Software

- There are many kinds of software
- System software
  - Operating system (Windows, Linux, Solaris, etc.)
  - Device driver (for printer, network card, etc.)
  - Compiler (gcc)
  - Library (DLLs)
- User-level software
  - E.g., simulator, word processor, spreadsheet, game, etc.
- Internet software
  - Google
  - YouTube
  - Etc.

Software Faults/Errors

- We all know that software has bugs
- Types of bugs (or errors/failures that are due to bugs)
  - Incorrect algorithm
  - Array bounds violation
  - Memory leak (C, C++, but not Java)
    - Allocating memory, but not deallocating it
  - Reference to NULL pointer (C, C++, but not Java)
  - Incorrect synchronization in multithreaded code
    - Allowing more than 1 thread in critical section at a time
    - Blocking when holding a lock
  - Inability to handle unanticipated inputs
- Any other bugs that we recall from the Oppenheimer paper ("Why Internet Services Fail")?
**Operating System Errors**

- "An Empirical Study of Operating System Errors" (Chou et al., SOSP 2001)
- Studied 21 different versions of Linux over 7 years
- Used static compiler analysis to find bugs
  - Check for NULL pointers
  - Do not block a thread when interrupts are disabled or lock is held
  - Several other checks, including array bounds checking
- Discovered that vast majority of bugs are in drivers
  - Some also in:
    - arch/i386/ (architecture specific code)
    - net/ (network code)
    - fs/ (file system)
  - Why are drivers a primary culprit?

**OS Bug Characteristics**

- Bugs tend to be in larger functions
  - More complexity → more chance of a bug
- Tend to be clustered in small number of files
  - A bad programmer tends to mangle a small number of inter-related files
- Tend to be more prevalent in newer code and code that is less thoroughly integrated and tested
  - Code hacked onto the side of the main code is problematic
  - Helps to explain why drivers are so buggy

**Software Failures**

- What happens if we exercise a software bug? What kind of failures can occur?
- User-level software
  - Incorrect data
  - Livelock/deadlock (can escape with Control-C)
  - Exception that triggers OS to kill process
    - Segmentation fault
    - Bus error
- Operating system software (including device drivers)
  - Livelock/deadlock (may be able to escape with Ctrl-Alt-Delete)
  - Crash and reboot
  - Incorrect I/O

**Software Aging**

- "Proactive Management of Software Aging" (Castelli et al.)
SPN Example From Software Aging

Models system with \( n \) nodes, initially all up. Tokens represent nodes.

- \( P_{\text{up}} \): Node in up state
- \( P_{\text{sysfail}} \): System in failure state
- \( P_{\text{nodefail1}} \): Node 1 in failure state
- \( P_{\text{nodefail2}} \): Node 2 in failure state
- \( n \): Number of nodes
- \( t_{\text{common mode-fail}} \), \( t_{\text{node-repair}} \), \( t_{\text{failprob}} \), \( t_{\text{Prob}[c]} \), \( t_{\text{Prob}[1-c]} \), \( t_{\text{node-repair}} \), \( t_{\text{Prob}[c]} \), \( t_{\text{Prob}[1-c]} \)

- \( g_1 \): true, if token in \( P_{\text{sysfail}} \)
- \( g_2 \): true, if token in \( P_{\text{up}} \)
- \( g_3 \): true, if \( a \leq n \) nodes down

Figure 22 from Castelli et al.