6.1800 Spring 2024

Lecture #13: The application layer
serving content as content evolves
**bandwidth-based scheduling:** can we allocate specific amounts of bandwidth to some traffic?

**round robin:** can’t handle variable packet sizes (and in its most basic form doesn’t allow us to weight traffic differently)

**deficit round robin:** handles variable packet sizes (even within the same queue), near-perfect fairness and low packet processing overhead

---

**in each round:**

- for each queue q:
  - if q is not empty:
    - `q.credit += q.quantum`
  
  - while `q.credit >= size of next packet p`:
    - `q.credit -= size of p`
    - send p
  - else:
    - `q.credit = 0`

---

the quantums for each queue are chosen to be **proportionate to the packet sizes**. too big and we have poor short-term fairness, too small and it just takes too long to build credit.
standardizations let us communicate across machines

TCP, DNS, OSPF (link-state routing), etc., all have standards that describe the protocols in detail including packet formats

You won't be able to find these new emoji on your keyboard just yet, though. While version 15.1.0 of the Unicode was approved on Sept. 12 (you can read all of it here), there's always a lag between the approval and its release. Emojipedia had estimated that the new emoji would be available in early 2024.
6.1800 in the news

who sets the standards?

Having a company like Apple literally influencing our language is probably not a great way of doing things.

Keith Weinstein is a Computer Science Professor at Stanford.

read this comic and let my friend keith explain it to you!

https://thenib.com/who-makes-emoji/

Ready for New iPhone Emojis? Here Are the 118 Icons Coming in iOS 17.4

The iPhone adds more emoji you never knew you needed.


<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>hosts.txt</td>
<td>distance-vector routina</td>
<td>TCP, UDP</td>
<td>OSPF, EGP, DNS</td>
<td>congestion collapse</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>policy routing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CIDR</td>
</tr>
</tbody>
</table>

- application: the things that actually generate traffic
  - transport: sharing the network, reliability (or not)
    - examples: TCP, UDP
  - network: naming, addressing, routing
    - examples: IP
  - link: communication between two directly-connected nodes
    - examples: ethernet, bluetooth, 802.11 (wifi)

**today:** how do all of the lower layers affect application-layer protocols? specifically, how do we **deliver content** on the Internet?

CAIDA's IPv4 AS Core, January 2020
(https://www.caida.org/projects/cartography/as-core/2020/)
how do we share a file — or deliver content — on the Internet?

we know that a client-server model is (relatively) simple, but doesn’t scale well; let's understand more about the other two technologies, to see where they end up in terms of complexity, scalability, etc.

as part of this endeavor, we'll also see why the underlying network matters in these designs
1. download .torrent file from known website

2. contact tracker for list of peers

3. communicate with (some) peers to download and upload blocks

seeders have the entire file

.tracker file
- file name
- file size
- information about the “blocks” of the file
- tracker URL

.list of peers
question: are there any **incentives** for peers to upload data to another peer? are there any **drawbacks**?
how do we incentivize users to upload?

round $t$

13 ← 10
4 ← 12
7 ← 9
15 ←

round $t+1$

→
→
→
→
question: are there any central points of failure in this network?
.torrent file
- file name
- file size
- information about the “blocks” of the file
- tracker URL

most modern BitTorrent clients used a decentralized tracker, where no machine in the tracker network knows the full set of peers.
requires some specific organization of the content (e.g., well-defined “blocks”), the ability to discover other peers, and some incentives to get users to upload. In practice, scalability is limited by end-users’ upload constraints.
1. geographically distribute the servers

2. replicate a particular piece of content $p$ on some of them

3. when a client requests $p$, direct them to the “best” server that has a copy of $p$

question: what do you think makes a server the “best”?
1. geographically distribute the servers

2. replicate a particular piece of content \( p \) on some of them

3. when a client requests \( p \), direct them to the “best” server that has a copy of \( p \)
CDNs

requires a great deal of coordination and organization among the edge servers (all of which are owned by a single company). not as “organic” as P2P networks, but can provide better performance guarantees, in part by finding alternate routes and improving transport-layer performance.

P2P

requires some specific organization of the content (e.g., well-defined “blocks”), the ability to discover other peers, and some incentives to get users to upload. in practice, scalability is limited by end-users’ upload constraints.

doesn’t scale well, but a lot (a lot) less complicated than CDNs!
the technologies for sharing content on the Internet have changed as the way we use the Internet has changed

the underlying network affects how well these technologies work, and there are also interesting challenges in terms of how to keep data up-to-date and consistent across multiple machines, and how to deal with failures

these are challenges we’ll address starting after spring break
on the Internet, we have to solve all of the “normal” networking problems (addressing, routing, transport) at massive scale, while supporting a diverse group of applications and competing economic interests.