The End-to-End Argument, Physical and Data Link Layers

Adolfo Rodriguez

CPS 214

January 15, 2004

The End-to-End Argument

Internet History

- Goal: effective multiplexed use of existing networks
  - Minimal support from underlying networks

    e.g., no support for multicast, real-time, fast failover, congestion control, etc.
  - Packet switching (fine-grained resource sharing)

    AT&T said it could not be built
  - Routers connecting networks
  - Sense of collaborative community

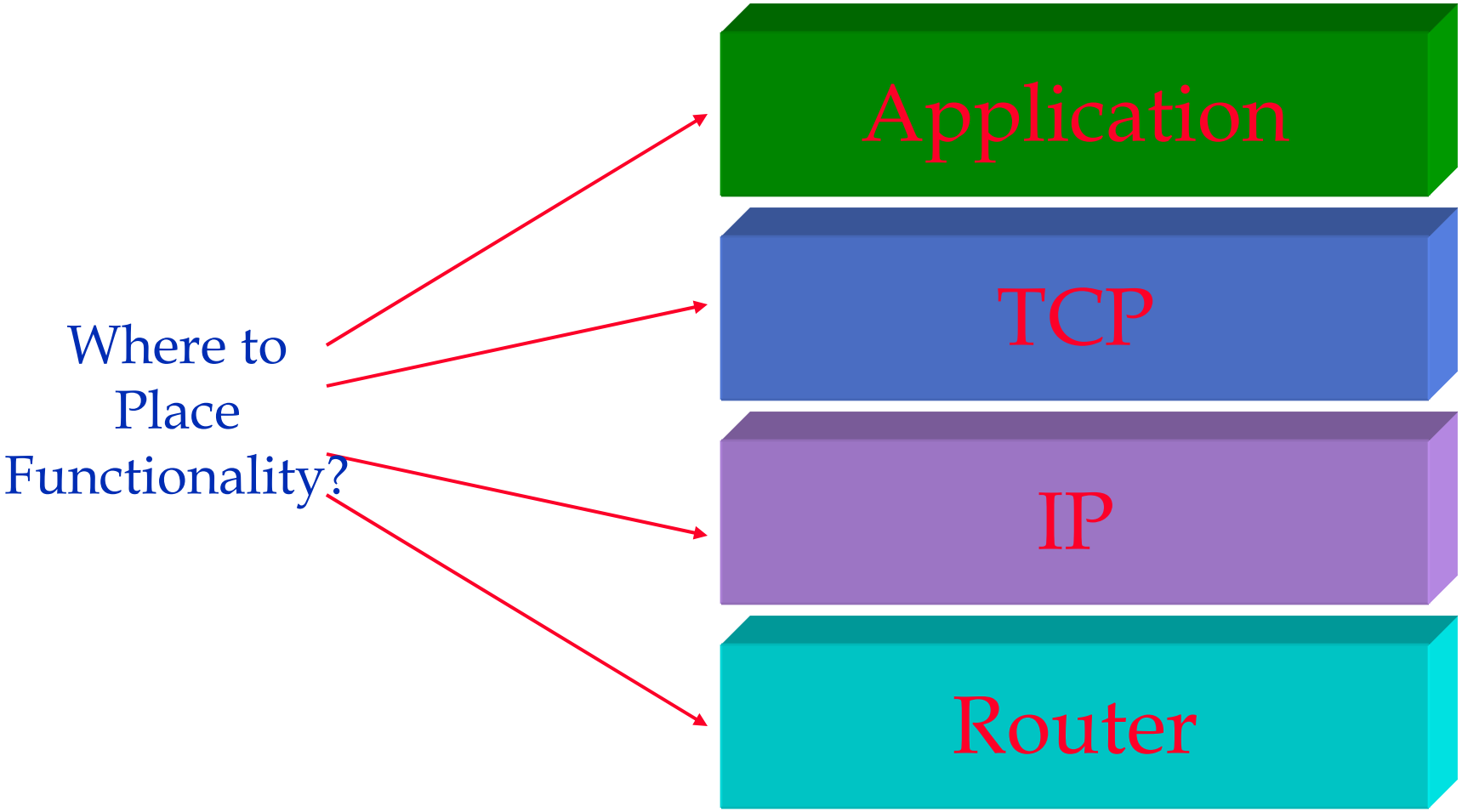
Internet History: Other Goals

- Survive hardware failure
- Support multiple types of applications
- Run on wide variety of networks
- Distributed management of resources
- Cost-effective
- Low cost host attachment
- Accounting

Survivability

- Internet approach
  - Cheap, commodity components
  - Stateless routers + self-healing
  - Keep routing simple (non-adaptive)
  - End to end recovery
- Telephone approach
  - Ultra reliable switches
  - Make the network very smart

End-To-End Argument



End-to-End Argument

- Functionality should be implemented at a lower layer if and only if it can be correctly and completely implemented there
  - Should not be implemented at lower level if redundant with higher level
  - Performance optimizations are not a violation
- Early example
  - ARPANet provided reliable link transfers between switches
  - Packets could still get corrupted on host-to-switch link, or inside switches
  - Want to know if host *acted* on the request not whether it *received* it

Example: Reliable File Transfer

- From disk on file (web) server over network to client
  - Disk can introduce bit errors
  - Host I/O buses can introduce bit errors
  - Packets can get garbled, dropped, misordered at any node
- Solution: integrity check on file, not per packet or per hop

Hop by Hop as Performance Optimization

- Does not violate end to end argument to provide reliability at link layer, if not required for correct operation
- For file transfer application, consider varying conditions:
  - Prob(corrupted/lost packet per link) = p
  - Prob(packet lost end to end), avg. 15 hops across Internet

    p = 0.0001% => Prob(loss) = 0.0015%

    p = 1% => Prob(loss) = 14%
- Chance of file corruption grows with size of file
  - Potentially retransmit entire file for one lost packet?

The Need for Application-Specific Semantics

- Example: move reliability into the network communication protocol (such as TCP)
  - Certain computational and bandwidth overheads to implementing reliable, in-order delivery in the network
  - Not all applications want to pay this overhead (use UDP)
- Real-time voice/audio
  - Better to drop a packet, rather than hold up later packets
  - *On-time* delivery more important than *reliability*
- Applications should be able to pick and choose the semantics they require from underlying system

  ➔*Active Networks?*

Implication of End to End Principle

- Internet assumption: minimal support from underlying network
  - Ensure Internet can run on anything (IP on top of anything)
- Implications
  - Almost everything done at end hosts
  - Requires intelligent end hosts
  - Overlay networks
- Telephone network has stupid endpoints
  - What happens when light switch runs TCP?
  - Should your light switch run TCP?

Examples

- What should be done at the end hosts, and what by the network?
  - Addressing/routing?
  - Reliable delivery?
  - Sequenced delivery?
  - Congestion control/resource allocation?
  - Real-time guarantees?
  - Security?
  - Multicast?

What's Changed?

1980's Internet	2000's Internet
Low bandwidth * delay	High bandwidth * delay
Low drop rates, < 1%	High drop rates, > 5%
Few, long-lived flows	Many short-lived flows
Every host a good citizen	TCP "accelerators"
Symmetric routes & universal reachability	Asymmetric routes & private peering
Hosts powerful & routers overwhelmed	Hosts = toasters & routers intelligent?
Limited understanding of packet switching	ATM and MPP network design experience