Transport Layer & Flows

Flows

Definition 3.1 (Flow, Rate). *Let s, t be two nodes in a directed graph. A* ***flow*** *from* **source** *s to* **destination** *t (also called an* ***s-t-flow****) is a function $F : E \to \mathbb{R}_{\geq 0}$ such that the following hold:*

$$F(e) \leq c(e) \qquad \text{for all } e \in E \qquad \text{(capacity constraints)}$$

$$\sum_{e \in in(v)} F(e) = \sum_{e \in out(v)} F(e) \text{ for all } v \in V \setminus \{s,t\} \text{ (flow conservation)}$$

We call $F(e)$ the **rate** *of F on directed edge e and the net flow leaving s $\left(\sum_{e \in out(s)} F(e) - \sum_{e \in in(s)} F(e)\right)$ the* **rate** *of F, also denoted by F.*

Multi-commofity Flows

Definition 3.2 (Multi-Commodity Flow). *A* ***multi-commodity flow*** $\mathcal{F} = (F_1, ..., F_k)$ *is a collection of* s_i*-*t_i*-flows* F_i *such that for each edge* $e \in E$ *the sum of the flows' rates on* e *does not exceed the capacity of* e*, i.e.,*

$$\sum_{i=1}^{k} F_i(e) \leq c(e) \qquad \textit{for all } e \in E.$$

- **Commodity:** source-destination pair.
- All flows in a multi-commodity flow should satisfy **flow conservation**.

Multi-commodity flows are harder!

Linear Programming

- Applied to wide range of optimization problems.

- **Optimization problem:** maximize o minimize a function given some constrains.

- In Linear Programming the function and the constraints are linear.

Cocktail Party Example

Minimize $f(\mathbf{x}) = x_1 + 3x_2$
subject to

1. $x_1 + x_2 \geq 50$
2. $x_1 + \frac{1}{2}x_2 \leq 30$
3. $x_1 \geq 0$
4. $x_2 \geq 0$

Figure 3.4: Linear program for throwing a party

Note that function and constraints are indeed **linear.**

Linear Program (LP) – Canonical Form

Definition 3.5 (Linear Program, LP). *A* ***linear program (LP)*** *consists of a set of m inequalities*

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \ldots & + & a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \ldots & + & a_{2n}x_n & \leq & b_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \ldots & + & a_{mn}x_n & \leq & b_m \end{array}$$

and a linear function

$$f(\mathbf{x}) = c_1x_1 + c_2x_2 + \cdots + c_nx_n \quad .$$

The a_{ji}, b_i and c_i are given real-valued parameters and a vector $\mathbf{x} = (x_1, \ldots, x_n)^T$ is a solution to the linear program if $x_i \geq 0$ for all $1 \leq i \leq n$ and $\mathbf{x}$ maximizes $f(\mathbf{x})$.

Linear Program (LP)

- Short notation of the canonical form: $\max\{\mathbf{c}^T\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0\}$
- A linear optimization problem can alwalys be written in **canonical form**.

- **Geometrical interpretation:**
  - An LP corresponds to an n-dimensional **convex polytope**.
  - The hyperplanes bounding the polytope are given by the **constraints**.
  - The maximum is attained in one of the **vertexes** of the polytope.

Simplex Algorithm

- Algorithm to solve LPs:

Algorithm 3.6 Simplex Algorithm

```
choose a vertex x of the polytope
while there is a neighboring vertex y such that f(y) > f(x) do
    x := y
end while
return x
```

Solving **integer** LP is usually NP-hard

Commodity Flows as LPs

x_e is a variable indicating the amount of flow in edge *e*

Maximize $f(\mathbf{x}) = \sum_{e \in \mathtt{out}(s)} x_e$
subject to

1. $x_e \geq 0$ for all $e \in E$
2. $x_e \leq c(e)$ for all $e \in E$
3. $\sum_{e \in \mathtt{in}(v)} x_e = \sum_{e \in \mathtt{out}(v)} x_e$ for all $v \in V \setminus \{s, t\}$
4. $\sum_{e \in \mathtt{in}(s)} x_e = 0$

1. The amount of flow is non-negative in each edge.
2. Edge capacities are not violated.
3. Flow conservation
4. Avoid returning flow

Unsplittable Flow

Definition 3.8 (Unsplittable Flow). *An s-t-flow F is called* ***unsplittable*** *if the edges $e \in E$ with $F(e) > 0$ form a path from s to t. If we do not impose this path restriction on a flow, it is called* ***splittable****.*

- The notion of unsplittable flow extends to multi-commodity flows.
- If the path is not unsplittable, simple LP cannot solve the problem.

Fairness

- **Demand:** rate at which a flow wants to transmit.
- If we only maximize throughput, some flows may starve:

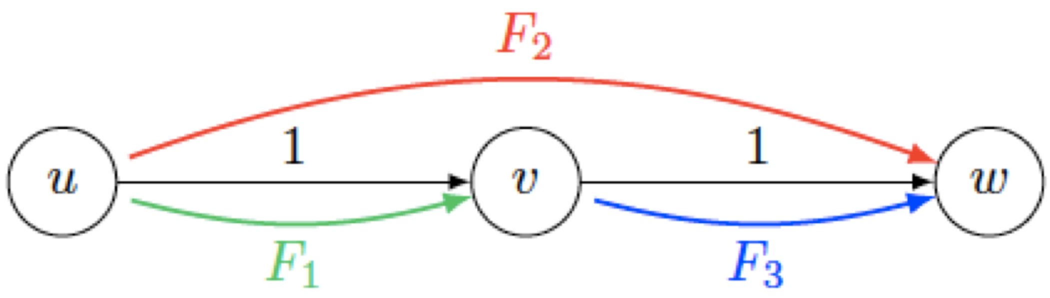

Figure 3.10: We have three flows, all with demand 1.

We need som **fairness** in the network!

Max-min Fairness

Definition 3.11 (Max-Min-Fairness). *A bandwidth allocation is called* ***max-min-fair*** *if increasing the allocation of a flow would necessarily decrease the allocation of a smaller or equal-sized flow.*

There is only one max-min-fair allocation in a network.

Max-Min-Fair Algorithm

Algorithm 3.12 Max-Min-Fair Allocation

Given a graph G, a set $\mathcal{F} = \{F_1, \ldots, F_k\}$ of flows with initial rate 0 on all edges, paths $p_1, \ldots, p_k$ along which the respective flows are to be routed and demands $d_1, \ldots, d_k$

while $\mathcal{F} \neq \emptyset$ **do**
    repeat
        increase rate of all flows in $\mathcal{F}$ evenly, but at most up to the respective demands
    until there is an edge $e \in E$ such that $\sum_{i:e\in p_i} F_i = c(e)$
    for all such edges e **do**
        for all i such that $e \in p_i$ **do**
            $\mathcal{F} := \mathcal{F} \setminus \{F_i\}$
        end for
        $E := E \setminus \{e\}$
    end for
end while

Ports

Definition 3.13 (Port). *A **port** is a numeric identifier used in transport protocols to identify which application sent the packet and which application should receive it on the destination computer.*

Ports provide **distinction** between different applications.

Client-Server Model

Definition 3.14 (Client-Server Model). *In the client-server model, the **client** actively initiates the communication, while the **server** passively waits for a client to connect. The client is regarded as a consumer of the services offered by the server.*

When communicating with a server the client transmits its port, so that the server knows where to reply.

UDP

Protocol 3.15 (UDP). *The user datagram protocol (UDP) is a no-frills transport protocol that allows an application to send packets from client to server.*

- UDP is **encapsulated** inside the payload of the IP packet.
- UDP header: source and destination **ports**, checksum, length.
- UDP is **connectionless**.
- UDP does not handle packet **loss** and does not guarantee any **order**.
- UDP does not provide **congestion control**.

But, has little size and latency overhead -> **Good for real-time!**

TCP

Protocol 3.17 (TCP). *The transmission control protocol (TCP) is a connection-oriented transport protocol guaranteeing that lost packets are being retransmitted and that packets are delivered in the same order they are sent.*

- **Connection:** bidirectional long-term relationship between a client and a server to transmit data.
- TCP is **encapsulated** inside the payload of the IP packet.
- TCP also uses **ports** to distinguish between applications.
- TCP establishes a connection and does not release resources until all data is transmitted.
- TCP abstracts data packets into a **continuous stream**.

Connection Establishment

- **Acknowledgment (ACK):** Confirmation that a packet has been received.
  - ACK number is the last data byte of the received packet plus 1.
  - May be void of actual data.

- **Three-way handhsake** for establishing a connection:
  - The client sends a SYN (synchronize) packet to the server.
  - The server acknowledges the packet by sending back a SYN/ACK packet.
  - The client acknowledges the recpetion of the SYN/ACK packet by sending an ACK.
- A connection is terminated replacing SYN for FIN.

Flow Control and Congestion Control

- **Flow control:** avoiding congestion on the receipient's side.
  - The receiver specifies how many bits it can receive.

- **Congestion control:** the sender uses a window to limit the bits to send.
- The (congestion) window is controlled with AIMD.

- **AIMD, Additive Increase/Multiplicative decrease:**
  - No congestion: increase window additively
  - Congestion: decrease window multiplicatively (half in TCP)

Congestion

- Congestion is detected when a packet is dropped = no ACK is received
- **Round-Trip Time (RTT):** Time it takes a packet to travel from sender to receiver and back.
- **Timeout:** if a packet is not acknowledged in some time frame it is considered to be lost -> In TCP timeout is smoothed RTT.
- AIMD roughly converges to a max-min-fair allocation.
- **Slow start:** multiplicative increase at the beginning until congestion is detected, then change to AIMD.
  - Accelerate start-up.

NAT

Protocol 3.17 (TCP). *The transmission control protocol (TCP) is a connection-oriented transport protocol guaranteeing that lost packets are being retransmitted and that packets are delivered in the same order they are sent.*

- Shortage of IPv4 addresses: all machines behind an entry node (router) get **private addresses**.
  - Private addresses are not unique: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0./16
- Nodes outside private network cannot route to private addresses.

Solution: the entry router **translates** ports into IP addresses.