OS Design Issues

- Hints for Computer System Design - Butler Lampson
- End-to-end arguments in system design
- Brittle metrics in operating systems research

Hints for Computer System Design

- Functionality
  - KISS (Keep it simple)
  - If in doubt, leave it out
  - Exterminate features

  - Get it right
  - Keep basic interfaces stable

  - Don’t hide power with abstractions
  - Continuity: MS Achilles heal

Making implementations work

- Plan to throw one away
- Keep secrets of the implementation
- Divide and conquer
- Reuse a good idea instead of generalizing it
- Fault tolerance - e2e argument

E2E

- End-to-End Arguments in System Design – J. H. Saltzer, D. P. Reed and D. D. Clark – MIT (1980)
  - Jerome Saltzer – Multics
  - David Reed – Visicalc, Lotus 1-2-3, TCP/IP
  - David Clark – Multics, Internet

- Recent followup article:

http://web.mit.edu/Saltzer/www/publications/endtoend/ANe2ecomment.html

- KISS principle (Keep It Simple, Stupid)

Modular vs End-to-End

- Modular:
  - Design, well designed modules, each of which performs a task completely
  - E.g. networking 7 layer OSI model

- End-to-end:
  - Some tasks a better left to higher layers
  - Keep the lower layers simple, perform error checking and other operations at lower layers if it is simple. Higher layers will duplicate this functionality anyway
  - E.g. Internet (only supports IP datagrams)

How does it matter?

- Internet:
  - When ARPANET (->DARPANET->INTERNET) was being designed, there was a tension between adding intelligence in the network. Earlier networks provided “reliable service”.
- Ultimately they chose IP data gram as the transport mechanism. IP provides a best effort service.
- That has enabled the technology to be useful for such a long time. People built TCP (reliable), UDP (unreliable), RTSP (streaming), IP-SEC (secure IP) etc on top of IP.
- Same argument for RISC/CISC (architecture)

Example 1: Careful file transfer

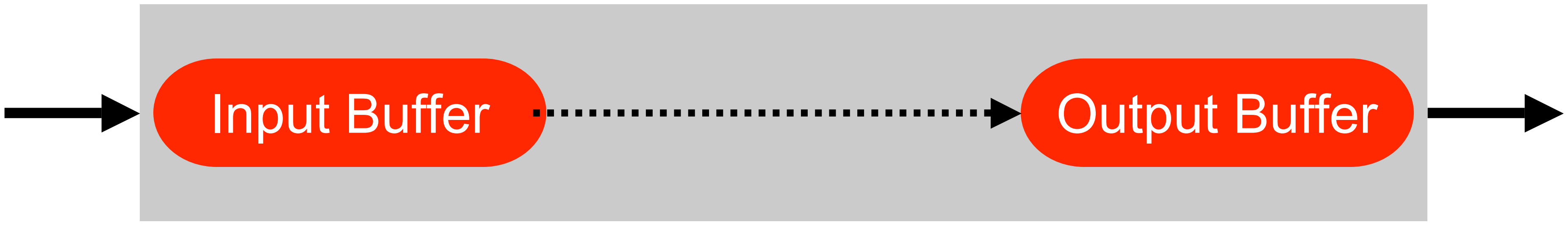
Host A
Host B

- At host A, the file system reads the file and gives to file transfer program
- File transfer program asks network to send file in packets
- Network moves packets from A to B
- Network takes packets from network and gives it to file receive program
- File receive program puts data in the file system

Modular Approach

- Make sure that each of these steps mask failures
  - Read from disk, perform check sums to verify data integrity
  - Make sure the communication network delivers data reliably etc
- End-to-end approach
  - Transfer the file, perform a checksum of the complete file, if it matches then data was correct
  - Optimization: Perform checksums after parts of the file are received

Example 2: Faulty router



- With a modular design, proper check sums verified before putting packets in input buffer. Packets got corrupted inside the router before it copies data to the output buffer. Went undetected

Performance aspects

- Modular implementations do help similar applications
- Frees end user from complexities

- The tradeoff is that the lower levels should provide some functionality, which is easy to implement and leave complicated schemes to higher level mechanisms

Example 3: Delivery guarantees

- Lower level delivery guarantee mechanism can tell when a message was delivered to a computer.
- However, what users really care is if a message was understood by the recipient
  - E.g. Return-Receipt-To: email header will trigger mail servers to send a receipt when email is delivered.
  - Return-View-To: will prompt the email program to send a response as soon as someone opens the email.
  - What you really want is when users read the email and understand it. After message is delivered to the system, the disk can crash, access can be turned off so that lower level acknowledgement is useless. If you user understands the message, they will respond back
- You can force the end system to guarantee delivery to application but simpler at the application level

Example 4: Secure transmission of data

- Data transmission system performs encryption and decryption
  - Trust lower levels with key management
  - Data will be in clear text in the operating system
  - Authenticity of message must still be checked

- Leave end-to-end authorization to higher layers.
- Lower level can perform basic encryption
  - E.g. data between wireless access point and the wireless card is encrypted. Can be broken but it stops casual hackers and is not too hard to implement

Example 5: Duplicate message suppression

- Applications sometime know if a message is a duplicate and drop the second update

Brittle mechanisms

- Develop general traces that capture typical operating system behavior

Threats to end-to-end paradigm

- Quality of Service (QoS) – Routers treat packets differently based on the source, $$$ etc. Internet2 provides heavy support for QoS.

- Firewalls, Caches, NAT
  - Prevents end-to-end interaction