A Few Classics

Phil Gibbons

15-712 F15

Lecture 2

Today's Reminders

- **Enroll in Piazza (26 have as of 2 pm)**
- **Drop/Waitlist**

Waitlist Status

- **As of 2 pm: 30 registered, 2 more let in, 4 on waitlist**
  - Thus 1 slot still open—send me email if interested
- **Admittance priority: CSD PhD, ECE PhD, other SCS PhD, CSD Masters, ECE Masters, other Masters**
- **Priority among Masters students based on relevant courses taken (e.g., 15-213, 15-410) and grades obtained**
- **Last Fall, course capped at 24 students (project face-time limit)**
  - I will admit qualified students off waitlist only if enrollment drops below 33
  - If you're not going to take class, please drop so we know the real # of students

Today's Papers

- **"Hints for Computer System Design"**
  Butler Lampson 1983
- **"End-to-End Arguments in System Design"**
  Jerome Saltzer, David Reed, David Clark 1984
- **"The UNIX Time-Sharing System"**
  Dennis Ritchie and Ken Thompson 1974

Further Reading:

- **"Programming Semantics for Multiprogrammed Computations"**
  Jack Dennis and Earl Van Horn 1966

CS is a Fast Moving Field: Why Read/Discuss Old Papers?

"Those who cannot remember the past are condemned to repeat it."

- **George Santayana, The Life of Reason, Volume 1, 1905**

See what breakthrough research ideas look like when first presented

Hints for Computer System Design
Butler Lampson 1983

- **Designing a computer system is different from designing an algorithm**
  - The external interface is less precisely defined, more complex, more subject to change
  - The system has much more internal structure, and hence many internal interfaces
  - The measure of success is much less clear
- **Defining interfaces is the most important part of system design**
  - Conflicting goals: simple, complete, admit a small/fast implementation
  - Each interface is a small programming language

Functionality: Keep it Simple

"Perfection is reached not when there is no longer anything to add, but when there is no longer anything to take away."

"Everything should be made as simple as possible, but no simpler." – Einstein

"We are faced with an insurmountable opportunity."

- **Do one thing well**
  - Don't generalize
  - Get it right
  - Don't hide power
  - Use procedure arguments
  - Leave it to the client

Functionality: Keep it Simple

- **Service must have fairly predictable cost.**
- **Interface must not promise more than the implementer knows how to deliver.**
- **Bad choice for interface can lead to inefficiencies**
  - O(n^2) algorithm for FindNamedField
- **Clients should only pay for power they want**
- **RISC vs. CISC argument**
- **Purpose of abstraction is to hide undesirable properties; desirable ones should not be hidden**
- **Using program arguments limits compiler optimizations**

Functionality: Continuity

- **Keep basic interfaces stable**
- **Keep a place to stand**
  - Backward compatibility package: old interface on new system
  - World-swap debugger

Making Implementations Work

"Perfection must be reached by degrees; she requires the slow hand of time."

- **Plan to throw one away**
- **Keep secrets**
  - ability to improve each part separately
- **Use a good idea again**
  - instead of generalizing it
- **Divide and conquer**

Handling All the Cases

"One crash a week is usually a cheap price to pay for 20% better performance."

- **Make normal case fast**
- **Make worse case ensure progress**
  - E.g., reserve resources to free one item

Speed: Interface

- **Split resources in a fixed way**
  - Dedicated is faster than Shared
- **Use static analysis**
- **Dynamic translation from convenient to fast**

Speed: Implementation

- **Cache answers**
- **Use hints (may be wrong)**
- **When in doubt, use brute force**
- **Compute in background**
- **Batch processing**

Speed: Completeness

"Be wary then; best safety lies in fear."

"The nicest thing about the Alto is that it doesn't run faster at night." – Jim Morris

- **Safety first**
  - Strive to avoid disaster
  - Paging systems avoid thrashing
- **End-to-end**
- **Shed load**
  - Red button user interface idea

Fault-tolerance

- **"The unavoidable price of reliability is simplicity."**
- **End-to-end**
  - For reliability
  - Can add intermediate checks for performance reasons, error codes for visibility reasons
  - Problems: Need cheap test for success. Can mask severe performance defects until operational at scale.
- **Log updates to record the truth about the state of an object**
- **Make actions atomic or restartable (idempotent)**

End-To-End Arguments in System Design

Saltzer, Reed, Clark 1984

- **David Reed designed UDP**
- **David Clark was chief protocol architect in the development of the Internet (1981-1989)**
- **Jerry Saltzer was a team leader for Multics, advisor of Reed and Clark**

Q: What award have David Clark, Dennis Ritchie, Ken Thompson, and Richard Hamming all won?

- IEEE Richard Hamming Award

"Choosing the proper boundaries between functions is perhaps the primary activity of the computer system designer."

The End-to-End Argument

- **The function in question can completely and correctly be implemented ONLY with the knowledge and help of the application standing at the endpoints of the communication system.**
- **Therefore, providing that questioned function as a feature of the communication system itself is not possible.**
- **(Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)**

Applications of End-To-End Argument

- **Careful file transfer**
- **Delivery guarantees**
- **Secure transmission of data**
- **Duplicate message suppression**
- **Guaranteeing FIFO message delivery**
- **Transaction management**

Limitations of End-to-End

- **When are end-to-end checks more/less efficient than low-level checks?**
  - Other apps use low-level & don't need these checks
  - Low-level may have too little information to do checks well
- **How to identify the "ends"?**
  - Real-time phone conversation vs. leaving voice mail
- **Tension with clean layering, clean APIs**

The UNIX Time-Sharing System
Dennis Ritchie & Ken Thompson 1974

Q: What award have Lampson, Hamming, Ritchie, and Thompson all won?
ACM Turing Award

- **Key Features of UNIX (according to authors):**
  - Hierarchical file system incorporating demountable devices
  - Compatible file, device, inter-process I/O
  - Ability to initiate asynchronous processes
  - System command language selectable on a per-user basis
  - Over 100 subsystems including a dozen languages

UNIX

- **Only 50K bytes (Windows 10 is 2.8 GB)**
- **"User-visible locks for the file system are neither necessary nor sufficient"**
- **i-list system table, indexed by a file's i-number; i-node contains attributes of the file – unique feature of UNIX**

"The success of UNIX is largely due to the fact that it was not designed to meet any predefined objectives."

Our goals…building a comfortable relationship with the machine and with exploring ideas and inventions in operating systems

What Influenced the Design

- **Make it easy to write, test, and run programs**
  - Interactive use
  - Interface to the file system is extremely convenient
  - Contents of a program's address space are the property of the program (e.g., no file system control blocks)
- **Severe size constraint on the system & its software**
  - Encouraged economy and elegance of design
- **The system was able to, and did, maintain itself**

Monday's Paper

Implementing Remote Procedure Calls

Andrew Birrell and Bruce Nelson 1984