Lecture 3 Overview

- Paper reviews
- Packet Forwarding
- IP Addressing
  - Subnetting/CIDR
- Data-plane processing
  - NAT
  - Middle boxes
Paper review tips

● Demonstrate you *understood* the paper
  ◆ Don’t just give short synopses
  ◆ Analogies/examples are good ways to demonstrate thought

● Understand the context of the paper
  ◆ Where/when was it published? By whom?

● Think outside of the box
  ◆ We selected “good” papers
  ◆ Shortcomings are likely to lie outside of their stated scope

● We’re looking for implications they didn’t state
  ◆ What impact did/could this paper have beyond the goals listed specifically in the paper
CONTRIBUTIONS
The primary contribution of the paper is to fully articulate the end to end principle and to provide a number of examples and arguments in support of this design principle. Though the principle itself was not an innovative new idea at the time the paper was written, it’s still useful to have a single paper explaining and providing strong examples of the reasoning and potential benefits. Essentially the authors argue that because end point applications must implement certain functions anyways in order to ensure their requirements are met, there is not necessarily a great need or benefit to implement these functions completely at the low level (i.e., in the network system). While sometimes there may be a performance benefit to implementing such functions in the network, the paper argues that there must be a careful decision made about at what point it will cost more to do at the low level compared to leaving it to the applications on the hosts.

SHORTCOMINGS
The first issue I had with this paper is that it seems to downplay the potential benefits of implementation in the network. While it’s true this will lead to redundancy, I think the fact that today we use TCP rather than UDP for most network communication, where reliable transmission is handled by the transport layer rather than just the applications, suggests that there’s a lot more value than the author’s give credit for. This may be because of changes in the amount and types of traffic on networks, or other technological changes since the paper was written, but I think it’s still something they should have given more consideration. The paper also fails to account for other changes, such as the introduction of massively distributed applications that change the structure of the system away from a simple end to end design, or simple lightweight mobile end points where we may not want (or have the power) to have all the functions provided by applications on the host. We can’t expect them to have thought of everything that would happen over 35 years of technological advancement, of course, but there doesn’t seem to be a lot of consideration given to any possible changes that may have come, and different types of systems that may have different design goals, where the end to end principle may not be as beneficial.

IMPLICATIONS
TCP aside, the internet was mostly built using the end to end principle, so clearly this paper (or at least the ideas contained within) were extremely influential in shaping the last 35 years of network design. We still do not build a lot of complexity into intermediary nodes (routers, switches, etc) in the network, and most of our functionality is provided by software on endpoints, as the paper argues for. However, things have changed, in terms of the type and amount of traffic, the design of applications, and the types of users and their requirements for the systems they use. There are a lot more reasons today, in a highly connected, high bandwidth, mobile, everything-as-a-service world, why we might want to build more functions into the network rather than only on the end points. Higher demand for performance, security, usability and simplicity of hosts, and being able to have some control over the network all could move us away from the end to end principle as we go forward. However, as a general principle of network (or just system) design, there is still a lot of merit to the ideas in this paper, given that they can help to simplify systems, reduce costs, and make them more generally usable for a variety of applications.
CONTRIBUTIONS
1. This paper introduced a design principle in terms of communication function "placement". That is to layer the network and move the function upward in order to be closer to the application, i.e. the end point that implement the function.

2. The paper analyzed why the performance doesn't justify implementing the function at lower-levels and argues that higher-level function get benefits because the end points have a better knowledge of the provided service. The authors also argues that such a system leads to low-cost, high efficiency, minimal and flexible design and lower-level can focus on performance enhancement of transmission purpose.

3. It is not an absolute rule, but provides guidance to application and protocol design analysis. The paper brought security into design consideration also, and it really points out the fact that addresses practical problems and have experimental results/examples support.

SHORTCOMINGS
1. The paper mentioned that the end points must be identified. But this brings two questions: 1. what is a trusted end-point. 2. How difficult it is to identify such an end-points. The complexity of identifying end-points should be studied when building the system. There is no rule-of-thumb in identifying the end-points. The internet today are not simple as two end-points communications. Perhaps we don’t even actually trust each other.

2. Pushing everything to the application layer may lead to problems. There would always be tradeoff between putting the burden on the edge of network, and providing it on lower-level. So the question would be, on what extent should we place function on a certain layer. The paper doesn’t solve this problem, there is no definite way of telling how deep we should place the mechanisms.

3. There are possibilities that the network load could increase due to the large retransmission of incorrect packets. This will takes up a large portion of capacity of the communication link. This may not be desired at certain application occasions, like energy-efficient environments.

4. For the security example, only implement the security on the end-point is dangerous because the attacker may attack lower layer. Security should be a redundantly implemented at multiple layers.

5. End-to-end approach may cause redundancy if multiple application at the same level implement some similar functions. That module can be potentially reused if implemented at lower level.

IMPLICATIONS
The paper is published during the time when TCP/IP is being developed. The "end-to-end" idea and the "layering" in networks is the foundation of TCP/IP networks stacks (and possibly ISO/OSI). In most cases it introduced great...
Motivation

* Goal: efficiently deliver packets between arbitrary hosts in the network
* Key concerns: scalability, performance, robustness
* Key techniques:
  * Hierarchical design
    » Learn enough to do part of the job, hand task off to someone else
    » e.g., one technique for inter-domain routing, another for intra-domain
  * Soft state
    » Learn all the information you need to perform routing
    » State info not for correctness, just performance optimizations
Forwarding Options

- Source routing (Myrinet)
  - Packet carries path
- Table of global addresses (IP)
  - Stateless routers
- Table of virtual circuits (ATM)
  - Small headers, small tables

- How do hosts/switches learn optimal network routes?
  - Given packet header, how to determine forwarding port
  - Topic for Thursday…
## Comparison

<table>
<thead>
<tr>
<th>Source routing</th>
<th>Global addresses</th>
<th>Virtual circuits</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header size</strong></td>
<td>worst</td>
<td>OK ~ large addr</td>
</tr>
<tr>
<td><strong>Router table size</strong></td>
<td>none</td>
<td># of hosts (prefixes)</td>
</tr>
<tr>
<td><strong>Forward overhead</strong></td>
<td>best</td>
<td>Prefix matching</td>
</tr>
<tr>
<td><strong>Setup overhead</strong></td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td><strong>Error recovery</strong></td>
<td>Tell all hosts</td>
<td>Tell all routers</td>
</tr>
</tbody>
</table>
Packet Forwarding

- Control plane computes a forwarding table
  - Maps destination address(es) to an output link
- Handling an incoming packet
  - Match: destination address
  - Action: direct the packet to the chosen output link
- Switching fabric
  - Directs packet from input link to output link
Switch: Match on MAC

- MAC addresses are location independent
  - Assigned by the vendor of the interface card
  - Cannot be aggregated across hosts in the LAN

Implemented using a hash table or a content addressable memory.
IP Routers: Match on IP Prefix

- IP addresses grouped into common subnets
  - Allocated by ICANN, regional registries, ISPs, and within individual organizations
  - Variable-length prefix identified by a mask length

- Prefixes may be nested.
- Routers identify the *longest matching* prefix.
Address Resolution Protocol

- IP forwarding tables: one entry per *network* not *host*
  - Thus, routes designed to get packets to proper network
  - Network needs to take over from there to get to proper host

- Address resolution protocol (ARP) translates IP addresses to link-level addresses (e.g., Ethernet addr)
  - Broadcast request over network for IP→link-level mapping
  - Maintain local cache (with timeout)
ARP example

Broadcast: Anyone know the Ethernet address for 152.3.140.5?

Reply: Yes, I’m at 08-00-2b-18-bc-65

Packet arrives for host on the same physical network

(152.3.145.240) (152.3.140.5)
Internet: Hierarchical Routing

- Internet composed of many autonomous systems (AS’s)
  - Correspond to administrative domains
- Each AS can choose its own routing algorithm
  - Routing Information Protocol (RIP) used originally
    - Part of BSD distribution, distance vector
  - Open Shortest Path First (OSPF) currently most popular
    - Link state protocol w/authentication, basic load balancing
- Border Gateway Protocol (BGP) for routing between AS’s
  - Default: shortest number of AS’s in path
  - Sys admins can express policy control
    - Use AS x in preference to AS y
IP: The Internet Protocol

- Service mode: *best effort*
  - No guarantees about reliable, in-order, or error-free delivery
  - Enables IP to “run over anything”

<table>
<thead>
<tr>
<th>Version</th>
<th>HLen</th>
<th>TOS</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ident</td>
<td>Flags</td>
<td>Offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td>SourceAddr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DestinationAddr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (variable)</td>
<td>Pad (variable)</td>
<td>Data…</td>
<td></td>
</tr>
</tbody>
</table>
Fragmentation and Reassembly

- Problem: networks have different maximum transmission units (MTUs)
  - Ethernet: 1500 bytes, FDDI: 4500 bytes, etc.
  - Communicating hosts may be on networks w/similar MTUs
    But smaller MTU somewhere in the middle of the network
- To maintain uniform host-to-host communication, IP must fragment and then reassemble packets on end-to-end basis
- Hosts pick MTU of local network
- Fragmentation takes place at routers that forward packets along links with smaller MTU
  - Input on 1500-byte MTU link, output on 500-byte MTU link
IP Address Allocation

- Originally (classfull addrs), 4 address classes
  - “A”: 0 | 7 bit network | 24 bit host (1M each)
  - “B”: 10 | 14 bit network | 16 bit host (64K)
  - “C”: 110 | 21 bit network | 8 bit host (255)
  - “D”: 1110 | 28 bit multicast group #

- Assign net # centrally, host # locally
  - E.g., assign UCSD a class B address
IP Address Issues

- We can run out
  - 4B IP addresses

- We’ll run out faster if sparsely allocated
  - Rigid structure causes internal fragmenting
  - E.g., assign a class C address to site with 2 computers
    Waste 99% of assigned address space

- Need address aggregation to keep tables small
  - 2 million class C networks
  - Entry per network in IP forwarding tables
    Scalability?
Efficient IP Address Allocation

- Subnets
  - Split net addresses between multiple sites

- Supernets
  - Assign adjacent net addresses to same organization
  - Classless routing (CIDR)
    Combine routing table entries whenever all nodes with same prefix share same hop

- Hardware support for fast prefix lookup
CIDR

• Classless Interdomain Routing (CIDR)
  ◆ Balances entries in forwarding tables with need to efficiently distribute IP address space

• Example: site that requires 16 class-C IP addresses
  ◆ Use 16 contiguous class C addr, e.g., 192.4.16-192.4.31
  ◆ Top 20 bits are identical
  ◆ Have created something between a class B and class C addr “Classless”

• Need routing protocols to recognize CIDR
  ◆ BGP-4 sends updates in form <length,value> (e.g., where length is 20 for above example).
Network Address Translation

- Allows multiple machines to be assigned same IPV4 addr
- NAT separates internal from external hosts
  - Hosts only need internally unique address
- NAT translates each packet
  - Multiplex small set of externally unique addresses among active connections
  - Internal IP -> dynamically allocated ext. IP
- What if NAT crashes?
- Interaction with security?
Network Address Translation

NAT

10.0.0.1

10.0.0.2

138.76.29.7

inside

outside

CSE 222A – Lecture 3: Packet Forwarding
Buffering

- **Drop-tail FIFO queue**
  - Packets served in the order they arrive
  - ... and dropped if queue is full

- **Random Early Detection (RED)**
  - When the buffer is nearly full
  - ... drop or mark some packets to signal congestion

- **Multiple classes of traffic**
  - Separate FIFO queue for each flow or traffic class
  - ... with a link scheduler to arbitrate between them
Traffic Shaping

- Force traffic to conform with a profile
  - To avoid congesting downstream resources
  - To enforce a contract with the customer

- Leaky-bucket shaping
  - Can send at rate $r$ and intermittently burst
  - Parameters: token rate $r$ and bucket depth $d$

A leaky-bucket shaper for each flow or traffic class
Generalized Data Plane

- Streaming algorithms that act on packets
  - Matching on some bits, taking a simple action
  - ... at behest of control and management plane

- Wide range of functionality
  - Forwarding
  - Access control
  - Mapping header fields
  - Traffic monitoring
  - Buffering and marking
  - Shaping and scheduling
Middle Boxes

- **Router**
  - Forward on destination IP address
  - Access control on the “five tuple”
  - Link scheduling and marking
  - Monitoring traffic
  - Deep packet inspection

- **Switch**
  - Forward on destination MAC address

- **Firewall**
  - Access control on “five tuple” (and more)

- **NAT**
  - Mapping addresses and port numbers

- **Shaper**
  - Classify packets
  - Shape or schedule

- **Packet sniffer**
  - Monitoring traffic
Future of IP

- Is IP really an indirection layer?
  - Mobility
  - Overlay networks
  - Peer to peer networks
  - Application-layer routing
  - NAT

- Perhaps an IP address becomes a convenient intermediate handle for performing the next lookup?
For Next Class…

- Read P&D Chapter 4
- Read and review Paxson ’97
- Keep thinking about term project ideas/groups
  - Groups and ideas due next week