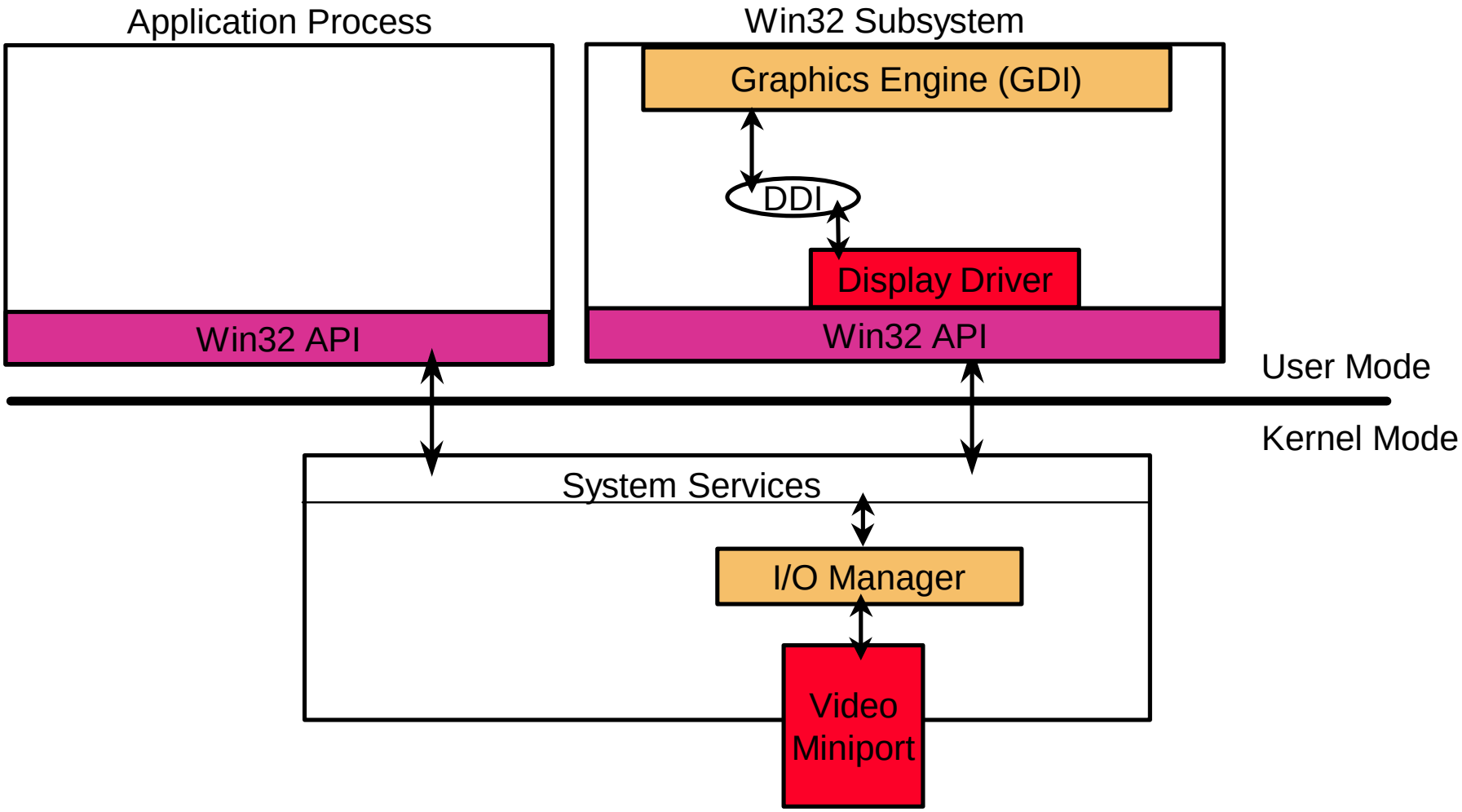
X Window Security

- Sever can limit what machines and users can connect and create windows
  - uses normal network based security protocols
  - also has a simple mode based on host names
- Window protection
  - can restrict access to only those windows the process has created

Windows (NT 3.51)

- Kernel exports a mapped device for video
- User Process (Win32) provides
  - screen protection
    - each process has a message queue for its events
  - Win32 API Windows services
    - dialog boxes
    - graphics primitives
  - Programs using API must be on the same machine

NT (3.5) Display Drivers



My Research Interests

- Parallel Computing
  - There are limits to how fast one processor can run
  - solution: use more than one processor
- Issues in parallel computing design
  - do the processors share memory?
    - is the memory “uniform”?
    - how do processors cache memory?
  - if not how do they communicate?
    - message passing
    - what is the latency of message passing

Parallel Processing

- What happens in parallel?
- Several different processing steps
  - pipeline
  - simple example: grep foo | sort > out
  - called: *multiple instruction multiple data* (MIMD)
- The same operation
  - every processor runs the same instruction (or no-instruction)
  - called: *single instruction multiple data* (SIMD)
  - good for image processing
- The same program
  - every processor runs the same program, but not “lock step”
  - called: *single program multiple data* (SPMD)
  - most common model

Issues in effective Parallel Computation

- Load balancing
  - every processor should to have some work to do.
- Latency hiding/avoidance
  - getting data from other processors (or other disks) is slow
  - need to either:
    - hide the latency
      - processes can “pre-fetch” data before they need it
      - block and do something else while waiting
    - avoid the latency
      - use local memory (or cache)
      - use local disk (of file buffer cache)
- Limit communication bandwidth
  - use local data
  - use “near” data (i.e. neighbors)

My Research:

- Given a parallel program and a machine
- Try to answer performance related questions
  - Why is the programming running so slowly?
  - How do I fix it?
- Issues:
  - how to measure a program without changing it?
  - how do you find (and then present) the performance problem, not tons of statistics?
- Techniques:
  - dynamic data collection
  - automated search
  - analysis of process interactions

My Research (I/O):

- Given lots of data to access, and lots of disks
- How do you make effective use of these disks?
- Questions:
  - What should I/O look like?
    - virtual memory
    - file pointer based I/O
    - direct I/O
  - Where should the data be placed?
    - central servers vs. distributed to each node
    - how do improve data locality
  - What information can the application provide?
    - hints about future access patterns?
    - what data is going to be re-used?