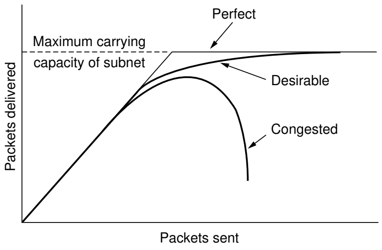
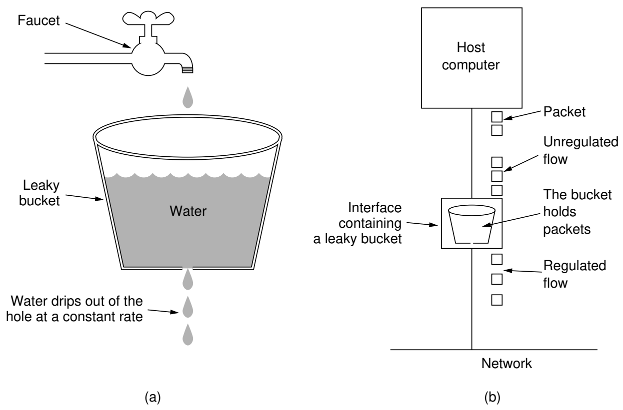
Congestion

Problem: when too many packets have to be transmitted through the network, we can get into a serious performance problem:



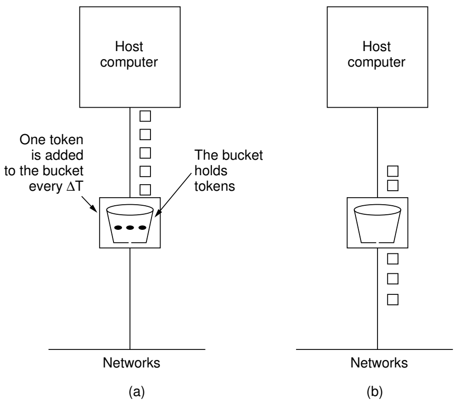
- Congestion can be caused by lack of bandwidth, but also by ill-configured or slow routers.
- Make a distinction between open loop and closed loop solutions: is feedback provided (closed) or not (open)?
- Avoid congestion by avoiding bursts: **shape** your packet traffic, and let the network provider do the **traffic policing** (monitor the flow).

Leaky Bucket



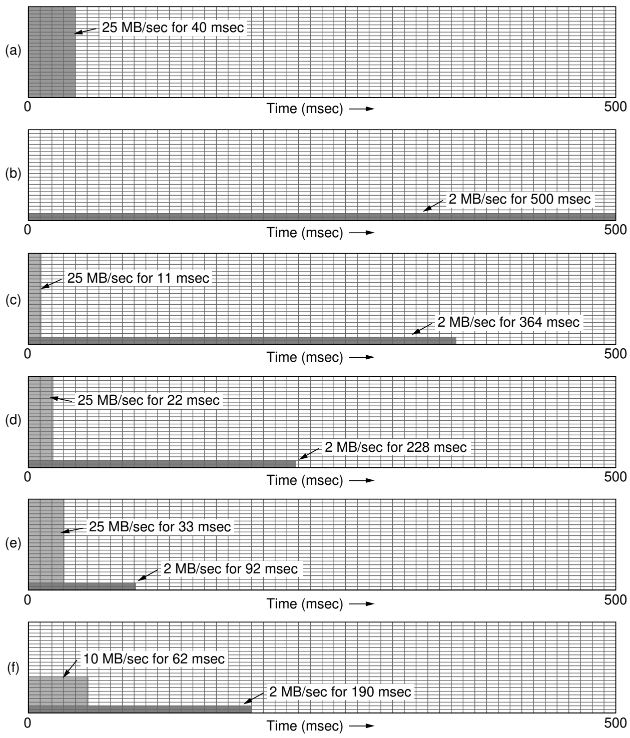
Note: we've eleminated bursts completely: packets are passed to the network when available, and all at the same rate. This may be a bit overdone. Also, packets can get lost.

Token Bucket



- Tokens are added at a constant rate; as soon as enough tokens have been saved, one or more packets can be sent.
- To avoid still too much burstiness, put a leaky bucket behind a token bucket (with a larger rate – why?).

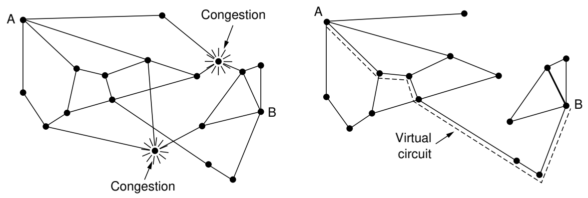
Effects of Buckets



Congestion Control – Virtual Circuits

Principal: when you set up a circuit, be sure that congestion can be avoided.

- Admission control: if it's too busy, just refuse to set up a virtual circuit. This is the same as refusing new users at an FTP site.
- Select alternative routes when a part in the network is getting overloaded (i.e., temporarily rebuild your view of the network):



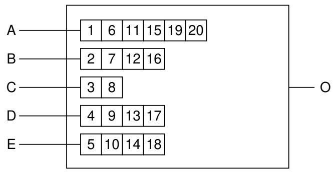
(a) (b)

- Negotiate the quality of the circuit in advance, so that the network provider can reserve buffers and the like. Resources are guaranteed to be there.

Choke packets

Basic idea: Send info back to the source when the networks starts to perform bad. A router checks the status of an output line: if it's too occupied, it sends a choke packet to the source. The host is assumed to be cooperative, and that it will slow down (not always a good assumption).

Better idea: To enforce cooperation, use **weighted fair queuing**:



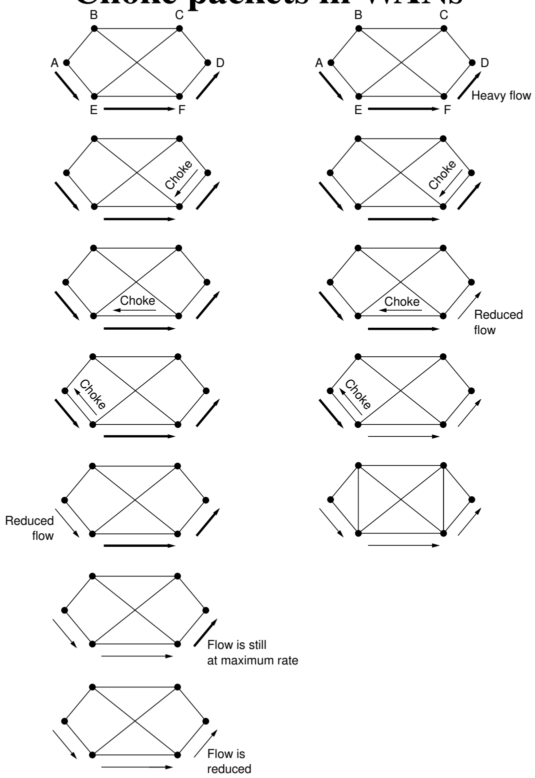
(a)

Packet	Finishing time
C	8
B	16
D	17
E	18
A	20

(b)

Problem: The return path for a choke packet may be so long, that the destination is going to get into trouble anyway – there may simply be too much on the way already. Then use a "push-back" approach:

Choke packets in WANs



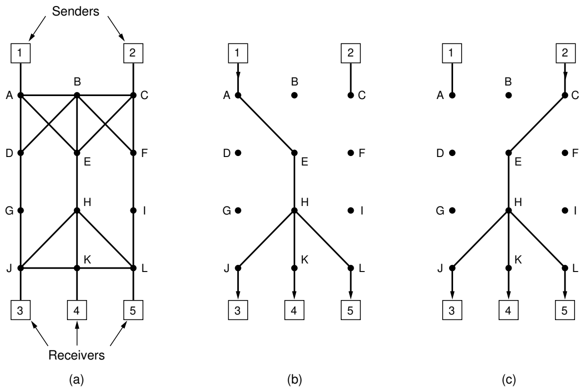
(a) (b)

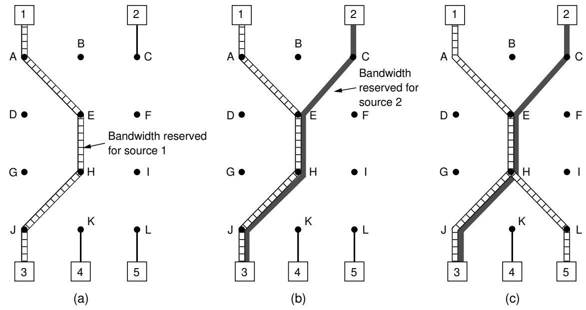
Multicasting & Congestion Control

This is fun: In the advent of all kinds of services on the Internet that deal with broadcasting streams of data (voice and video), and the limited bandwidth, we need a way to avoid that streams don't come through anymore.

Basic idea: We set up multicast trees from sources to destinations, but this time take into account the bandwidth needed by the receivers. Bandwidth can be reserved on **pre-constructed** trees. If there's not enough bandwidth as required by the receiver, a failure is reported back.

Resource reSerVation Protocol





Internetworking

Problem: Great! We have all these networks, with all different protocol stacks, and now we just to let them talk to each other.

Non-Solution: Enforce all networks to run the same protocol stack. That's asking for a lot of trouble, and effectively saying that we are not allowed to make any progress.

Solution: Construct all kinds of **gateways** that connect to different kinds of networks:

- **repeaters** at the physical layer for boosting signals.
- **bridges** to make the interconnection at the data link layer.
- **multiprotocol routers** for forwarding, and possibly splitting up packets (bridges can't do the latter).
- **transport gateways** for coupling *byte streams* in different networks.
- **application gateways**, e.g., for handling electronic mail between OSI and TCP/IP networks.

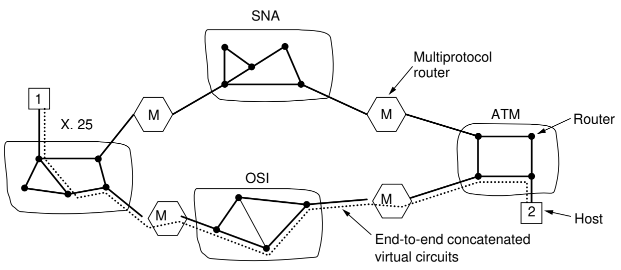
The Differences

Issue	Differences
Service offered (*)	Connection-oriented vs connectionless
Protocols (*)	IP, IPX, CLNP, AppleTalk, etc.
Addressing	Flat (802) vs hierarchical
Multicasting	Present versus absent
Packet size (*)	Network-specific maximum
Quality of Service	None versus a lot versus who knows
Error handling	Reliable, (un)ordered delivery
Flow control	Sliding window, rate control, etc.
Congestion control	Leaky bucket, choke packets, etc.
Security	Privacy rules, encryption, etc.
Parameters	Timeouts, flow specs, etc.
Accounting	Connect time, packets, bytes, none

Don't worry: it's impossible to resolve all differences. The solution is to just take a simple approach (like the Internet). We now only consider the starred (*) issues.

Concatenated Virtual Circuits

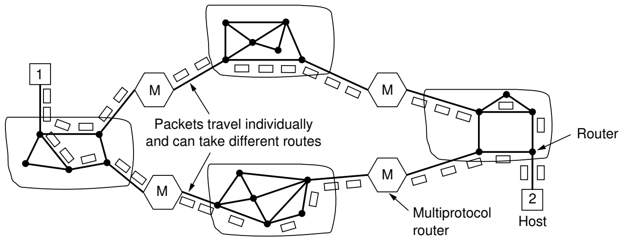
Basic idea: Assume the constituent networks support virtual circuits. In that case, the internet can use virtual circuit technology through concatenation.



Note: If one of the constituent networks does not support virtual circuits, or, for example, provides only unreliable data transport, simple concatenation will be hard.

Using Datagrams

Basic idea: Have the network layer offer only datagram services: unreliable, unorderded packet flow. Often, connection-oriented services are not even supported (Internet).

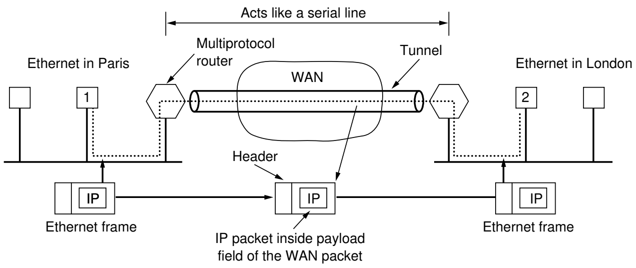


Main problem: Addressing – different networks use completely different addresses, so how do I address a host in an IP network when I've only got CLNP?

Solution: You don't solve it, but instead consider, for example, IP as a universal network protocol. Lots of universal network protocols are available

Tunneling

Basic idea: We can solve a lot of the internetworking problems when we can assume that the source and destination are on the same type of network. In that case, we need only to **tunnel** packets through intermediate networks.

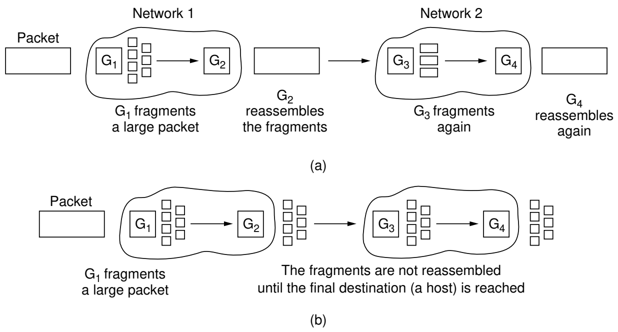


Question: We already came across tunneling somewhere else. Where?

Fragmentation

Problem: Different networks may impose different maximum packet sizes. This means that we may have to split a packet into smaller ones when forwarding it through an intermediate network ⇒ **fragmentation**.

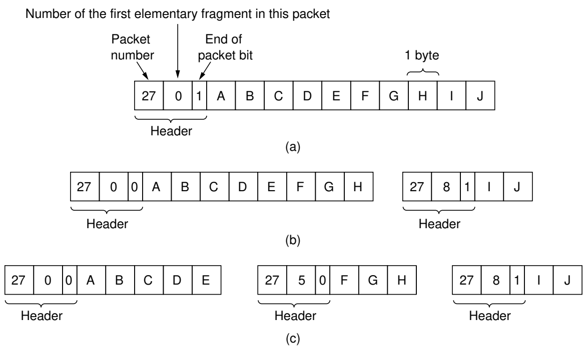
Note: We had the same problem when dealing with bridges, but it couldn't be solved there (why not?).



Fragmentation – reassembly

Problem: When we create fragments, how do we paste them together again:

- a fragment may be fragmented again by successive intermediate networks ⇒ when a fragment arrives at the destination, we have to know exactly where it fits into the original packet. **Solution:**

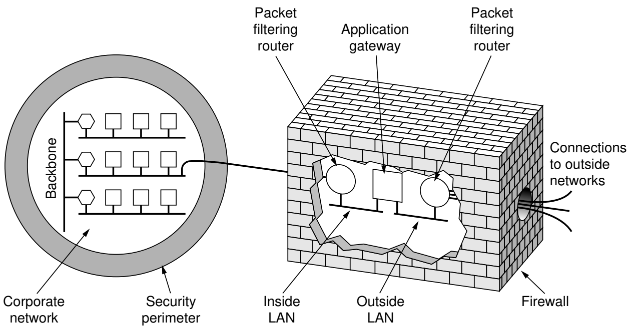


- fragments may pass through unreliable networks, and may be lost ⇒ we need to decide what to do about lost fragments. **Solution:** Discard the entire packet (what do you think will happen then?).

Firewalls

Problem: Sometimes it's better when not all your computers can be easily reached by others, or vice versa.

Non-solution: protect your company by a firewall. It's a non-solution from a theoretical point of view. It's a great solution in practice.



Question: What do you think would be the biggest breach in firewalls?