Last week on Communication Networks
Video Streaming

HTTP-based

E-mail

MX, SMTP, POP, IMAP
Video Streaming

E-mail

HTTP-based
We want the highest video quality
Without seeing this …
Fast Internet

Screen size: 1920 x 1080 px With *fast* internet.

Video plays at **high quality**

1920 x 1080 px with no buffering

---

Slow Internet

Screen size: 1920 x 1080 px With *slower* internet.

Video plays at **medium quality**

1280x 720 px with no buffering
Simple solution for encoding: use a “bitrate ladders”

<table>
<thead>
<tr>
<th>Bitrate (kbps)</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>235</td>
<td>320x240</td>
</tr>
<tr>
<td>375</td>
<td>384x288</td>
</tr>
<tr>
<td>560</td>
<td>512x384</td>
</tr>
<tr>
<td>750</td>
<td>512x384</td>
</tr>
<tr>
<td>1050</td>
<td>640x480</td>
</tr>
<tr>
<td>1750</td>
<td>720x480</td>
</tr>
<tr>
<td>2350</td>
<td>1280x720</td>
</tr>
<tr>
<td>3000</td>
<td>1280x720</td>
</tr>
<tr>
<td>4300</td>
<td>1920x1080</td>
</tr>
<tr>
<td>5800</td>
<td>1920x1080</td>
</tr>
</tbody>
</table>

[netflix.com]
Your player download “chunks” of video at different bitrates
Depending on your network connectivity, your player fetches chunks of different qualities.
Your player gets metadata about chunks via "Manifest"

```xml
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="urn:mpeg:DASH:schema:MPD:2011"
    xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011"
    profiles="urn:mpeg:dash:profile:isoff-main:2011"
    type="static"
    mediaPresentationDuration="PT0H9M56.46S"
    minBufferTime="PT15.0S">
    <BaseURL>http://witestlab.poly.edu/~ffund/video/2s_480p_only/</BaseURL>
    <Period start="PT0S">
        <AdaptationSet bitstreamSwitching="true">
            <Representation id="0" codecs="avc1" mimeType="video/mp4"
                width="480" height="360" startWithSAP="1" bandwidth="101492">
                <SegmentBase>
                    <Initialization sourceURL="bunny_2s_100kbit/bunny_100kbit.mp4"/>
                </SegmentBase>
            </Representation>
        </AdaptationSet>
    </Period>
</MPD>
```
Encoding

Replication

Adaptation
Capacity < current rate ⇒ decrease rate
Buffer-based adaptation

Network

Nearly full buffer $\Rightarrow$ large rate
Buffer-based adaptation

Network

Nearly empty buffer $\Rightarrow$ small rate
Buffer-based adaptation

Buffer occupancy

Next chunk’s rate

R_{min}

R_{max}

Risk Area

Safe from Unnecessary rebuffering

Low buffer:

High buffer:

A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service, Huang et al., ACM SIGCOMM 2014
Video Streaming

E-mail

MX, SMTP, POP, IMAP
We looked at e-mail from three different perspectives

- **Content**
  - Format: Header/Content
  - Encoding: MIME

- **Infrastructure/Transmission**
  - SMTP: Simple Mail Transfer Protocol
  - IMAP: Internet Message Access Protocol

- **Retrieval**
  - POP: Post Office Protocol
Format: Header/Content

Encoding: MIME
Email relies on 7-bit U.S. ASCII...

How do you send non-English text? Binary files?

Solution

Multipurpose Internet Mail Extensions

commonly known as MIME, standardized in RFC 822
MIME defines

- additional headers for the email body
- a set of content types and subtypes
- base64 to encode binary data in ASCII
MIME relies on Base64 as binary–to–text encoding scheme

Relies on 64 characters out of the 128 ASCII characters the most common and printable ones, i.e. A-Z, a-z, 0-9, +, /

Divides the bytes to be encoded into sequences of 3 bytes each group of 3 bytes is then encoded using 4 characters

Uses padding if the last sequence is partially filled i.e. if the |sequence| to be encoded is not a multiple of 3
<table>
<thead>
<tr>
<th>Binary input</th>
<th>0x14fb9c03d97e</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bits</td>
<td>00010100 11111011 10011100</td>
</tr>
<tr>
<td></td>
<td>00000011 11011001 01111110</td>
</tr>
<tr>
<td>6-bits</td>
<td>000101 001111 101110 011100</td>
</tr>
<tr>
<td></td>
<td>000000 111101 100101 111110</td>
</tr>
<tr>
<td>Decimal</td>
<td>5 15 46 28 0 61 37 62</td>
</tr>
<tr>
<td>base64</td>
<td>F P u c A 9 1 +</td>
</tr>
</tbody>
</table>
Content

Infrastructure/Transmission

Retrieval

SMTP: Simple Mail Transfer Protocol

Infrastructure
mail servers
We can divide the e-mail infrastructure into five functions

<table>
<thead>
<tr>
<th>Mail</th>
<th>User</th>
<th>Agent</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Agent</td>
<td>Use to read/write emails (mail client)</td>
</tr>
<tr>
<td>Mail</td>
<td>Submission</td>
<td>Agent</td>
<td>Process email and forward to local MTA</td>
</tr>
<tr>
<td>Mail</td>
<td>Transmission</td>
<td>Agent</td>
<td>Queues, receives, sends mail to other MTAs</td>
</tr>
<tr>
<td>Mail</td>
<td>Delivery</td>
<td>Agent</td>
<td>Deliver email to user mailbox</td>
</tr>
<tr>
<td>Mail</td>
<td>Retrieval</td>
<td>Agent</td>
<td>Fetches email from user mailbox</td>
</tr>
</tbody>
</table>
MSA/MTA/MDA and MRA/MUA are often packaged together leading to simpler workflows.
Simple Mail Transfer Protocol (SMTP) is the current standard for transmitting e-mails

SMTP is a text-based, client-server protocol
client sends the e-mail, server receives it

SMTP uses reliable data transfer
built on top of TCP (port 25 and 465 for SSL/TLS)

SMTP is a push-like protocol
sender pushes the file to the receiving server (no pull)
The sender MUA uses SMTP to transmit the e-mail first to a local MTA (e.g. mail.ethz.ch, gmail.com, hotmail.com)
The local MTA then looks up the MTA of the recipient domain (DNS MX) and transmits the e-mail further.
Today on Communication Networks
ICMP

Network Control Messages

its use for discovery

NAT

Network Address Translation

its use for sharing IPs

+ a little bit of SDN and course recap.
Network Control Messages

its use for discovery
What Errors Might A Router See?

- Dead-end: No route to destination
- Sign of a loop: TTL expires
- Can’t physically forward: packet too big
  - And has DF flag set
- Can’t keep up with traffic: buffer overflowing
- Header corruption or ill-formed packets
- ....
What should network tell host about?

- No route to destination?
  - Host can’t detect or fix routing failure.
- TTL expires?
  - Host can’t detect or fix routing loop.
- Packet too big (with DF set)?
  - Host can adjust packet size, but can’t tell difference between congestion drops and MTU drops
- Buffer overflowing?
  - Transport congestion control can detect/deal with this
- Header corruption or ill-formed packets?
  - Host can’t fix corruption, but can fix formatting errors
Router Response to Problems?

- Router doesn’t really need to respond
  - Best effort means never having to say you’re sorry
  - IP could conceivably just silently drop packets

- Network is already trying its best
  - Routing is already trying to avoid loops/dead-ends
  - Network can’t reduce packet size (in DF packets)
  - Network can’t reduce load, nor fix format problems

- What more can/should it do?
Error Reporting Helps Diagnosis

- Silent failures are really hard to diagnose
- IP includes feedback mechanism for network problems, so they don’t go undetected
- Internet Control Message Protocol (ICMP)
- The Internet “print” statement
- Runs on IP, but viewed as integral part of IP
Internet Control Message Protocol

- Triggered when IP packet encounters a problem
  - E.g., *Time Exceeded* or *Destination Unreachable*

- ICMP packet sent back to the source IP address
  - Includes the error information (e.g., type and code)
  - IP header plus 8+ byte *excerpt* from original packet

- Source host receives the ICMP packet
  - Inspects *excerpt* (e.g., protocol/ports) to identify socket

**Exception**: not sent if problem packet is ICMP
  - And just for fragment 0 of a group of fragments
Types of Control Messages

- **Need Fragmentation**
  - IP packet too large for link layer, DF set

- **TTL Expired**
  - Decremented at each hop; generated if $\rightarrow 0$

- **Unreachable**
  - Subtypes: network / host / port
    - (who generates Port Unreachable?)

- **Source Quench**
  - Old-style signal asking sender to slow down

- **Redirect**
  - Tells source to use a different local router
Using ICMP

- ICMP intended to tell host about network problems
  - Diagnosis
  - Won’t say more about this….

- Can exploit ICMP to elicit network information
  - Discovery
  - Will focus on this…. 
Discovering Network Path Properties

- **PMTU Discovery**: Largest packet that can go through the network w/o needing fragmentation
  - Most efficient size to use
  - (Plus fragmentation can amplify loss)

- **Traceroute**: What is the series of routers that a packet traverses as it travels through the network?

- **Ping**: Simple RTT measurements
Ping: Echo and Reply

- ICMP includes simple “echo” functionality
  - Sending node sends an ICMP Echo Request message
  - Receiving node sends an ICMP Echo Reply

- Ping tool
  - Tests connectivity with a remote host
  - … by sending regularly spaced Echo Request
  - … and measuring delay until receiving replies
Path MTU Discovery

- **MTU** = Maximum Transmission Unit
  - Largest IP packet that a link supports
- **Path MTU (PMTU)** = minimum end-to-end MTU
  - Must keep datagrams no larger to avoid fragmentation
- How does the sender know the PMTU is?
- **Strategy (RFC 1191):**
  - Try a desired value
  - Set **DF** to prevent fragmentation
  - Upon receiving **Need Fragmentation** ICMP …
    - … oops, that didn’t work, try a smaller value
Issues with Path MTU Discovery

• What set of values should the sender try?
  • Usual strategy: work through “likely suspects”
  • E.g., 4352 (FDDI), 1500 (Ethernet),
    1480 (IP-in-IP over Ethernet), 296 (some modems)
• What if the PMTU changes? (how could it?)
  • Sender will immediately see reductions in PMTU (how?)
  • Sender can periodically try larger values
• What if Needs Fragmentation ICMP is lost?
  • Retransmission will elicit another one
• How can The Whole Thing Fail?
  • “PMTU Black Holes”: routers that don’t send the ICMP
Discovering Routing via *Time Exceeded*

- Host sends an IP packet
  - Each router decrements the time-to-live field
- If **TTL** reaches 0
  - Router sends *Time Exceeded* ICMP back to the source
  - Message identifies router sending it
    - Since ICMP is sent using IP, it’s just the IP source address
    - And can use PTR record to find name of router
Traceroute: Exploiting *Time Exceeded*

- Time-To-Live field in IP packet header
  - Source sends a packet with TTL ranging from 1 to *n*
  - Each router along the path decrements the TTL
  - “TTL exceeded” sent when TTL reaches 0
- *Traceroute* tool exploits this TTL behavior

**Diagram:**
- Source sends packets with TTL=1, 2, ...
- Time exceeded

Send packets with TTL=1, 2, ...
and record source of *Time Exceeded* message
Network Address Translation

its use for sharing IPs
Sharing Single Address Across Hosts

- Network Address Translation (NAT) enables many hosts to share a single address
  - Uses port numbers (fields in transport layer)

- Was thought to be an architectural abomination when first proposed, but it:
  - Probably saved us from address exhaustion
  - And reflects a modern design paradigm (indirection)
Special-Purpose Address Blocks

- **Limited broadcast**
  - Sent to every host attached to the local network
  - Block: `255.255.255.255/32`

- **Loopback**
  - Address blocks that refer to the local machine
  - Block: `127.0.0.0/8`
  - Usually only `127.0.0.1/32` is used

- **Link-local**
  - By agreement, not forwarded by any router
  - Used for single-link communication only
  - Intent: autoconfiguration (especially when DHCP fails)
  - Block: `169.254.0.0/16`

- **Private addresses**
  - By agreement, not routed in the public Internet
  - For networks not meant for general Internet connectivity
  - Blocks: `10.0.0.0/8`, `172.16.0.0/12`, `192.168.0.0/16`
Network Address Translation (NAT)

Before NAT...every machine connected to Internet had unique IP address
NAT (cont’d)

- Assign addresses to machines behind same NAT
  - Can be any private address range
  - e.g. 192.168.0.0/16
- Use **port numbers** to multiplex single address
NAT (cont’d)

- Assign addresses to machines behind same NAT
  - Usually in address block \textbf{192.168.0.0/16}
- Use port numbers to multiplex single address
NAT: Early Example of “Middlebox”

- Boxes stuck into network to deliver functionality
  - NATs, Firewalls,….

- Don’t fit into architecture, violate E2E principle

- But a very handy way to inject functionality that:
  - Does not require end host changes or cooperation
  - Is under operator control (e.g., security)

- An interesting architectural challenge:
  - How to incorporate middleboxes into architecture
Programmable Data Planes: The future of networking?
Programming
The Network Data Plane

Changhoon Kim
Beautiful ideas: What if you could ...

• Realize a small, but super-fast DNS cache
• Perform TCP SYN authentication for billions of SYNs per sec
• Build a replicated key-value store ensuring RW ops in a few usecs
• Improve your consensus service performance by ~100x
• Boost your Memcached cluster’s throughput by ~10x
• Speed up your DNN training dramatically by realizing parameter servers

... using *switches* in your network?
You couldn’t do any of those so far because ...

• No DIY – must work with vendors at feature level
• Excruciatingly complicated and involved process to build consensus and pressure for features
• Painfully long and unpredictable lead time
• To use new features, you must get new switches
• What you finally get != what you asked for
This is very unnatural to developers

• Because you all know how to realize your own ideas by “programming” CPUs
  – Programs used in every phase (implement, test, and deploy)
  – Extremely fast iteration and differentiation
  – You own your own ideas
  – A sustainable ecosystem where all participants benefit

Can we replicate this healthy, sustainable ecosystem for networking?
Reality: Packet forwarding speeds

Gb/s (per chip)


100000 10000 10000 1000 100 10 1 0.1

6.4Tb/s

Switch Chip
Reality: Packet forwarding speeds

Gb/s (per chip)

<table>
<thead>
<tr>
<th>Year</th>
<th>Switch Chip</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1990</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>1995</td>
<td></td>
<td>1.1</td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td>1.2</td>
</tr>
<tr>
<td>2005</td>
<td></td>
<td>1.3</td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>2015</td>
<td></td>
<td>1.5</td>
</tr>
<tr>
<td>2020</td>
<td>6.4Tb/s</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Switch Chip: 80x increase from 1990 to 2020.
CPU: Increase from 1990 to 2020.
What does a typical switch look like?

A switch is just a Linux box with a high-speed switching chip.
Networking systems have been built “bottoms-up”

“This is roughly how I process packets ...”
Turning the tables “top-down”

“This is precisely how you must process packets”

in P4
Evidence: Tofino 6.5Tb/s switch (arrived Dec 2016)

The world’s fastest and most programmable switch. No power, cost, or power penalty compared to fixed-function switches. An incarnation of PISA (Protocol Independent Switch Architecture)
Domain-specific processors

Computers
- Java Compiler
- OpenGL Compiler

Graphics
- Signal Processing
  - Matlab Compiler

Machine Learning
- TensorFlow Compiler

Networking
- Language Compiler

CPU

GPU

DSP

TPU
Domain-specific processors

Computers
- Java Compiler
- OpenGL Compiler

Graphics
- Matlab Compiler

Signal Processing
- Matlab Compiler

Machine Learning
- TensorFlow Compiler

Networking
- P4 Compiler

PISA
PISA: An architecture for high-speed programmable packet forwarding
PISA: Protocol Independent Switch Architecture
PISA: Protocol Independent Switch Architecture
PISA: Protocol Independent Switch Architecture

**Match Logic**
(Mix of SRAM and TCAM for lookup tables, counters, meters, generic hash tables)

**Action Logic**
(ALUs for standard boolean and arithmetic operations, header modification operations, hashing operations, etc.)

Programmable Packet Generator

Programmable Parser

Ingress match-action stages (pre-switching)

Buffer

Egress match-action stages (post-switching)

Recirculation

CPU (Control plane)

Generalization of RMT [sigcomm’13]
Why we call it protocol-independent packet processing
Device does not understand any protocols until it gets programmed

Logical Data-plane View (your P4 program)

Switch Pipeline

Queues
Mapping logical data-plane design to physical resources

Logical Data-plane View (your P4 program)

Switch Pipeline

Queues
Re-program in the field

Logical Data-plane View (your P4 program)

Switch Pipeline
P4.org (http://p4.org)

- **Open-source community to nurture the language**
  - Open-source software – Apache license
  - A common language: P4
  - Support for various types of devices and targets

- **Enable a wealth of innovation**
  - Diverse “apps” (including proprietary ones) running on commodity targets

- **With no barrier to entry**
  - Free of membership fee, free of commitment, and simple licensing
If you are interested, consider taking
Advanced Topics in Communication Networks [adv-net.ethz.ch]

This class will introduce students to advanced, research-level topics in the area of communication networks, both theoretically and practically. Coverage will vary from semester to semester. Repetition for credit is possible, upon consent of the instructor. During the Fall Semester of 2018, the class will concentrate on network programmability and network data plane programming.

Lectures
Weekly lectures in the first part of the semester (more details coming soon)

Exercises
Ungraded theoretical and practical exercises as well as paper readings (more details coming soon)

Project
Graded practical project performed in groups (more details coming soon)
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>Application</td>
<td></td>
<td>network access</td>
</tr>
<tr>
<td>L4 Transport</td>
<td></td>
<td>end-to-end delivery (reliable or not)</td>
</tr>
<tr>
<td>L3 Network</td>
<td></td>
<td>global best-effort delivery</td>
</tr>
<tr>
<td>Link</td>
<td>Link</td>
<td>local best-effort delivery</td>
</tr>
<tr>
<td>Physical</td>
<td></td>
<td>physical transfer of bits</td>
</tr>
</tbody>
</table>
We saw three ways to compute valid routing state

<table>
<thead>
<tr>
<th>Intuition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1 Use tree-like topologies</td>
<td>Spanning-tree</td>
</tr>
<tr>
<td>#2 Rely on a global network view</td>
<td>Link-State</td>
</tr>
<tr>
<td>#3 Rely on distributed computation</td>
<td>Distance-Vector</td>
</tr>
<tr>
<td></td>
<td>SDN</td>
</tr>
<tr>
<td></td>
<td>BGP</td>
</tr>
</tbody>
</table>
We saw how to design a reliable transport protocol.

**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
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
Border Gateway Protocol
policies and more

1. BGP Policies
   Follow the Money

2. Protocol
   How does it work?

3. Problems
   security, performance, ...
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 …
We finally looked at what’s running on top of all this …

- Video Streaming
  - HTTP-based
- E-mail
  - MX, SMTP, POP, IMAP
... and filled-up some holes with 2 helpers protocols

ICMP
Network Control Messages
its use for discovery

NAT
Network Address Translation
its use for sharing IPs
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 at the exam

but I could ask you to describe a procedure in English.

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

No configuration question at the exam

but I could ask you to describe a configuration in English.

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