The teleology of Internet congestion control

Damon Wischik, Computer Science, UCL

What network am I looking at?

There is the physical network, that describes which computers and switches are connected by which links.

There is the logical network, that describes which users are using which paths through the network.

This talk is about traffic flows. Each traffic flow belongs to some user, and is carried over some collection of computers and switches. One user's traffic flow influences other users' flows indirectly, by virtue of the links they use in common.

teleology

from the Greek *τελος* (end) + *-λογια* (discourse, study)

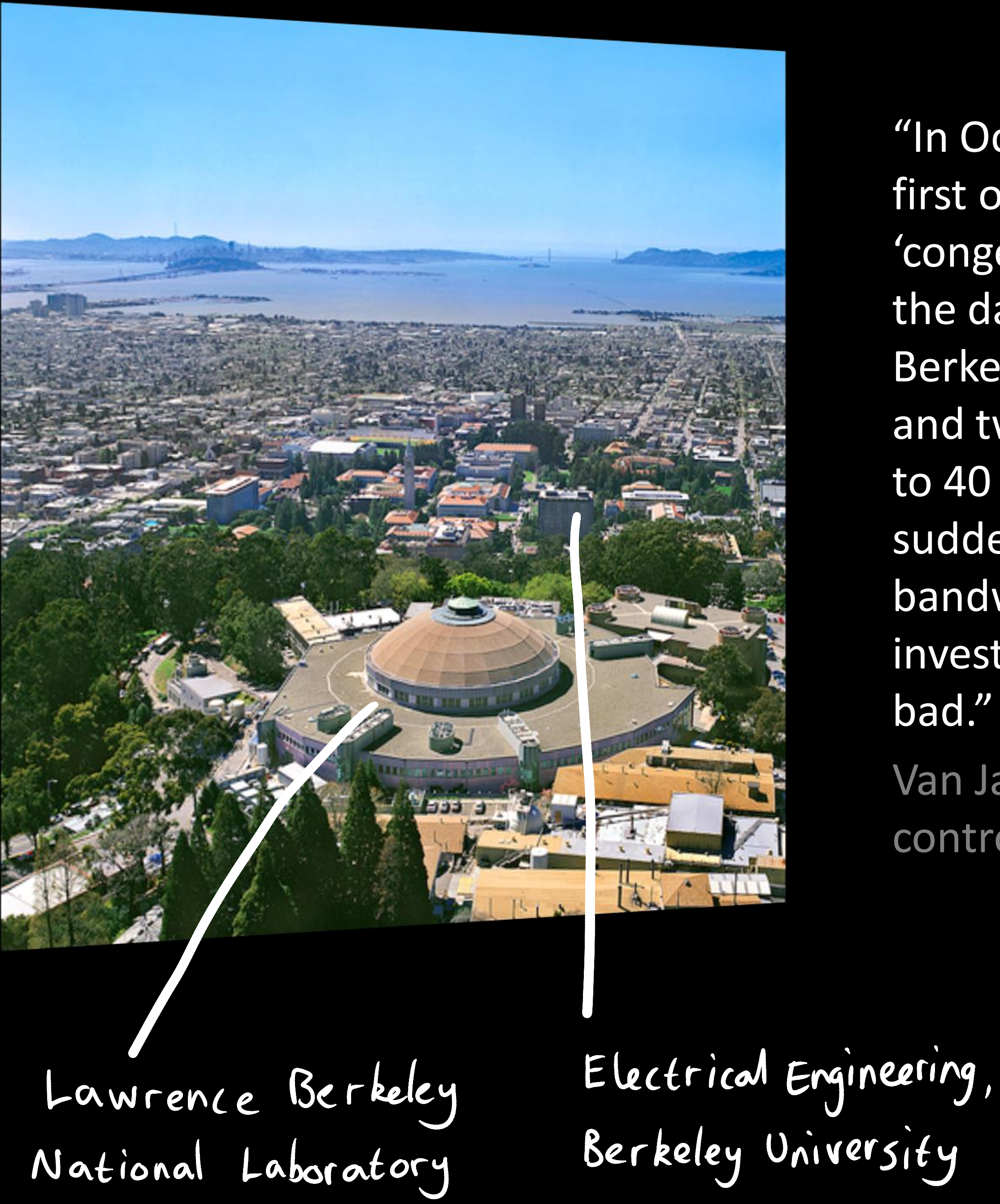
The doctrine or study of ends or final causes, especially as related to the evidences of design or purpose in nature; also *transf.* such design as exhibited in natural objects or phenomena.

The history of the Internet

- 1974: First draft of TCP/IP
  "A protocol for packet network interconnection",
  Vint Cerf and Robert Kahn

- 1983: ARPANET switches on TCP/IP

- 1986: Congestion collapse


Lawrence Berkeley National Laboratory
Electrical Engineering, Berkeley University

"In October of '86, the Internet had the first of what became a series of 'congestion collapses'. During this period, the data throughput from LBL to UC Berkeley (sites separated by 400 yards and two IMP hops) dropped from 32 Kbps to 40 bps. We were fascinated by this sudden factor-of-thousand drop in bandwidth and embarked on an investigation of why things had gotten so bad."

Van Jacobson, "Congestion avoidance and control", 1988

The history of the Internet

- 1974: First draft of TCP/IP
  "A protocol for packet network interconnection", Vint Cerf and Robert Kahn

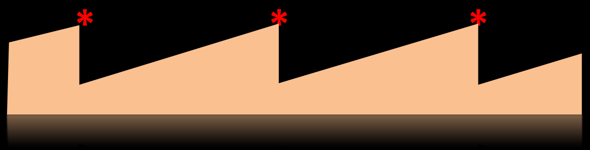
- 1983: ARPANET switches on TCP/IP

- 1986: Congestion collapse

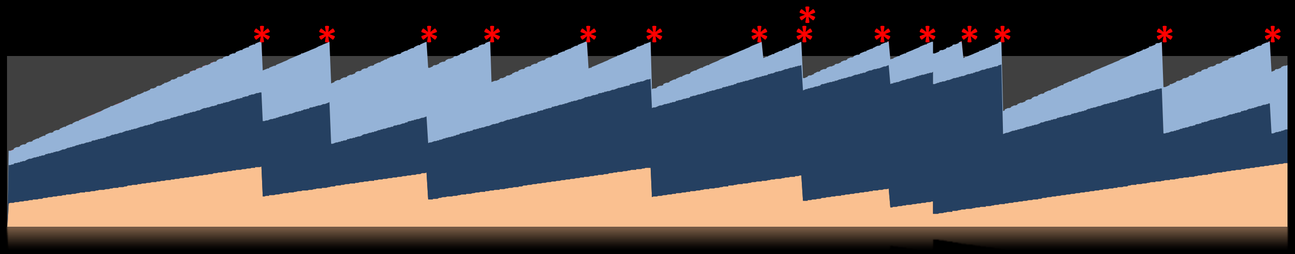
- 1988: Congestion control for TCP
  "Congestion avoidance and control", Van Jacobson

and not much has happened since (apart from the web, Google, eBay, Facebook, BitTorrent, …)

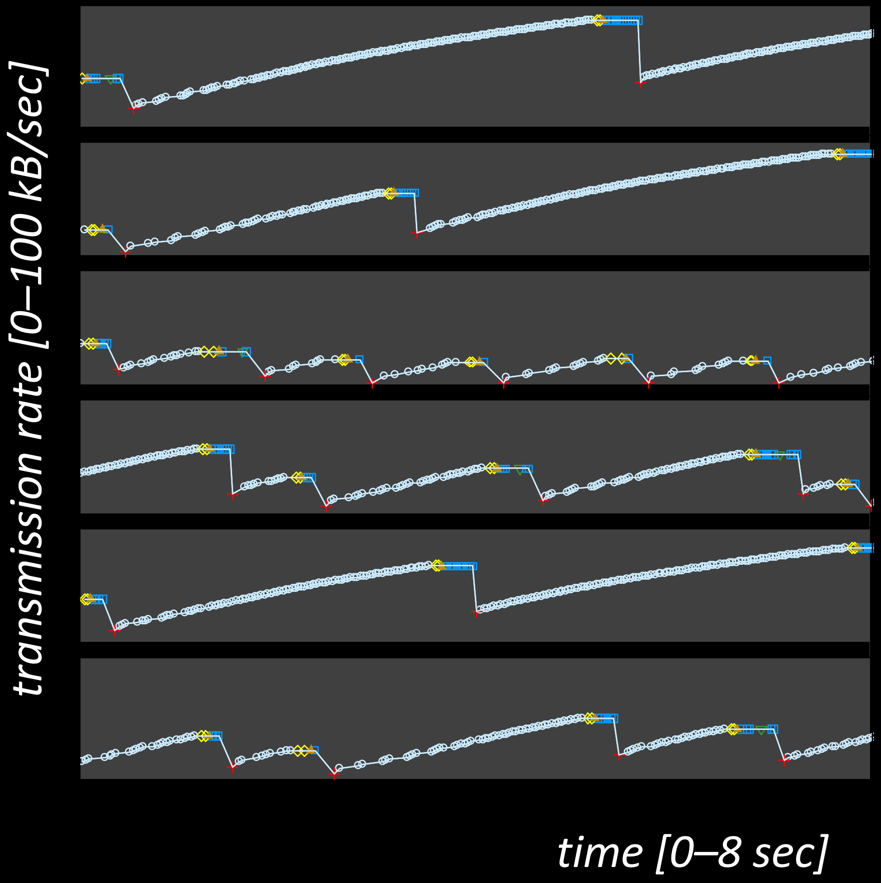
Jacobson’s big idea



Each user should increase his/her transmission rate when the network seems underused, and cut it when one of his/her packets is dropped (which signifies congestion).



If all users do this, the network ends up near-100% used, and the capacity is shared fairly.



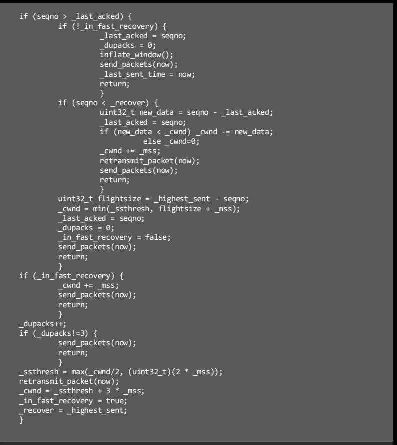
```
if (seqno > _last_acked) {
   if (!_in_fast_recovery) {
      _last_acked = seqno;
      _dupacks = 0;
      inflate_window();
      send_packets(now);
      _last_sent_time = now;
      return;
      }
   if (seqno < _recover) {
      uint32_t new_data = seqno - _last_acked;
      _last_acked = seqno;
      if (new_data < _cwnd) _cwnd -= new_data;
          else _cwnd=0;
      _cwnd += _mss;
      retransmit_packet(now);
      send_packets(now);
      return;
      }
   uint32_t flightsize = _highest_sent - seqno;
   _cwnd = min(_ssthresh, flightsize + _mss);
   _last_acked = seqno;
   _dupacks = 0;
   _in_fast_recovery = false;
   send_packets(now);
   return;
   }
if (_in_fast_recovery) {
   _cwnd += _mss;
   send_packets(now);
   return;
   }
_dupacks++;
if (_dupacks!=3) {
   send_packets(now);
   return;
   }
_ssthresh = max(_cwnd/2, (uint32_t)(2 * _mss));
retransmit_packet(now);
_cwnd = _ssthresh + 3 * _mss;
_in_fast_recovery = true;
_recover = _highest_sent;
}
```

Jacobson’s big idea was that congestion could be controlled by relying on users to respond sensibly.

The big telecoms companies did not believe him. (They still don't want to.)

Why should they?

We know the **microscopic** rules of behaviour of the Internet, i.e. the code.

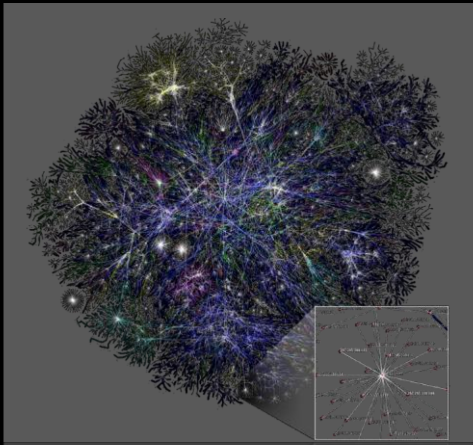


```
if (seqno > _last_acked) {
        if (!_in_fast_recovery) {
                _last_acked = seqno;
                _dupacks = 0;
                inflate_window();
                send_packets(now);
                _last_sent_time = now;
                return;
                }
        if (seqno < _recover) {
                uint32_t new_data = seqno - _last_acked;
                _last_acked = seqno;
                if (new_data < _cwnd) _cwnd -= new_data;
                        else _cwnd=0;
                _cwnd += _mss;
                retransmit_packet(now);
                send_packets(now);
                return;
                }
        uint32_t flightsize = _highest_sent - seqno;
        _cwnd = min(_ssthresh, flightsize + _mss);
        _last_acked = seqno;
        _dupacks = 0;
        _in_fast_recovery = false;
        send_packets(now);
        return;
        }
if (_in_fast_recovery) {
        _cwnd += _mss;
        send_packets(now);
        return;
        }
_dupacks++;
if (_dupacks!=3) {
        send_packets(now);
        return;
        }
_ssthresh = max(_cwnd/2, (uint32_t)(2 * _mss));
retransmit_packet(now);
_cwnd = _ssthresh + 3 * _mss;
_in_fast_recovery = true;
_recover = _highest_sent;
}
```

We can derive **macroscopic** formulae which tell us about the average behaviour of each component.

$$\frac{d}{dt}x_r(t) = \kappa\Big($$

But how does the whole behave?



Theorem

(Kelly et al. 1998, Towsley et al. 2000)

The Internet shares capacity *as if* there were an intelligent designer+controller who seeks to maximize the sum total of every user’s happiness with his/her lot, subject to available capacity on each link.

The Internet's algorithms behave as if the network as a whole were trying to solve an optimization problem.

I call this emergent teleology.

Theorem

(Kelly et al. 1998, Towsley et al. 2000)

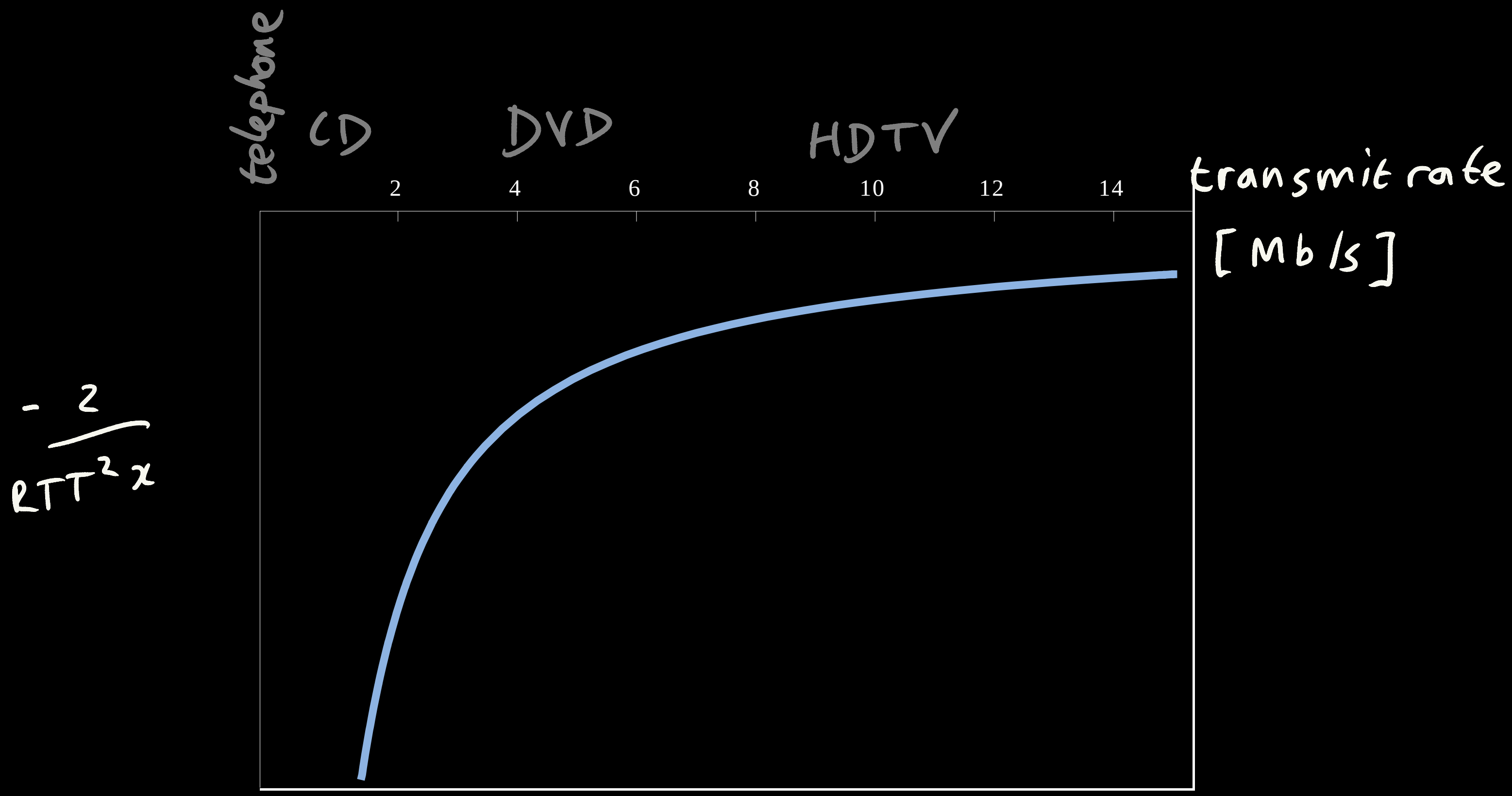
Jacobson's algorithm results in transmission rates x_r for every user r, which solve this optimization problem:

$$\text{maximize} \sum_r \frac{-2}{RTT_r^2 x_r} \quad \text{over } \underline{x} \geq 0$$

$$\text{such that} \sum_{r : r \text{ uses } j} x_r \leq C_j \quad \text{for all links } j$$

where r indexes users, x_r = throughput of user r, RTT_r = round trip time of user r, C_j = capacity of link j

Your implied utility function

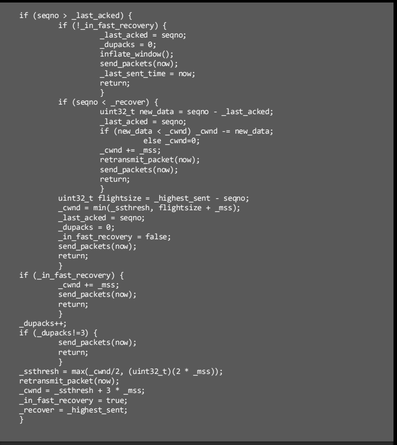


Research agenda

Microscopic rules of behaviour, specified by the code

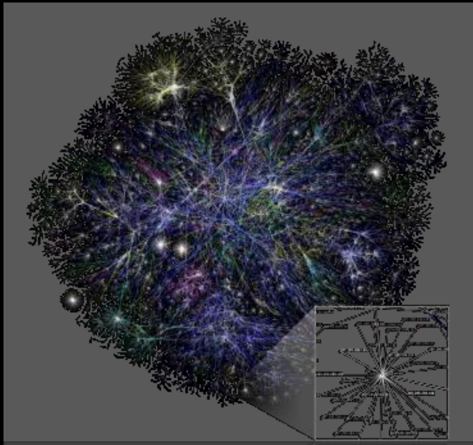
Macroscopic formulae for average behaviour of a component

Teleological descriptions of how the whole behaves



```
if (seqno > _last_acked) {
        if (!_in_fast_recovery) {
                _last_acked = seqno;
                _dupacks = 0;
                inflate_window();
                send_packets(now);
                _last_sent_time = now;
                return;
                }
        if (seqno < _recover) {
                uint32_t new_data = seqno - _last_acked;
                _last_acked = seqno;
                if (new_data < _cwnd) _cwnd -= new_data;
                        else _cwnd=0;
                _cwnd += _mss;
                retransmit_packet(now);
                send_packets(now);
                return;
                }
        uint32_t flightsize = _highest_sent - seqno;
        _cwnd = min(_ssthresh, flightsize + _mss);
        _last_acked = seqno;
        _dupacks = 0;
        _in_fast_recovery = false;
        send_packets(now);
        return;
        }
if (_in_fast_recovery) {
        _cwnd += _mss;
        send_packets(now);
        return;
        }
_dupacks++;
if (_dupacks!=3) {
        send_packets(now);
        return;
        }
_ssthresh = max(_cwnd/2, (uint32_t)(2 * _mss));
retransmit_packet(now);
_cwnd = _ssthresh + 3 * _mss;
_in_fast_recovery = true;
_recover = _highest_sent;
}
```

$$\frac{d}{dt}x_r(t) = \kappa\Bigg($$

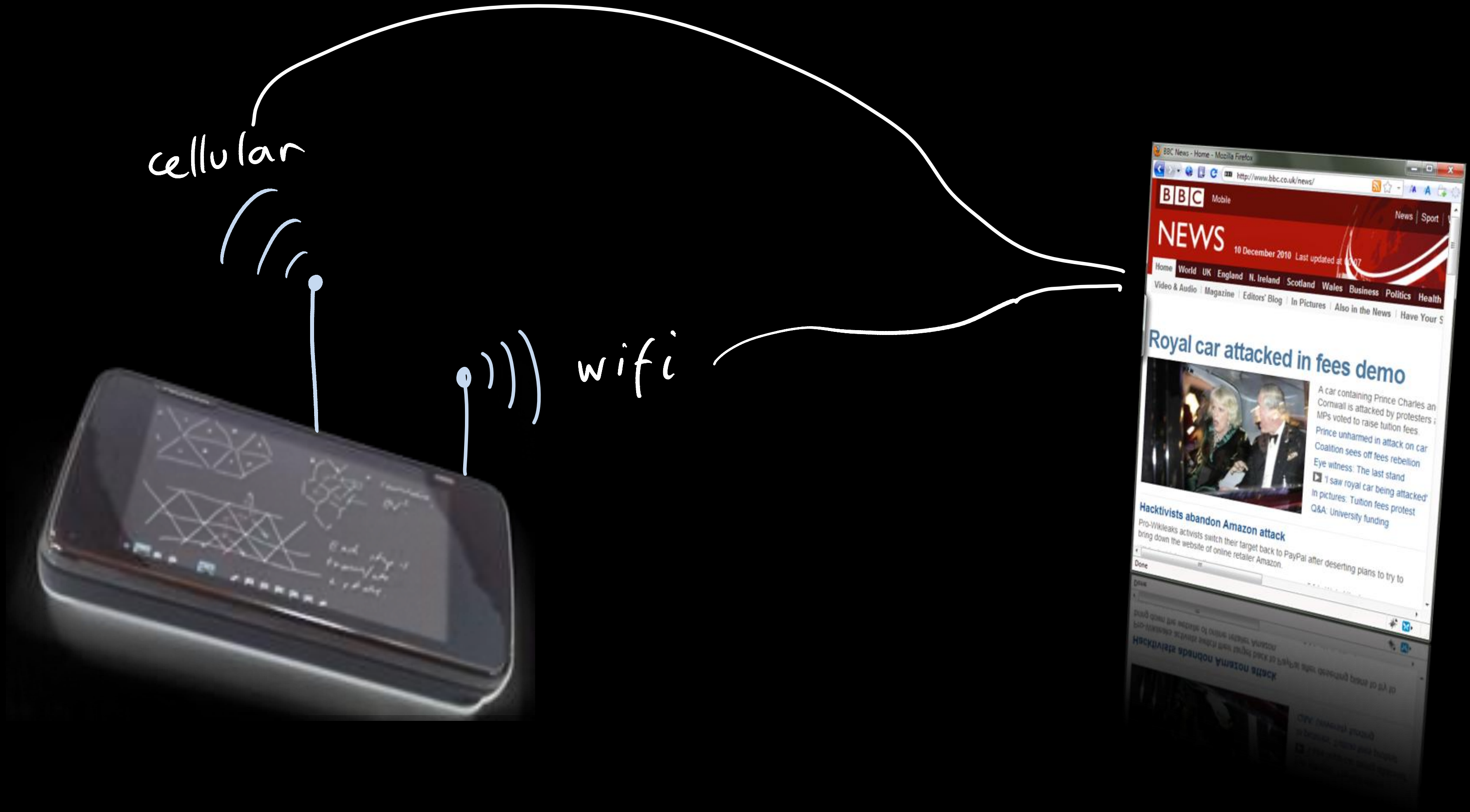


form + function
topology + algorithms

There has been much work on the structure of complex networks (scale free topologies etc.)

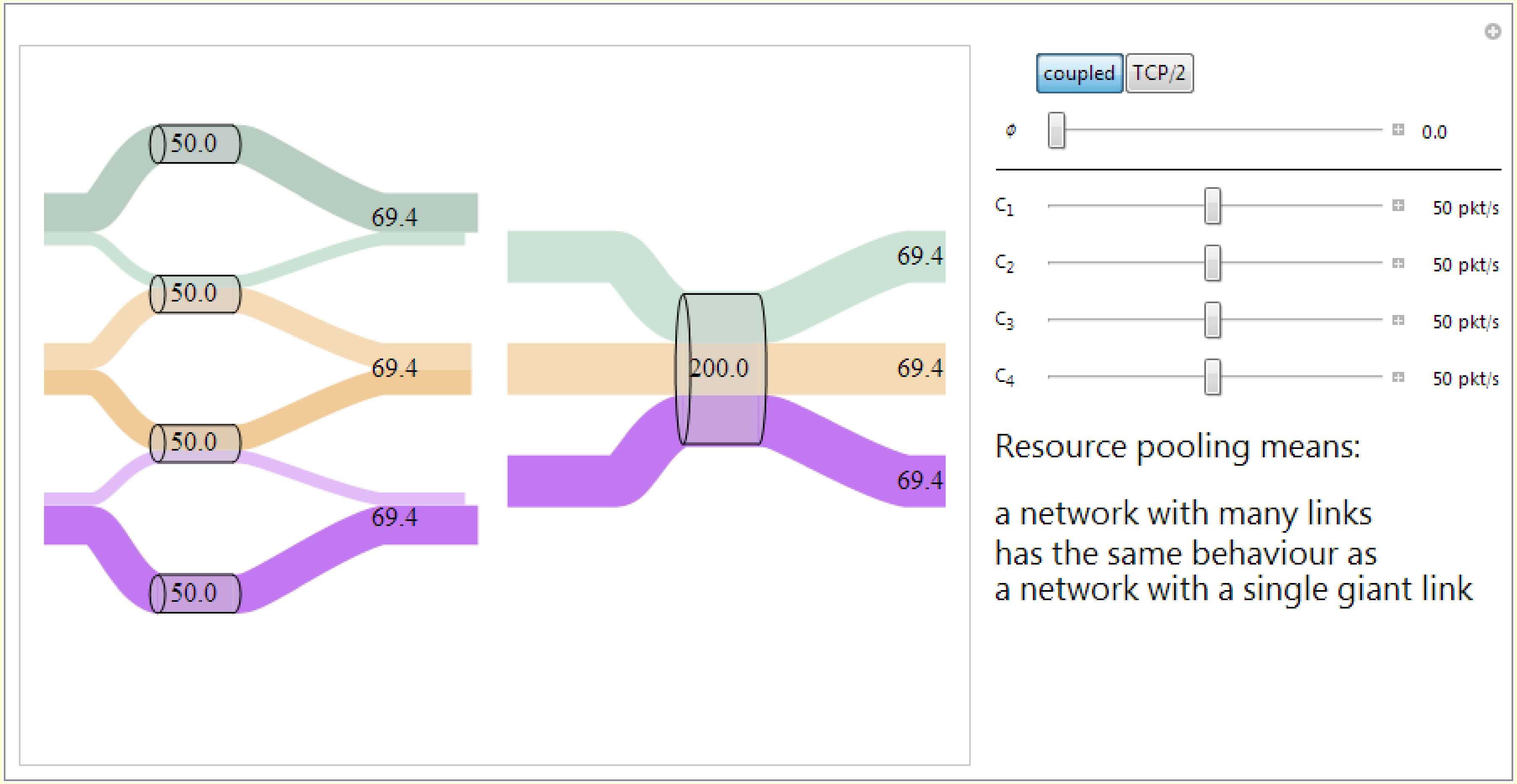
In Internet architecture, we are more interested in how algorithms function over the network. We treat the topology as a given, and we seek robust algorithms that work well with any topology or traffic pattern.

There is one area where the two fields overlap…


cellular
wifi
BBC NEWS
Royal car attacked in fees demo
Hacktivists abandon Amazon attack

Where topology and algorithms overlap:

We conjecture that if users have sufficiently diverse paths, and they balance their traffic appropriately, then the Internet will achieve resource pooling, i.e. it will behave as if there were a single giant link, fairly shared between all users.



ϕ 0.0

C_1 50 pkt/s

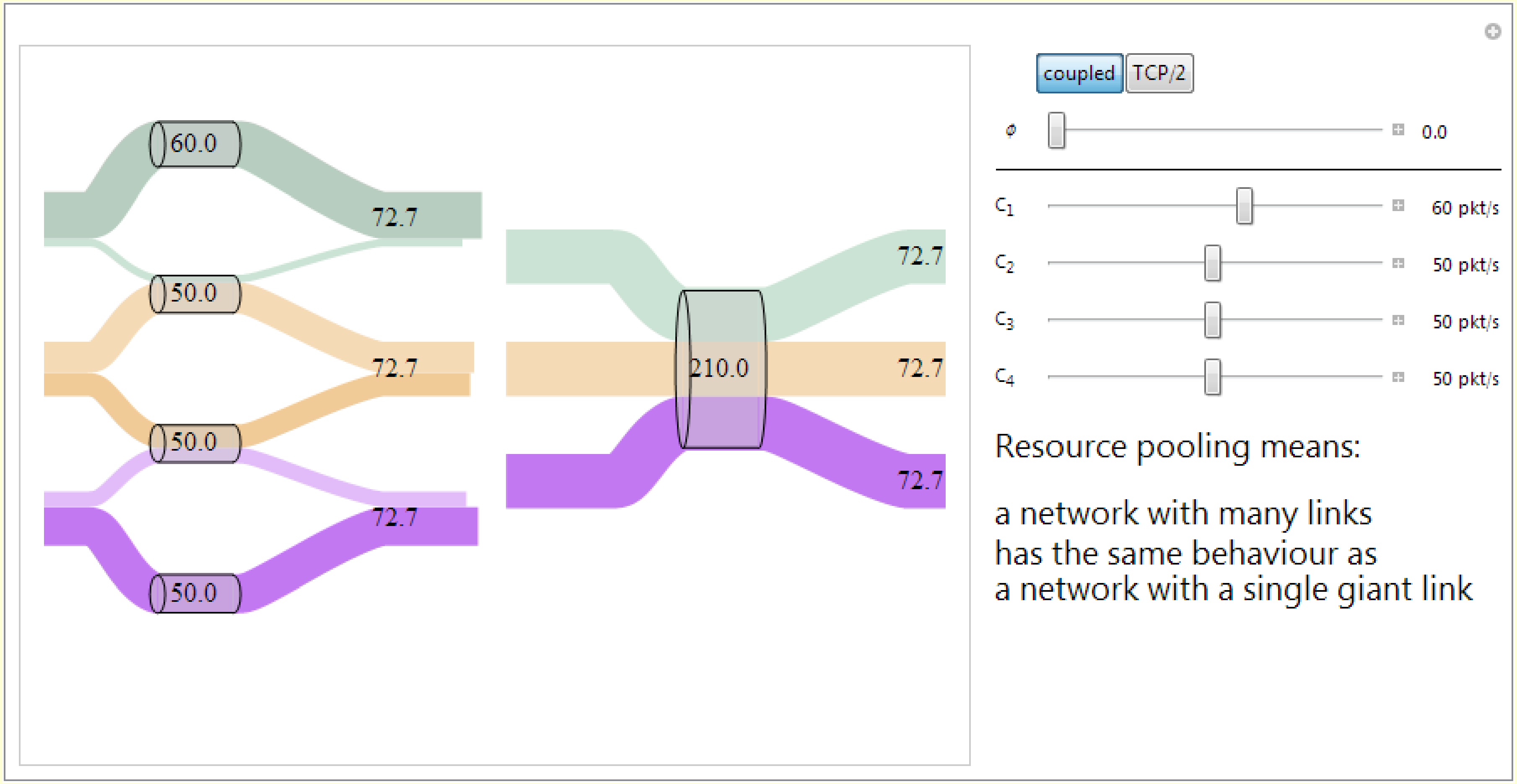
C_2 50 pkt/s

C_3 50 pkt/s

C_4 50 pkt/s

Resource pooling means:

a network with many links
has the same behaviour as
a network with a single giant link



C_1 60 pkt/s

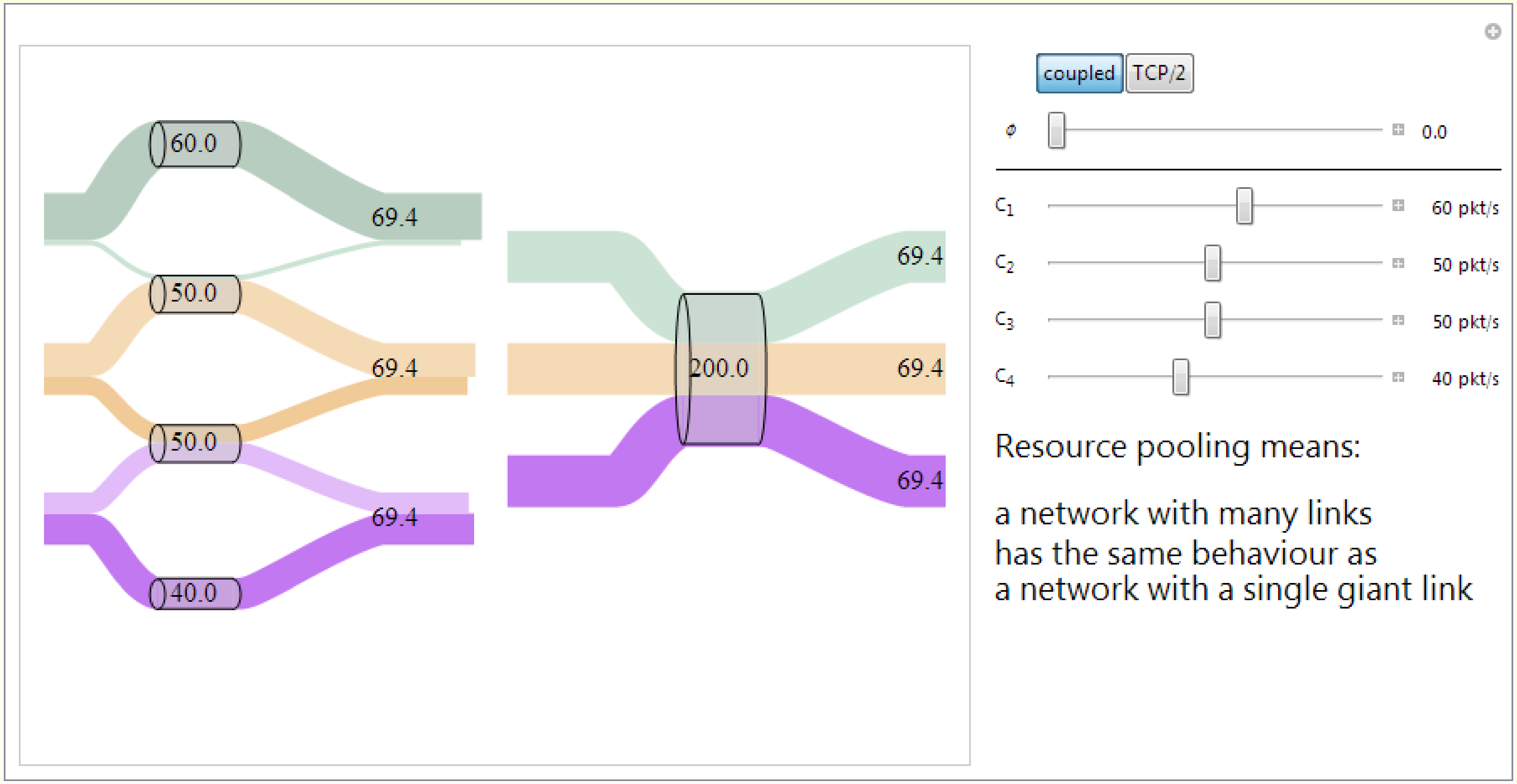
C_2 50 pkt/s

C_3 50 pkt/s

C_4 50 pkt/s

Resource pooling means:

a network with many links
has the same behaviour as
a network with a single giant link



ϕ 0.0

C_1 60 pkt/s

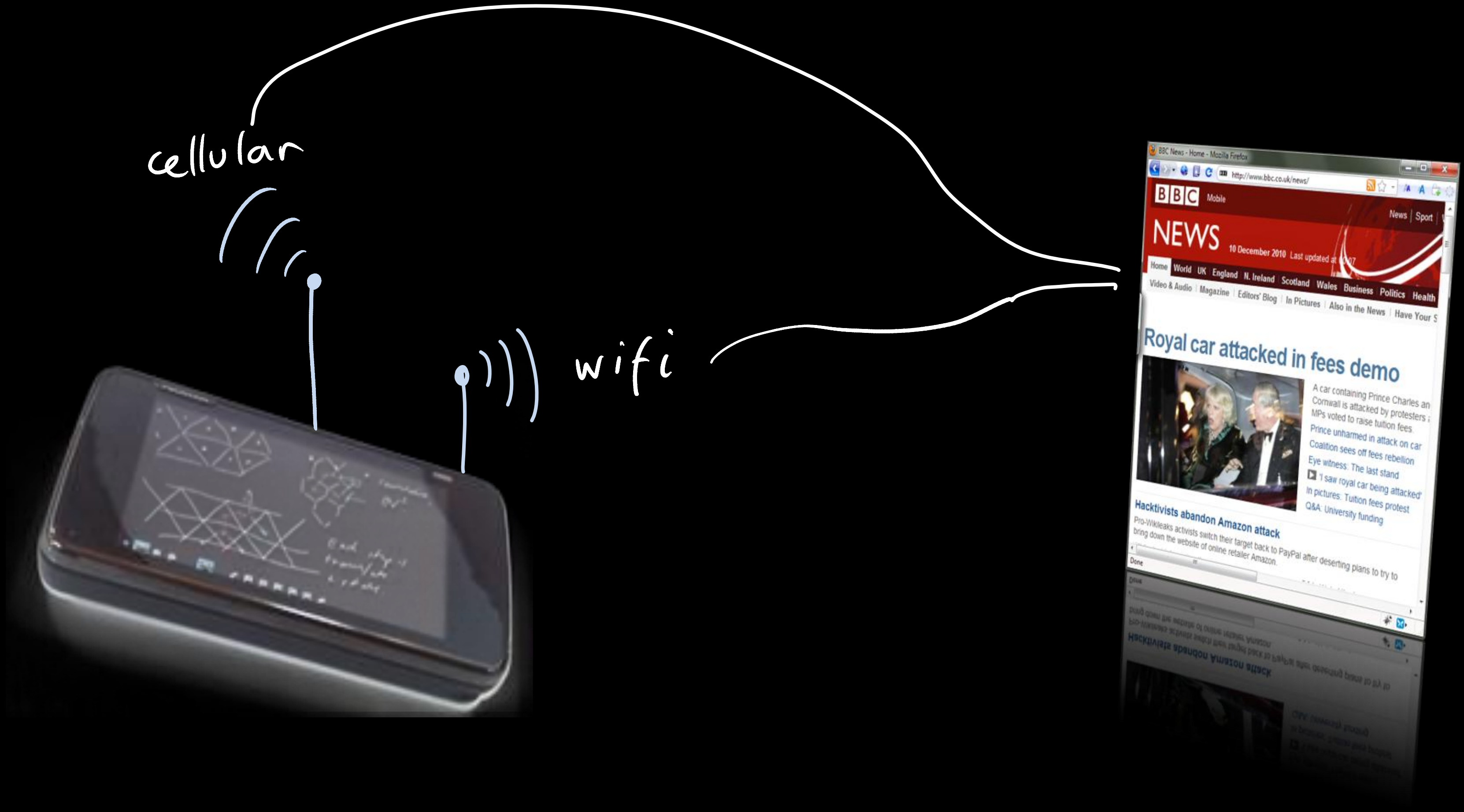
C_2 50 pkt/s

C_3 50 pkt/s

C_4 40 pkt/s

Resource pooling means:

a network with many links
has the same behaviour as
a network with a single giant link



BBC News - Home - Mozilla Firefox

http://www.bbc.co.uk/news/

BBC Mobile

NEWS 10 December 2010 Last updated at 16:07

Home | World | UK | England | N. Ireland | Scotland | Wales | Business | Politics | Health

Video & Audio | Magazine | Editors' Blog | In Pictures | Also in the News | Have Your S

Royal car attacked in fees demo

A car containing Prince Charles an Cornwall is attacked by protesters a MPs voted to raise tuition fees

Prince unharmed in attack on car

Coalition sees off fees rebellion

Eye witness: The last stand

'I saw royal car being attacked'

In pictures: Tuition fees protest

Q&A: University funding

Hacktivists abandon Amazon attack

Pro-Wikileaks activists switch their target back to PayPal after deserting plans to try to bring down the website of online retailer Amazon.

"I want to book a train to Aberdeen. There's spare capacity on the train to Bristol. Therefore my booking will be accepted."

Formally speaking, we believe that multipath congestion control will change the constraint in the teleology.

$$\max \sum_{\text{all users } r} \text{user's happiness with his/her rate } x_r \quad \text{over} \quad \underline{x} \geq 0$$

such that

~~total traffic on any link~~ $\leq$ ~~that link's capacity~~

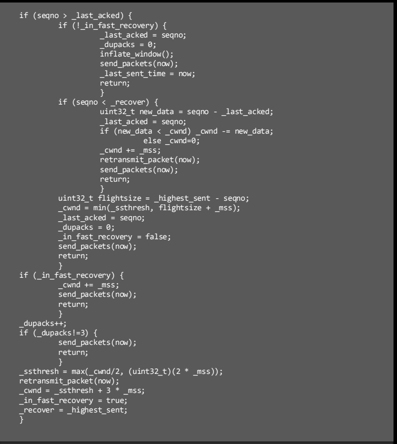
total traffic in the entire network $\leq$ total capacity in the entire network

Engineering agenda

Invent **microscopic** code that yields this macroscopic behaviour

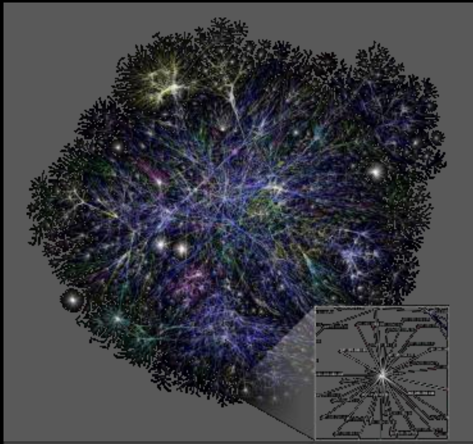
Work out **macroscopic** formulae for the average behaviour of a component

Decide on the **teleology** that we want



```
if (seqno > _last_acked) {
        if (!_in_fast_recovery) {
                _last_acked = seqno;
                _dupacks = 0;
                inflate_window();
                send_packets(now);
                _last_sent_time = now;
                return;
                }
        if (seqno < _recover) {
                uint32_t new_data = seqno - _last_acked;
                _last_acked = seqno;
                if (new_data < _cwnd) _cwnd -= new_data;
                          else _cwnd=0;
                _cwnd += _mss;
                retransmit_packet(now);
                send_packets(now);
                return;
                }
        uint32_t flightsize = _highest_sent - seqno;
        _cwnd = min(_ssthresh, flightsize + _mss);
        _last_acked = seqno;
        _dupacks = 0;
        _in_fast_recovery = false;
        send_packets(now);
        return;
        }
if (_in_fast_recovery) {
        _cwnd += _mss;
        send_packets(now);
        return;
        }
_dupacks++;
if (_dupacks!=3) {
        send_packets(now);
        return;
        }
_ssthresh = max(_cwnd/2, (uint32_t)(2 * _mss));
retransmit_packet(now);
_cwnd = _ssthresh + 3 * _mss;
_in_fast_recovery = true;
_recover = _highest_sent;
}
```

$$\frac{d}{dt}x_r(t) = \kappa\Big($$



(We don't want network topology to be a constraint, only total network capacity.)