Last Monday on
Communication Networks
IPv6

next generation of Internet addressing

programmable networks

next generation of network devices
IPv6

next generation of Internet addressing

programmable networks
Network Address Translation (NAT)

Sharing a single (public) address between hosts
Port numbers (transport layer) are used to distinguish

One of the main reasons why we can still use IPv4
Saved us from address depletion

Violates the general end-to-end principle of the Internet
A NAT box adds a layer of indirection
The Internet with NAT

Hosts behind NAT get a private address

<table>
<thead>
<tr>
<th>IP:port</th>
<th>src</th>
<th>dst</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.3.4:3001</td>
<td>5.6.7.8:80</td>
<td>9.10.11.12:5000</td>
</tr>
<tr>
<td>5.6.7.8:80</td>
<td>192.168.3.4:3001</td>
<td>9.10.11.12:5000</td>
</tr>
</tbody>
</table>
The Internet with NAT

The port numbers are used to multiplex single addresses
Let's talk about IPv6
IPv6 addresses are encoded in 128 bits

Notation
8 groups of 16 bits each separated by colons (:
Each group is written as four hexadecimal digits

Simplification
Leading zeros in any group are removed
One section of zeros is replaced by a double colon (::)
Normally the longest section

Examples
1080:0:0:0:8:800:200C:417A  →  1080::8:800:200C:417A
FF01:0:0:0:0:0:0:101       →  FF01::101
0:0:0:0:0:0:0:1            →  ::1
There are three types of IPv6 addresses: unicast, anycast, and multicast

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unicast</td>
<td>Identifies a single interface</td>
<td>Packets are delivered to this specific interface</td>
</tr>
<tr>
<td>Anycast</td>
<td>Identifies a set of interfaces</td>
<td>Packets are delivered to the &quot;nearest&quot; interface</td>
</tr>
<tr>
<td>Multicast</td>
<td>Identifies a set of interfaces</td>
<td>Packets are delivered to all interfaces</td>
</tr>
</tbody>
</table>
The IPv6 packet header format

- **Version**: 6
- **Traffic Class**: Same as DSCP
- **Flow Label**: Unclear utilisation
- **Payload Length**: Size of packet content in bytes
- **Next Header**: Indicates type of next header. E.g. TCP/UDP or extension header
- **Hop Limit**: Same as TTL in IPv4
- **Source IPv6 address**
- **Destination IPv6 address**
How can a node obtain its IPv6 address(es)?

Manual configuration
As in the project, e.g. with ifconfig

From a server by using DHCPv6
Similar to the IPv4 version

Automatically
Using its link-local address and neighbor discovery
IPv6 autoconfiguration to find link-local address

Consider an end-system which has just started, it needs an IPv6 address to send ICMPv6 messages

Ethernet (MAC): 0800:200C:417A
Link-local: FE80::M₆₄(800:200C:417A)
M₆₄: 64-bit representation of the MAC address

Neighbor solicitation for FE80::M₆₄(800:200C:417A)
If no answer, the created link-local address is valid
IPv6 autoconfiguration
to obtain the IPv6 prefix of subnet

Routers periodically advertise the prefix
Sent to all end-systems: FF02::1

The advertisements can contain:
IPv6 prefix and length
Network MTU to use
Maximum hop limit to use
Lifetime of the default router
How long generated addresses are preferred
IPv6 autoconfiguration
to build **global unicast** address

Ethernet (MAC): 0800:200C:417A

Prefix: 2001:6a8:3080:1::/64

**Global unicast:**
2001:6a8:3080:1:M64(800:200C:417A)

contains MAC address of host
Today on

Communication Networks
We'll finish up the introductory material on programmable networks before a recap of the lecture.
Communication Networks

So what?!
Knowledge
Understand **how** the Internet works and **why**

from your network plug...

...to Google's data-center
List any technologies, principles, applications… used after typing in:

> www.google.ch

and pressing enter in your browser
Insight

Key concepts and problems in Networking

Naming    Layering    Routing    Reliability    Sharing
Skill

Build, operate and configure networks

Trinity using a port scanner (nmap) in Matrix Reloaded™
The Internet is organized as layers, providing a set of services

<table>
<thead>
<tr>
<th>layer</th>
<th>service provided</th>
</tr>
</thead>
<tbody>
<tr>
<td>L5</td>
<td>Application</td>
</tr>
<tr>
<td>L4</td>
<td>Transport</td>
</tr>
<tr>
<td>L3</td>
<td>Network</td>
</tr>
<tr>
<td>L2</td>
<td>Link</td>
</tr>
<tr>
<td>L1</td>
<td>Physical</td>
</tr>
</tbody>
</table>
We started with the fundamentals of routing and **reliable transport**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Component</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>L4</td>
<td>Transport</td>
<td>end-to-end delivery (reliable or not)</td>
</tr>
<tr>
<td>L3</td>
<td>Network</td>
<td>global best-effort delivery</td>
</tr>
<tr>
<td></td>
<td>Link</td>
<td>local best-effort delivery</td>
</tr>
<tr>
<td></td>
<td>Physical</td>
<td>physical transfer of bits</td>
</tr>
</tbody>
</table>
We saw three ways to compute valid routing state:

1. Use tree-like topologies
   - Example: Spanning-tree

2. Rely on a global network view
   - Example: Link-State, SDN

3. Rely on distributed computation
   - Example: Distance-Vector, BGP
We saw how to design a reliable transport protocol plus, you're implementing one right now

goals

**correctness** ensure data is delivered, in order, and untouched

**timeliness** minimize time until data is transferred

**efficiency** optimal use of bandwidth

**fairness** play well with other concurrent communications
In each case, we explored the rationale behind each protocol and why they came to be

Why did the protocols end up looking like this?
minimum set of features required

What tradeoffs do they achieve?
efficiency, cost,…

When is one design more adapted than another?
packet switching vs circuit switching, DV vs LS,…
We then climbed up the layers, starting from layer 2.
Communication Networks

Part 2: The Link Layer

#1 What is a link?

#2 How do we identify link adapters?

#3 How do we share a network medium?

#4 What is Ethernet?

#5 How do we interconnect segments at the link layer?
We then spent multiple weeks on layer 3
Internet Protocol and Forwarding

1. IP addresses
   use, structure, allocation

2. IP forwarding
   longest prefix match rule

3. IP header
   IPv4 and IPv6, wire format

source: Boardwatch Magazine
We also talked about IPv6
Internet routing
from here to there, and back

1. Intra-domain routing
   - Link-state protocols
   - Distance-vector protocols

2. Inter-domain routing
   - Path-vector protocols
Internet routing comes into two flavors: *intra-* and *inter-domain* routing

- **inter-domain routing**
  - Find paths between networks

- **intra-domain routing**
  - Find paths within a network
inter-domain routing

intra-domain routing

Find paths *between* networks
traffic to Google
Google can be reached via NEWY, WASH, ATLA, HOUS
Find paths within a network
NEWY can be reached via SALT
Border Gateway Protocol
policies and more

1 BGP Policies
   Follow the money

2 Protocol
   How does it work?

3 Problems
   security, performance, ...
Business relationships conditions

*route selection*

For a destination $p$, prefer routes coming from

- customers over
- peers over
- providers

*route type*
Business relationships conditions

route exportation

send to

customer peer provider

customer

from peer

provider
Routes coming from customers are propagated to everyone else.
Routes coming from peers and providers are only propagated to customers

<table>
<thead>
<tr>
<th>send to</th>
<th>customer</th>
<th>peer</th>
<th>provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>customer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>from peer</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>from provider</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
$4 = 3 + 1$
We looked at the requirements and implementation of transport protocols (UDP/TCP)

Data delivering, to the correct application
  • IP just points towards next protocol
  • *Transport needs to demultiplex incoming data (ports)*

Files or bytestreams abstractions for the applications
  • Network deals with packets
  • *Transport layer needs to translate between them*

Reliable transfer (if needed)
Not overloading the receiver
Not overloading the network
We then looked at **Congestion Control** and how it solves three fundamental problems:

1. **bandwidth estimation**
   - How to adjust the bandwidth of a single flow to the bottleneck bandwidth?
   - could be 1 Mbps or 1 Gbps...

2. **bandwidth adaptation**
   - How to adjust the bandwidth of a single flow to variation of the bottleneck bandwidth?

3. **fairness**
   - How to share bandwidth “fairly” among flows, without overloading the network
... by combining two key mechanisms

detecting congestion

reacting to congestion
We finally looked at what’s running on top of all this …

google.ch 172.217.16.131

http://www.google.ch
We finally looked at what’s running on top of all this ...

Video Streaming

HTTP-based

E-mail

MX, SMTP, POP, IMAP
We finally spoke about network programmability
Protocol Independent Switch Architecture (PISA) for high-speed programmable packet forwarding
Programmable networks: The future of networking?
Your final grade

- Exam: 80%
  - written, open book
- Projects: 20%
The exam will be open book, most of the questions will be open-ended, with some multiple choices to verify your understanding of the material.
Make sure you can do all the exercises, including the ones in previous exams

https://comm-net.ethz.ch/#tab-exam
Don't forget the assignments, they matter

No programming question, no Python at the exam

but we could ask you to describe a procedure in English

What would you change in your solution to achieve $X$?

No configuration question, no Quagga at the exam

but we could ask you to describe a configuration in English

How would you realize policy $X$?
Now you (better) understand this!

http://www.opte.org