Rethinking the End-to-End Principle

EE4204: Computer Networks
Mehul Motani
motani@nus.edu.sg

Outline

- Background
  - The end-to-end principle underlies the Internet architecture.
  - Along with packet switching, it is one of the most key choices in Internet design
  - A faithful adherence to the e2e principle is viewed as the chief reason for the Internet's successful operation despite an enormous increase in the load and the introduction of unanticipated new applications
- Our goal
  - Ask whether we have been overly faithful to the e2e principle at the expense of performance, and whether it has stifled innovation
- The plan of the talk
  - Revisit the e2e principle (the original statement and later interpretations)
  - Discuss instances where it is better to "violate" it
  - Explore a more systematic approach to the e2e principle's relevance

Note: Thanks to Balaji Prabhakar at Stanford and the Clean Slate Design for the Internet Project

The e2e argument

- Original statement in Saltzer, Reed and Clark (1981)
  - http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf
  - Motivated by a need to clarify the boundaries between functions
  - Consider a system which includes a communication subsystem
    - Where should a function be implemented?
    - In the client? The communication subsystem? Jointly?
- The e2e argument
  - "The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points 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.)"

The Saltzer, Reed and Clark paper

- The e2e argument is stated in the 2nd paragraph
  - As a guiding principle for distributed system design
- The rest of the paper
  - Tests the e2e argument for implementing functions like reliable file transfers, providing delivery guarantees, secure transmission, FIFO delivery, detecting host crashes, delivery receipts
  - It concludes that the e2e argument suggests the better way of performing these functions; i.e. at the end-systems, not in the network
- Another terrific paper on Internet architecture is
  - David Clark (1998): "The Design Philosophy of the DARPA Internet Protocols."
  - It links the Internet architecture to design goals (interconnectivity, survivability, diversity, distributed resource management, cost, easy host addition, accountability)

Why revisit the e2e argument?

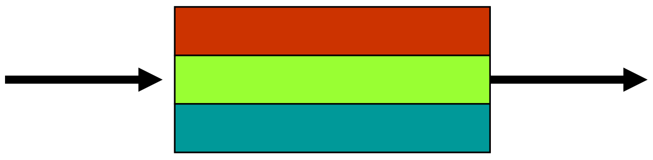
- Because it has spawned many (inviolable) corollaries
  - Don't do *anything* in the network, keep it as dumb as possible
  - Don't recognize flows, deal with packets; flow-recognition is expensive
  - Indirectly, other aspects of Internet architecture have been linked up with e2e, and therefore viewed as being equally immutable: e.g. packet switching, layering
- But times have changed: flow-recognition is key
  - Bandwidth sharing (fair queueing)
    - the benefits were obvious: guaranteed bandwidth to flows
    - shot down because of cost: requires per-flow state, hence expensive
    - because of customer demand, there is no option but to provide this service
  - Label (circuit) switching: MPLS widely used for providing flow isolation and for QoS; the network needs to maintain flow state to do this
  - Explicit Loss Notification. Transport media are diverse: copper, fiber, wireless, satellite; transport is bad if it is purely e2e, ELN improves performance significantly
  - WWW and flow size distribution. Most packets belong to a few large flows (elephants), most flows are small (mice); therefore, controlling the elephants and letting the mice whiz by reduces delays tremendously
  - Security: In order to provide security, the network needs to be conscious of flows, because flows are malicious, not aggregates

In summary

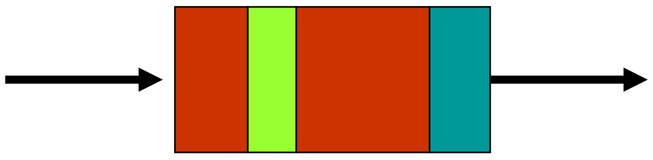
- It all boils down to this
  - The Internet is a mechanism for transporting flows
  - However, its switches and routers only recognize packets, not flows
  - This state of affairs suits everyone except the user, who has begun to demand (e.g. for bandwidth partitioning and security) some form of flow-recognition
  - The e2e argument has been used by Internet architects and (less explicitly) by router manufacturers to avoid the expense of flow-recognition
- The e2e argument needs to be revisited because
  - It is *relatively cheap* for routers to recognize flows, and therefore to obtain the benefit of flow-processing
- Let' s look at two examples.

Example 1: Bandwidth Allocation

- While it is true that flow-state maintenance is expensive, it is relatively cheap to maintain approximate flow-state using randomized algorithms.
- Example 1: Bandwidth partitioning. Split a link's capacity among the flows that traverse it.



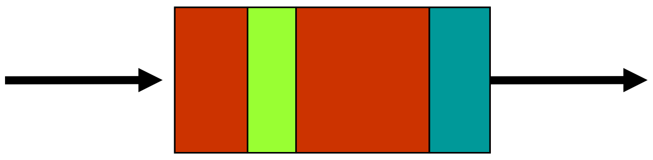
Solution 1
Fair queueing
Expensive



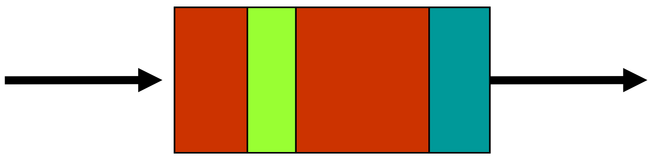
Solution 2
FIFO buffers, TCP at end- hosts
Doesn't work

Bandwidth partitioning

- Randomized algorithm, first cut: When FIFO buffer is full, drop a randomly chosen packet. More likely to toss out red packets. Doesn't work.
- Second try: When buffer is full, choose two packets, *drop both* if they are of the same color. This algorithm (called CHOKe) works.



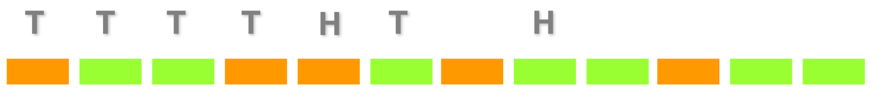
CHOKe: Automatically identifies and penalizes a small number of dominant flows.
(Prabhakar, Pan and Psounis)



AFD: Does more work than CHOKe, but controls almost all large (elephant) flows.
Being implemented in 2 Cisco platforms.
(Prabhakar, Pan, Bonomi, Breslau, Shenker)

Example 2: Elephant traps

- The SIFT algorithm for finding large flows at a switch or a router.
- Flip a coin with bias *p* (= 0.1, say) for heads on each arriving packet, independently from packet to packet
- A flow is "sampled" if one its packets has a head on it

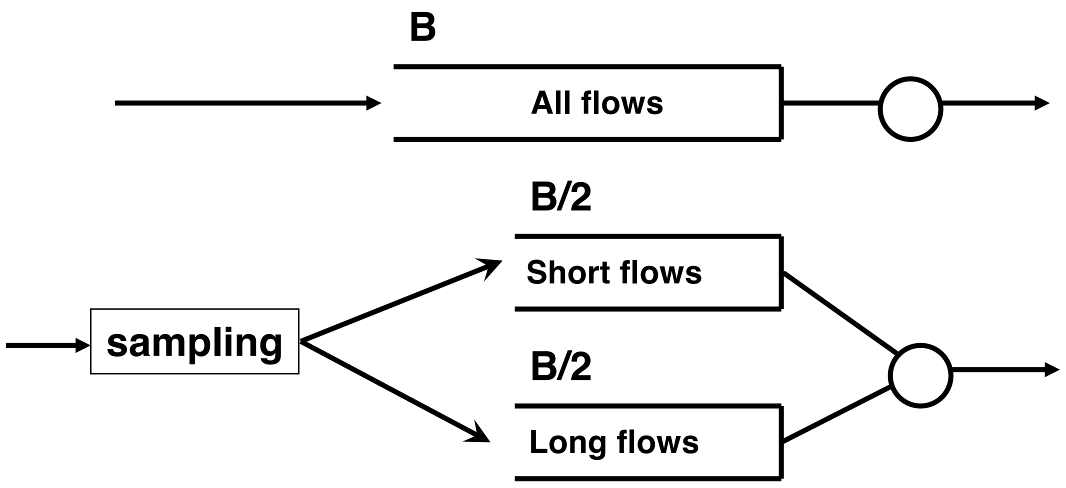


- A flow of size X has roughly 0.1X chance of being sampled
  - flows with fewer than 5 packets are sampled with prob $\cong$ 0.5
  - flows with more than 10 packets are sampled with prob $\cong$ 1
- Most short flows will not be sampled, most long flows will be

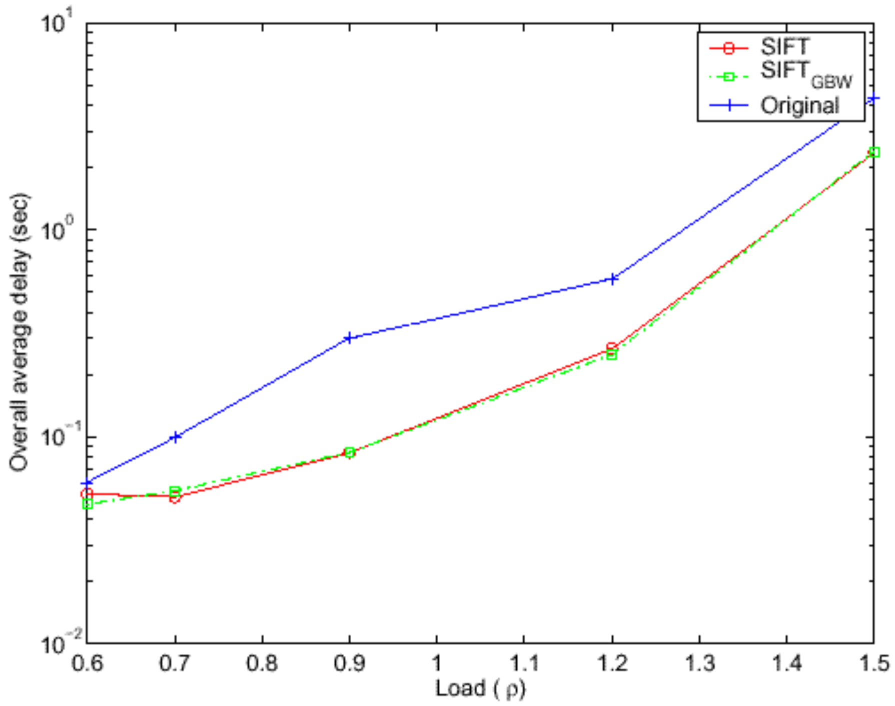
- How can we use this sampler?

SIFT at an (edge) router

- Sample incoming packets
- Place the packets of sampled flows in the low priority buffer

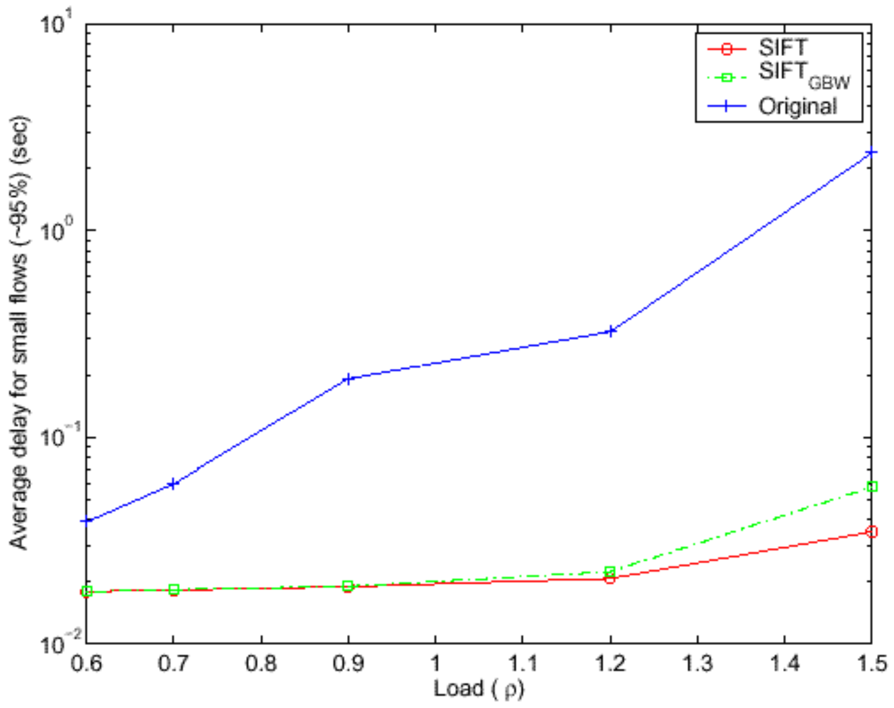


Overall Average Delays



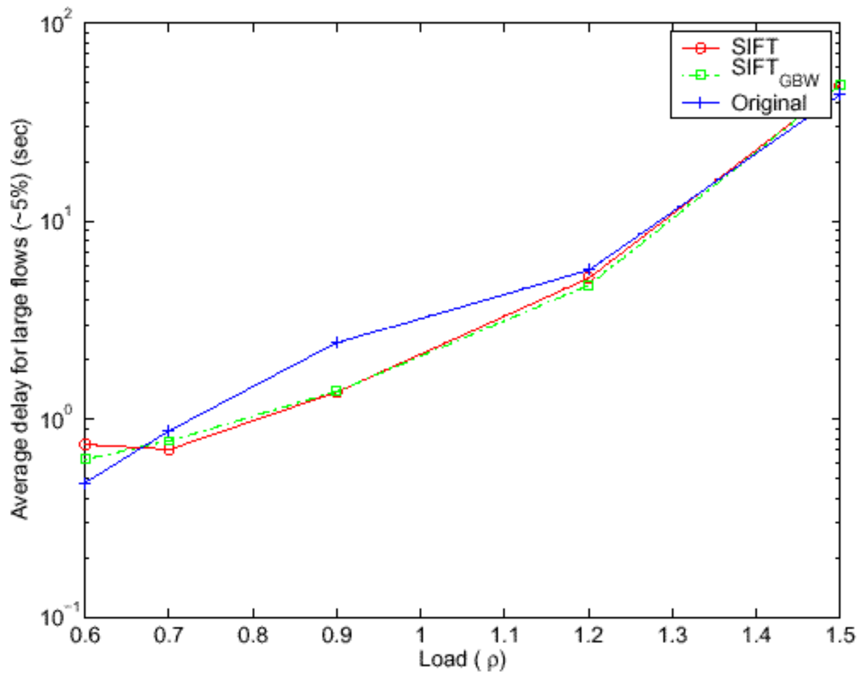
SIFT: a simple algorithm for trucking elephant flows and taking advantage of power laws Konstantinos Psounis, Arpita Ghosh, Balaji Prabhakar, and Gang Wang, 43rd Allerton Conference on Communication, Control, and Computing, September 2005.

Average Delay for Short Flows



From the SIFT paper.

Average Delay for Long Flows



From the SIFT paper.

Conclusions

- An argument to reconsider the e2e principle
  - First, because it is the most crucial part of the current Internet architecture
  - Second, because violating it is both easy and, some times, quite helpful
- Do you see an analogy to Cross Layer Design? (refer to week 1 of lecture)
- I believe that a slavish adherence to layering and e2e has stifled innovation at low levels in the network
- Does the e2e principle have a role in the next generation internet?
- What about cross layer design?