

CS123: Abstractions in Computer Science and Networking

George Varghese

April 1, 2010

Abstractions

Abstractions are idealizations we invent to deal with the perversity and complexity of reality. We use them all the time to deal with the complexity of the world around us. Examples include

- *Models in Physics:* We abstract planets as point masses interacting via gravitation to predict orbits ignoring different compositions, shapes, other forces etc.
- *Music:* We abstract the complexity of music (tone, timber) using notes
- *Math:* We abstract matrices and numbers as fields ignoring their differences.
- *Computer Science:* We would like to hide the messy details of computer and networks from programmers to make them more *productive* by hiding implementation details behind a programming interface.

Operating Systems: Abstracting a Computer

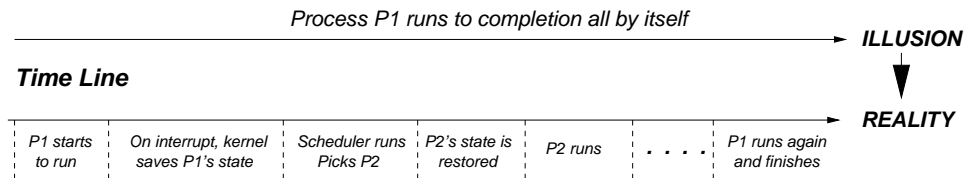
Create clean abstractions so that application programmers on say Windows OS can write large programs quickly and correctly. Examples include

- *Processes: Abstraction of Uninterrupted Computation:* Dealing with idiosyncrasies of interrupts would be intolerable.
- *Virtual Memory: Abstraction of Infinite Memory:* Dealing with the details of moving data from disk to main memory and vice versa would be a pain.
- *Simple I/O: Abstracting devices* Dealing with the idiosyncrasies of various devices (USBs, cameras, microphones) would be hard without abstracting them as memories that can be read and written.

Avoid dealing with dubious details of devices, simpler illusion of reading and writing devices

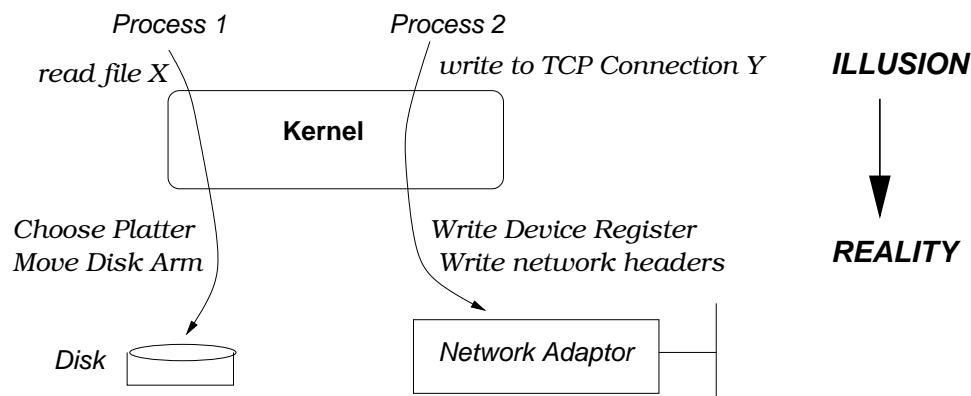
from memory.

OS Example: Uninterrupted Computation via Processes

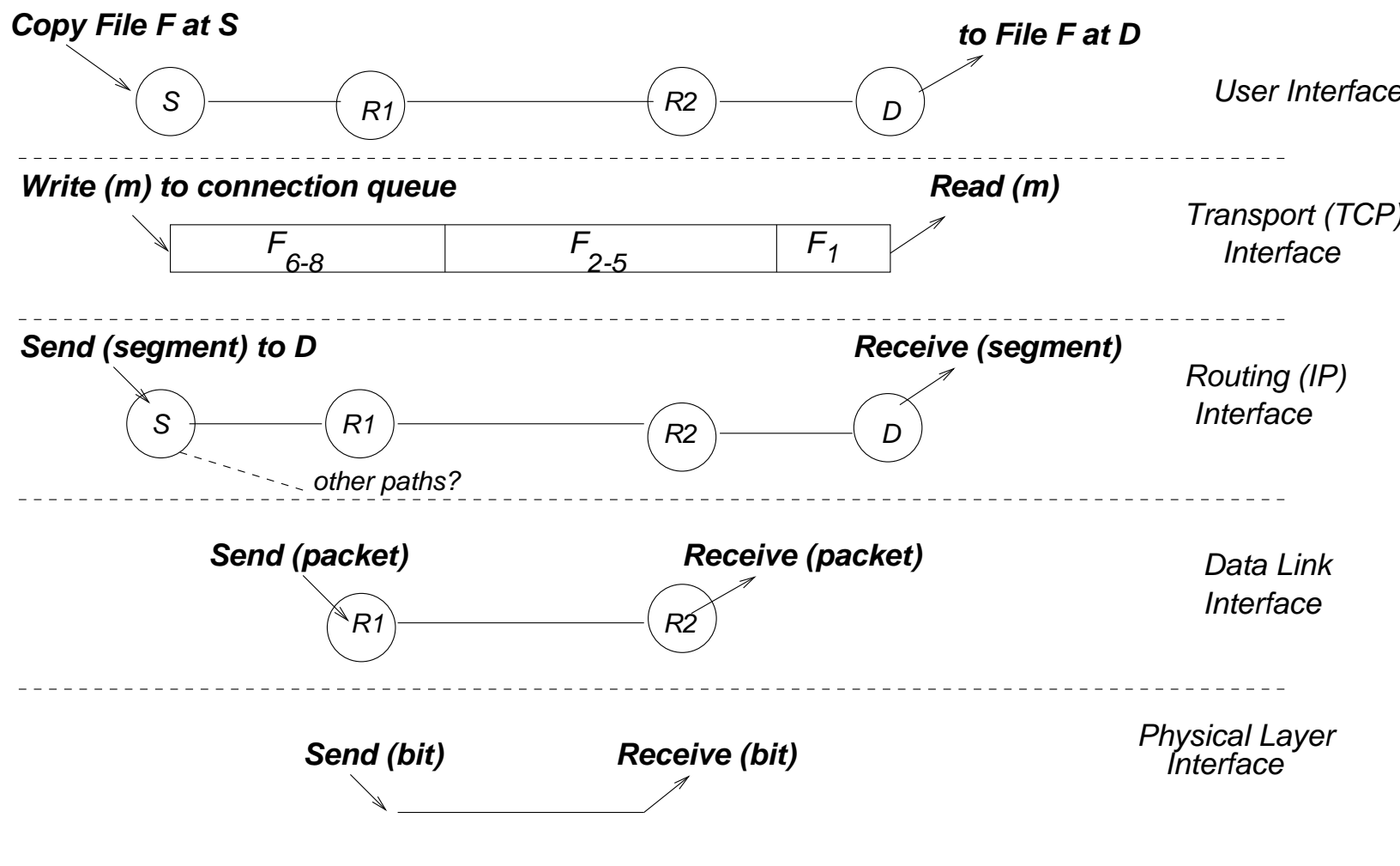


- 3 underlying mechanisms: Context switching (save and restore), scheduling, protection
- 3 flavors of “process”: interrupt handlers, threads, and user processes - order of increasing generality and cost.

Example 2: Simple I/O using System Calls



- If abstraction was only concern, I/O calls could be libraries. For security, I/O calls handled by device drivers in a core of essential services (kernel) that applications cannot be trusted to do.
- System calls, trap to kernel protection levels. More expensive function call because of hardware privilege escalation, and sanitizing checks. 2.31 usec on Pentium II 100 using Linux. Affects networking.



Abstracting Networks: From Files to Voltage Levels

- File Transfer implemented by two FTP processes on each machine. Shared queue is simplest asynchronous interface, which is what TCP provides.
- TCP implements shared queue abstraction by sending numbered segments, retransmitting if acks are not received. Requires being able to send segments to arbitrary destinations, which is what IP provides.
- IP computes routes (routing) and then forwards packet hop-by-hop. At each hop, IP requires sending a frame to directly connected neighbor, which is what Data Link provides.
- Data Link uses physical layer to send each bit of a frame; then puts together bits at receiver to form a frame and does error checks. Physical layer sends bits by transforming 0s and 1s into physical

energy that can travel distance.

Alternate Abstractions

- TCP and IP are not the only abstractions!
- IP is a post-office abstraction. Another abstraction is analagous to a telephone call; So-called *virtual circuit* abstraction. Set up a call and then send data.
- Many other higher level abstractions besides shared queue (although they can be built on TCP). All extensions of standard OS abstractions for processes to communicate within a single machine, but now across network:
 - *Shared queue*: — i TCP
 - *Shared memory*: — i Distributed Shared Memory
 - *Procedure Calls*: — i ?
 - You invent one

Plan for Rest of Class

- We will study the major abstractions starting bottom up with Physical Layer (bit pipe) to Data Link Layer (frame pipe) and Routing Layer (packets across the Internet).
- In each case our focus will be on the *mechanisms* (analogous to the way the OS scheduler uses scheduling to provide processes) underlying the abstractions.
- We will only briefly study the shared queue abstraction provided by TCP (2 lectures at end) as the follow-up class will cover it in detail as well as applications above.
- Before we dive into the physical layer, lets get a quick overview of some of the messy details underlying each abstraction by looking at what is involved in a Web transfer or Email.