COMPSCI 742 - 2014
Internet Quality of Service

Nevil Brownlee

Slides prepared by Brian Carpenter

- Topic 1: What is QoS?
  - End to end service
  - Service Level Agreements
  - Network Neutrality
- Topic 2: QoS technology
- Topic 3: Wireless QoS

References

- Internet Performance Survival Guide, by Geoff Huston (2000)
- Peterson & Davie section 6.5
- Internet QoS: Architectures and Mechanisms for Quality of Service, by Zheng Wang (2001)
- RFC 6057
- RFCs linked at:

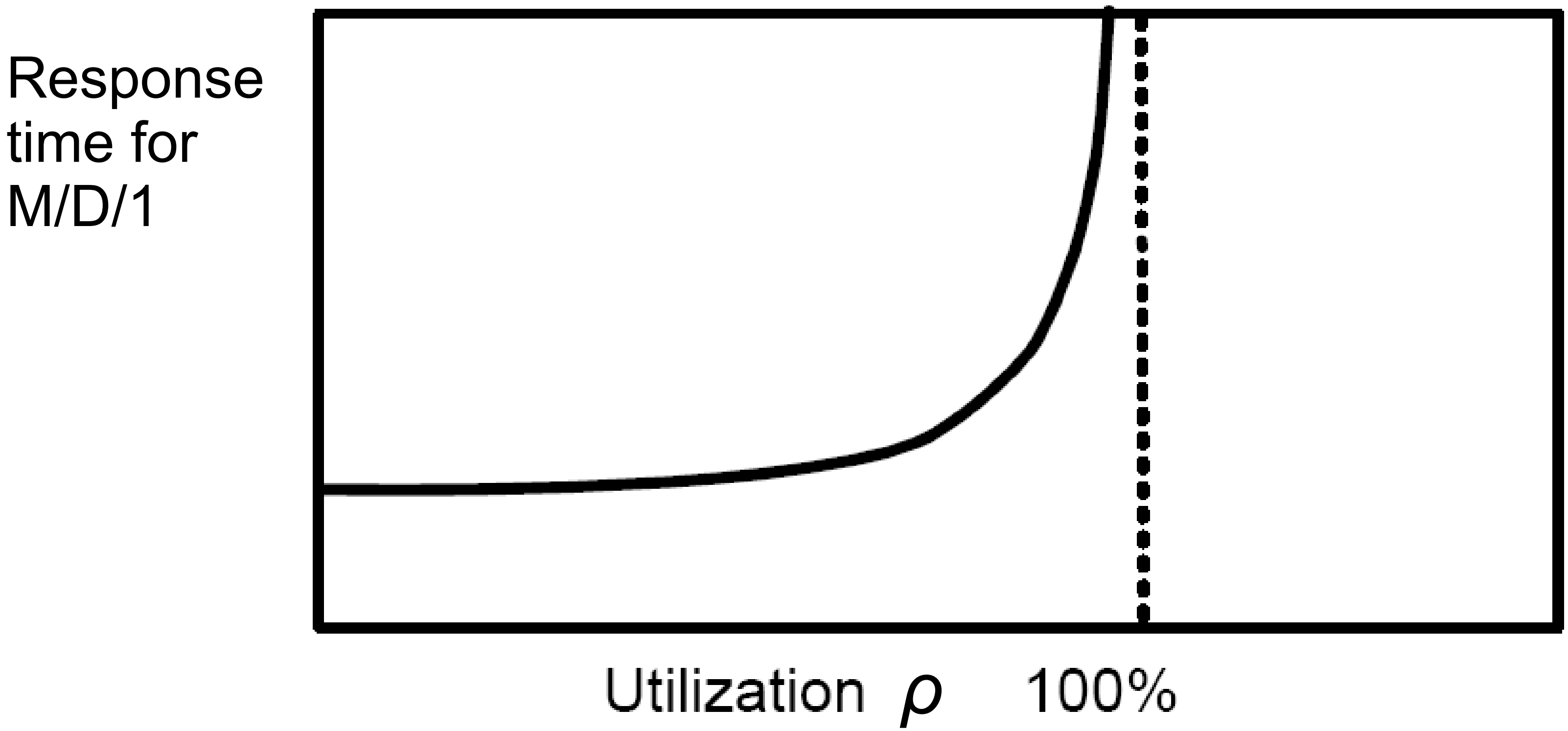
http://www.ietf.org/wg/concluded/pilc.html

- Some research papers on TCP over wireless

http://www.ee.unimelb.edu.au/atnac2006/papers/27.pdf

http://www.icir.org/padhye/canton_chahed.pdf

Topic 1. What is QoS?



$$W = \frac{\rho}{1-\rho}\frac{1}{\lambda}$$

Response time
for M/M/1

If it was this simple, all we would do is keep the utilization low, and there would be nothing to discuss.

In real life...

- diggers cut cables*
- electronics goes bad
- software has bugs
- everyone clicks at the same time
  - popular new web sites have been known to underestimate load by a factor 100
- network operators make mistakes
- network engineers under-estimate growth

* http://blog.level3.com/2011/08/04/the-10-most-bizarre-and-annoying-causes-of-fiber-cuts/

Impact of diggers

- When a cable gets cut, IP routing should switch automatically to another path
  - but not until routing timeouts expire and routing algorithms compute new routes
  - not if the alternate path happens to go through the same cable
  - not if the customer was too mean to pay for backup
  - not if the backup procedures (which are hard to test) themselves fail

Virtual diggers

- Router, switch or other electronics failures can take out one or more links
- Routing software glitches can divert or lose traffic, and can do so repeatedly without being fixed, due to very complex diagnosis
- Routing database failures, or DNS database failures, can chop off pieces of the network
- Operators can misconfigure anything, with the same effects

Load surges

- Massive surges in load can overwhelm network capacity, leading to instant congestive collapse in some part of the network.
- This is simply another way of describing self-similar traffic patterns.
- Not talking about 10 or 20% overload as for electricity brown-outs, but 100 or 1000%.
- *Nobody* installs that much spare capacity.

So, what is Service Quality?

- Users want the “*bilities*”...
  - reliability, stability - no glitches
  - availability – 365 x 24
  - predictability - "the same rotten service at 3 a.m. as at 3 p.m." regardless of other users
  - applicability - supports the user's applications
  - survivability - service must survive diggers and virtual diggers
  - scaleability - more capacity there when needed
  - responsibility - a help desk that helps

And what is Quality of Service?

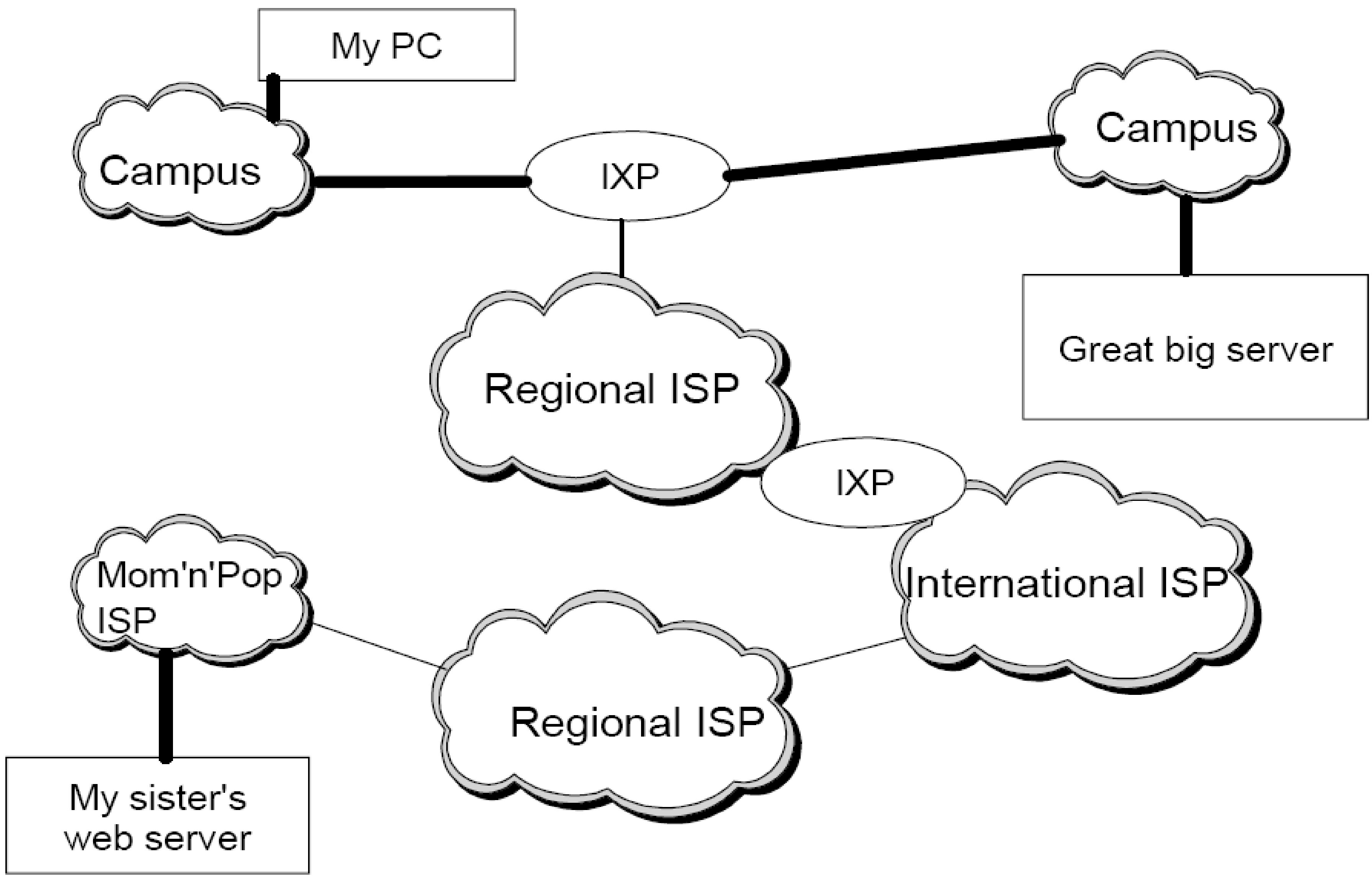
- Usually quite narrowly defined to mean a specific, measurable set of characteristics of a data stream, such as
  - transit delay
  - jitter (variable gaps between packets, IP delay variation)
  - loss rate
  - throughput

needed by a specific application or user

Traditional “Best effort” Internet

- No formal agreement and simple flat-rate charging
- This means “***I'll send your traffic on towards its destination, unless my routers are too congested, or the routing tables are messed up, or whatever, and in that case I'll probably just discard it***”
- Even if most packets get through, this just isn't good enough any more for the commercial Internet of the 21st century, or for real-time audio and video

The best effort environment



Service Level Agreements (SLAs) replace "best effort"

- Internet service provision is a competitive business. Many customers now require some assurance of service quality, which means they will require their service provider to sign up to a contract, usually called a Service Level Agreement (SLA)
  - An SLA lays down specific details for the "bilities"
  - An SLA sets clear expectations and responsibilities

Who signs an SLA? (1)

- The world is split into users and providers.
- Users include
  - “real” users (domestic subscribers, professors of archaeology, lawyers, musicians,...)
  - operators of web sites and other servers
  - campus network managers
  - local ISP managers
- Notice that some users are also providers

Who signs an SLA? (2)

- Providers include
  - campus or company networks
  - local, regional and national ISPs
  - Internet Exchange Points (IXPs)
  - international backbone ISPs
- ISPs “peer” (i.e. exchange traffic with each other on a peer-to-peer basis)
  - at IXPs
  - directly, via bilateral agreements

Typical contents of an SLA (1)

- An SLA regulates the arrangements between a user and its provider
- Expected performance level
  - reliability and availability - what's the % up time, what are the limits of scheduled down time?
  - performance metrics - average and maximum response time, average and maximum throughput, variance of response time and throughput, etc

Typical contents of an SLA (2)

- Problem reporting process
  - Where to report problems
    - By email or web site? What if the network's down?
  - Format for problem reports (typically including severity levels)
- Maximum time for provider to start responding
  - Daytime? Nighttime? Weekend/holiday?
- Time frame for problem resolution

Typical contents of an SLA (3)

- Monitoring and reporting
  - What statistics are collected
  - How statistics are accessed and reported to customer
- Price of service
  - flat rate or charged by traffic volume
  - penalties for missing SLA targets
- Legal stuff

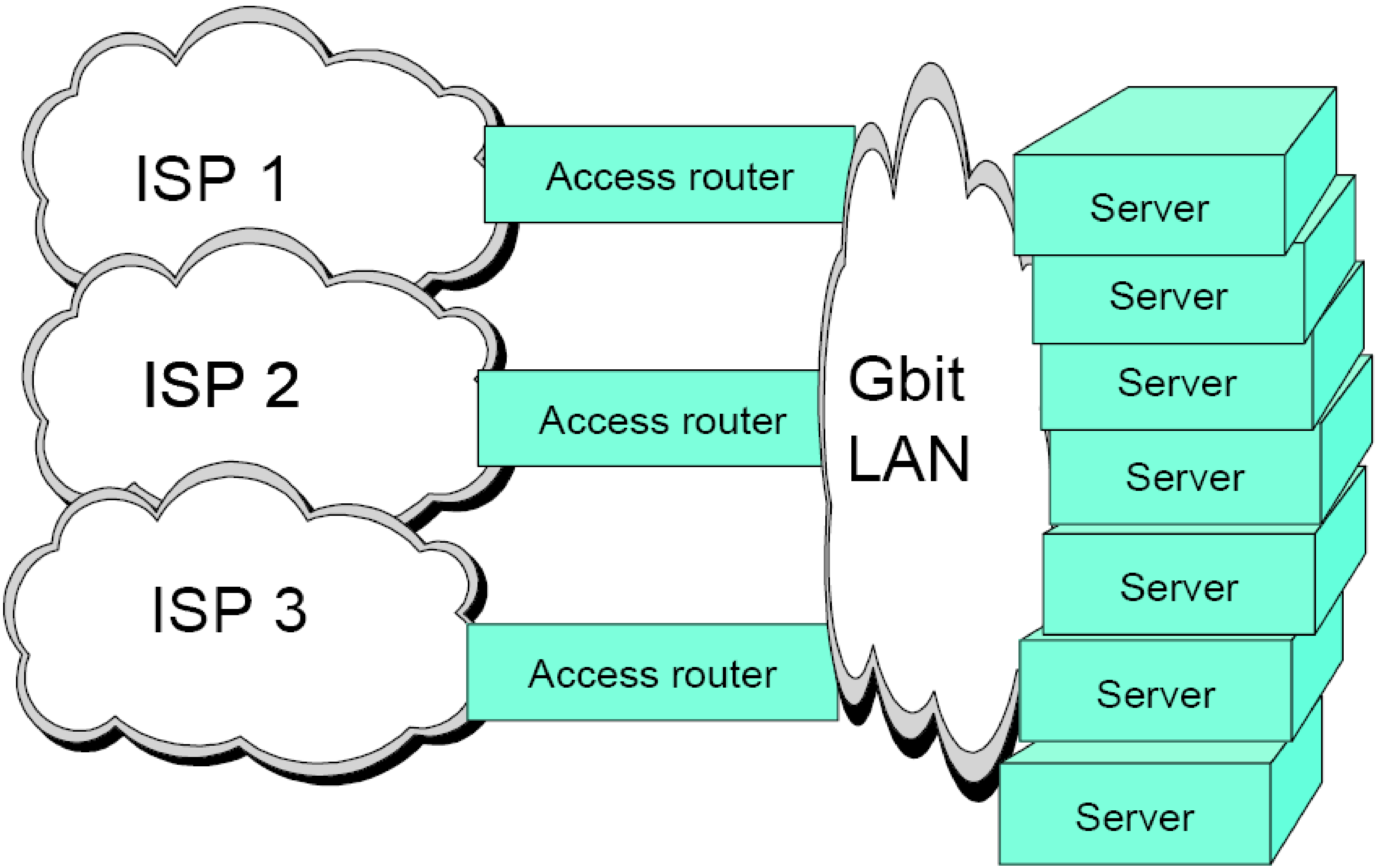
Reasonable and unreasonable promises in a best effort Internet

- ISP A can say to its direct customers
  - I guarantee you *X* QoS between you and any other customer who has the same deal
  - I guarantee to deliver your packets with *X* QoS to any egress from my network
    - (as long as you send less than *Y* traffic)
- ISP A *cannot* honestly say
  - I guarantee you *X* QoS between you and a customer of ISP B

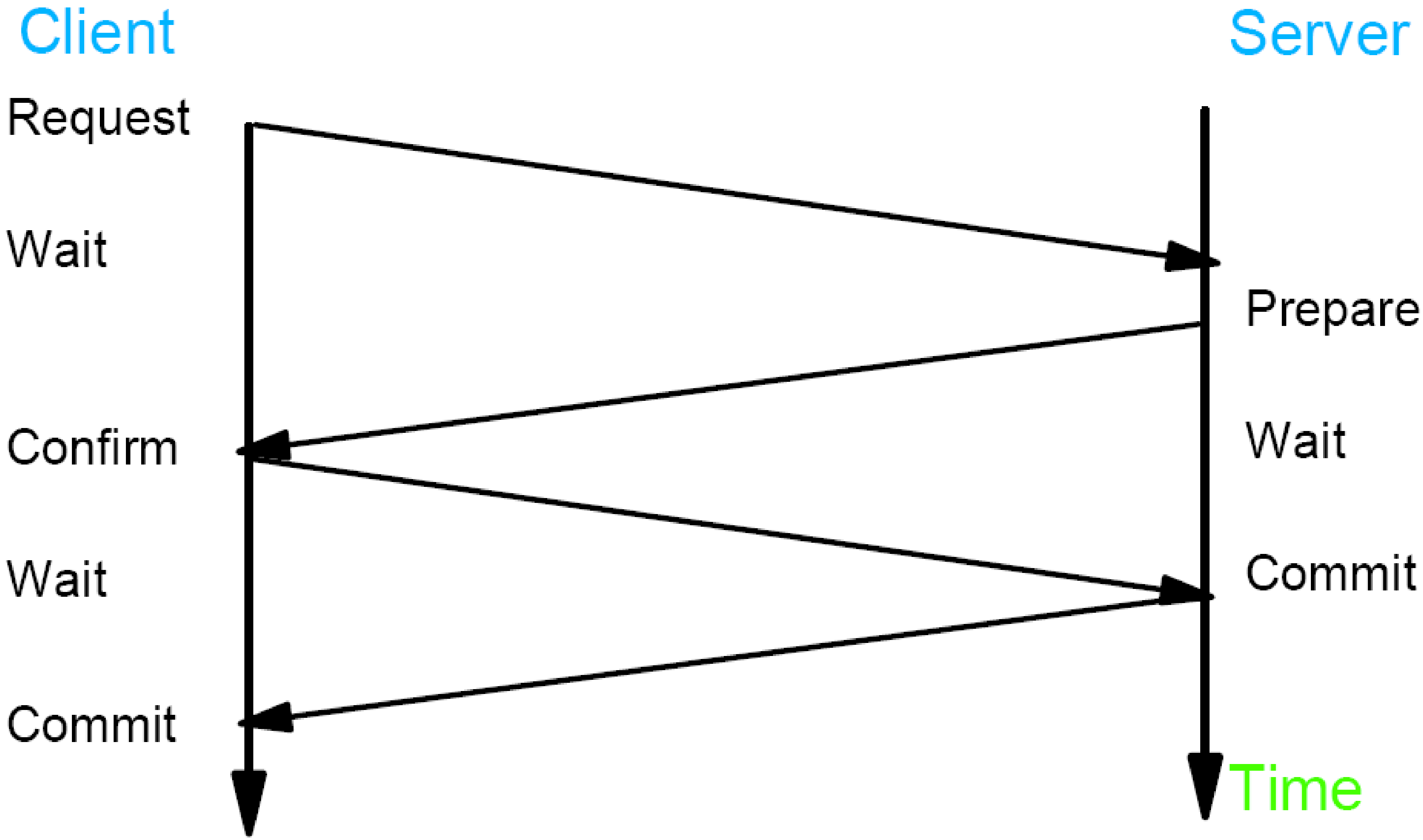
unless A and B have an SLA in place

Hosting application services

SLA must cover all aspects...



Components of response time for an application SLA



What determines the average delay

- There is a combination of server and network delays to be included in the calculation of the actual average delay
- Transactions compete for multiple resources at the server and each message competes for network transmission
- Boiling this down to “average transaction delay will be X milliseconds and 99% of transactions will complete in less than Y milliseconds” is a specialised skill

SLA Calculations

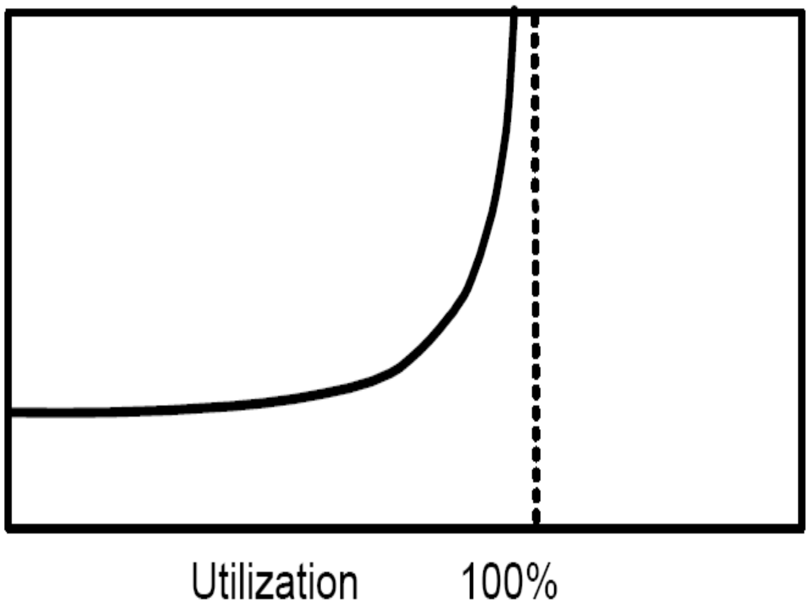
- To satisfy specific average delays for a mixture of transactions, both servers and network must deliver adequate performance
  - Servers must support the required transaction rate, say 100 transactions per second
  - Assume 10x parallelism giving 100 ms/transaction
  - If the required average transaction delay is (say) 1 second, does that mean the required average network delay is 900 ms?

Adding up the delays

- “Commit” transaction:
  - total delay = request + prepare + acknowledge + confirm + commit + acknowledge, e.g. 100 ms plus 4 network delays
  - Average network delay will need to be less than 900/4 = 225 ms

Average delay is not the whole story

- SLA will also need to specify 90th or 99th percentile delay - and there, the queuing theory curve comes in
- How high a utilisation will keep the average and 90th percentile response time low enough?
- Knowing that utilisation, how much spare server and network capacity must be paid for?



Network neutrality

- Is it OK for an ISP to give better service to its own video servers than to a video server hosted by another ISP?
- Is it OK for an ISP to detect BitTorrent or Skype traffic and block it or slow it down?
- If you think the answers are “no”, you are in favour of network neutrality
- This is a political question, but it has technical implications
  - How can you tell if an ISP is not being neutral?

Topic 2: QoS Technology

- How can QoS be described and measured?
- How do users request QOS?
- How do service providers deliver the requested QoS?
  - macroscopic scale: network capacity planning and traffic engineering, robustness and operational support
  - microscopic scale: meeting QOS needs of individual users

Network QoS metrics

- Delay (One-way, not RTT)
  - sum of queueing times plus transmission times from end to end
- Jitter (IP packet delay variation, IPDV) [RFC3393]
  - measured as the *difference* between successive one-way delays
  - measured statistically (e.g., variance of IPDV)
  - caused by the variability of queueing times
- Loss
  - caused by congestion (queue overflow) and by hardware errors
  - hardware errors normally insignificant, except on wireless links
- Throughput
  - set by fair share of capacity, TCP flow control, etc

Reminder: Bandwidth, Capacity, Throughput

- Bandwidth is the raw physics ability of a cable or wireless channel to carry electromagnetic signals
  - Measured in MegaHertz (millions of sine wave cycles per second)
- Capacity of a data link is the number of bits per second the electronics at each end of the link can successfully transmit and receive
  - Always much less than the bandwidth; measured in Mbit/s
  - Limited by coding techniques applied to electromagnetic signals
  - Often incorrectly called “bandwidth”
- Throughput
  - Usually <100% of the capacity

Metrics depend on application

- Example: Voice over IP
  - need acceptable delay (e.g. 200 ms)
  - need jitter and loss low enough that the audio codec can hide them
  - need a few kb/s throughput per session
- Example: File transfer
  - don't care about delay or jitter
  - need loss low enough that TCP doesn't keep entering slow start
  - need all the throughput you can get
- Useful throughput, discounting retransmissions and packet header overhead, is sometimes called **goodput**. It may be significantly less than raw throughput observed on the wire

UDP and TCP

- UDP traffic is not automatically reduced when there is congestion ("selfish" traffic)
  - but UDP is more suitable for real time voice and video, where we can't slow down whatever happens
  - lost packets are lost for ever, causing gaps in signal
- TCP traffic is automatically reduced by congestion ("polite" traffic)
  - lost packets are sent after a delay
  - causes a forced ...... pause in voice or .... video
- In either case, we need adequate throughout and low loss for successful voice and video

QoS granularity

- "Fine granularity" means that if you have a million phone calls in progress, there will be a million data structures supporting QoS for those calls
- How can QoS be provided for millions of individual communications sessions?
  - tradeoff between fine granularity and massive scale
  - choice between static or dynamic QoS mechanisms
  - fine granularity, dynamic mechanisms have much more overhead than coarse, static mechanisms and are harder to measure

The simplest QoS architecture

- “Just throw bandwidth at the problem”
  - i.e., engineer the network to have at least 50% spare capacity at all places and at all times
  - “best effort” service for every user and every packet
  - in such a network, any old queueing discipline will keep us down on the flat part of the response time curve, but
    - at all places means: not one single bottleneck
    - at all times means: including at the busiest second of the busiest hour of the busiest day

Why throwing bandwidth at the problem doesn't always work

- Such a network is called “over provisioned” (i.e. more capacity has been provided than will ever be needed)
- 50% excess capacity in the busiest second, given the extreme burstiness of Internet traffic, might mean having a capacity about 200 times the long term average load at every point in the network
- This makes no economic sense

Implementing “Best Effort”

- Since we are giving every packet the same service (including the same probability of being lost or delayed), all queues in all routers can be simple FIFO (first-in, first-out)
  - that makes routers simple to design
- TCP was designed assuming FIFO queues
  - but UDP streams can monopolise FIFO queues
  - hence the best effort/FIFO model cannot support specific QoS metrics

Mechanisms to consider

- Traffic engineering
- QoS-based routing
- Classification and admission control
- Traffic shaping
- QoS architectures

Traffic engineering

- Refers to the methods used by network operators to adjust traffic paths and capacities
  - BGP4 is used to control paths between ISPs
  - MPLS is widely used to control paths and capacities within a single ISP network
    - ATM was used for this previously
  - VPNs (over MPLS or otherwise) are widely used to control paths and capacities for individual business customers
  - Some devices (e.g. ADSL hubs) allow direct control of network capacity per subscriber

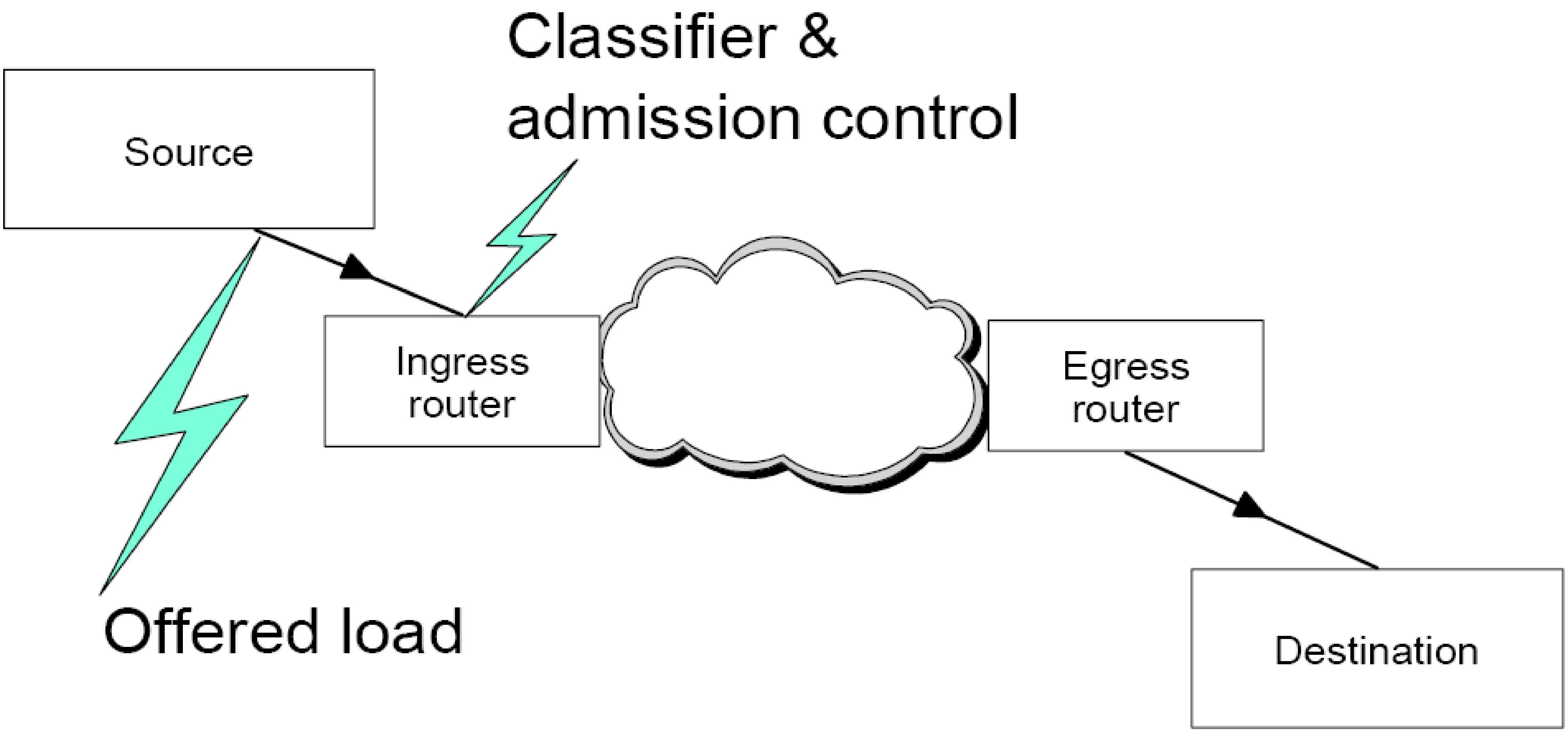
QoS-based routing

- In theory, routing protocols such as OSPF can be modified to support QoS requirements
  - QoS metrics would be considered as well as weights when running path computations
- In practice, QoS-based routing has never been seriously used

Classification, admission control

- A classifier examines each packet and assigns it to a QoS class
  - typically looks at the IP packet header 5-tuple
  - S1 ≤ source address ≤ S2
D1 ≤ dest address ≤ D2
protocol = UDP
P1 ≤ port ≤ P2 ===> Class X.
- Admission control only allows a given rate of Class X packets to enter the network

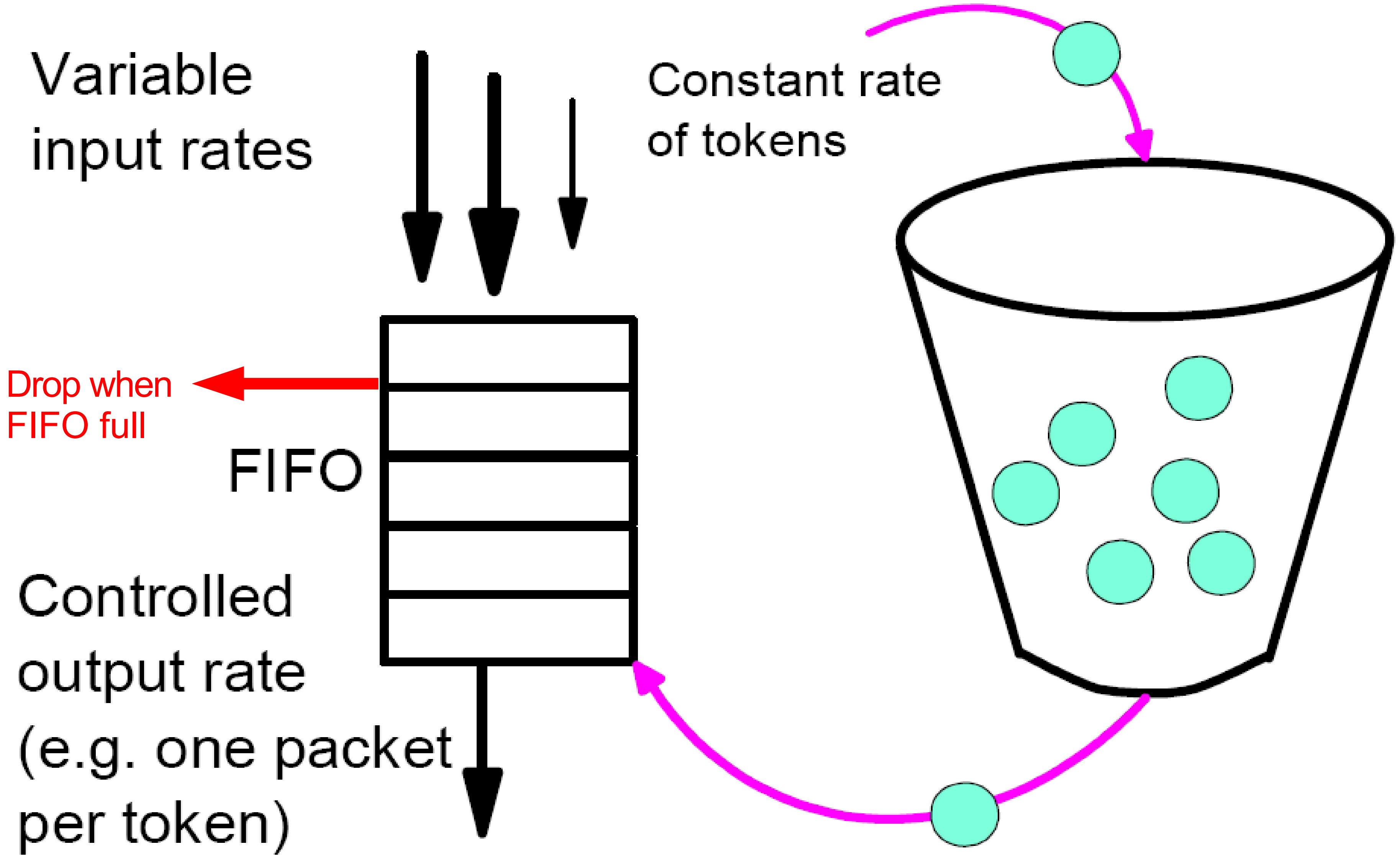
Admission control may reject part of offered load



Traffic shaping

- Making the offered load acceptable to the admission control:
  - rate controller: a device that limits the rate of packets to some given value
  - dropper: a device that drops (discards) selected packets, to limit the rate
  - scheduler: a device that schedules individual packets for onward transmission
- These are all operations on a queue of packets, typically a FIFO queue

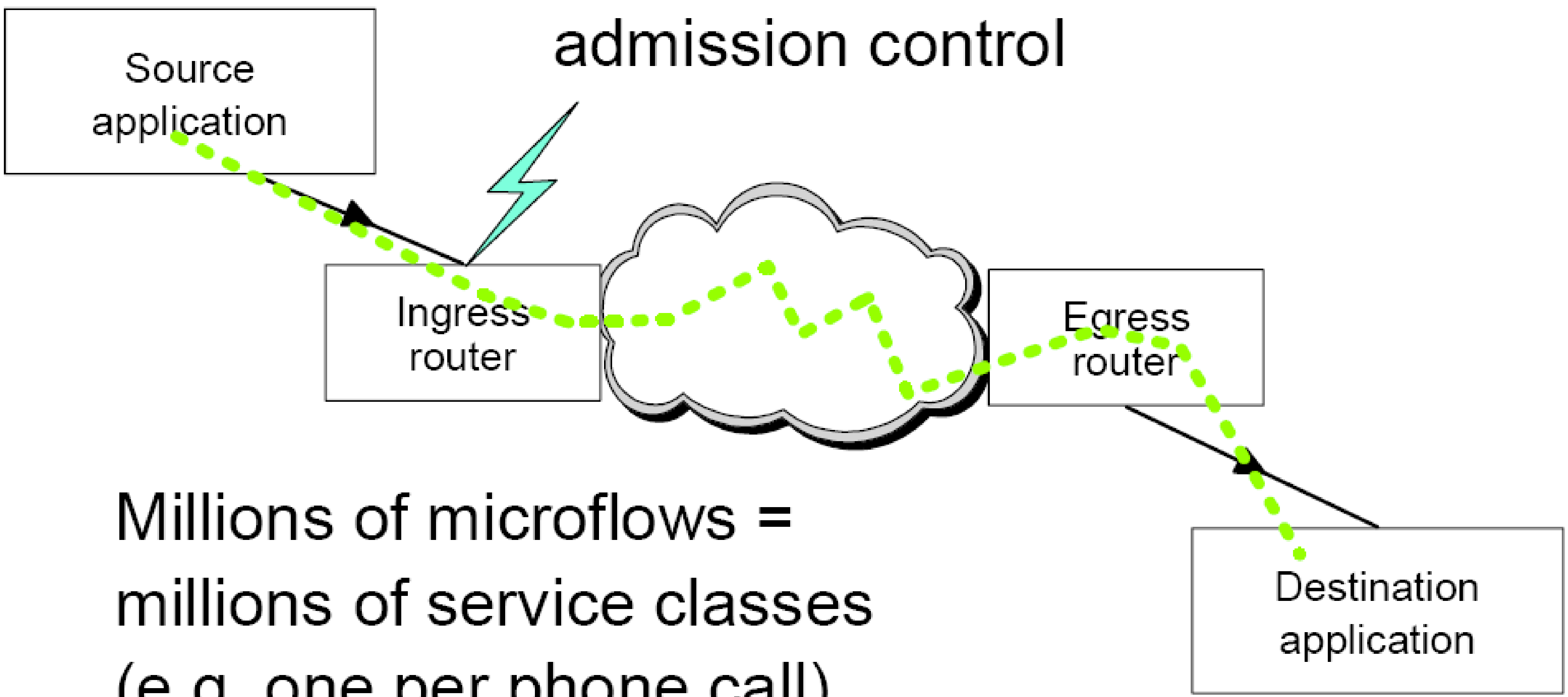
Token bucket shaper



Three QoS architectures

1. Provide each flow with the QoS it requires
   - perform admission control & shaping for each flow, and create a suitable service class
2. Inject each flow into an existing traffic class with suitable QoS
   - classify, and perhaps perform admission control & shaping for, each flow. Inject the flow into a pre-existing service class
3. Limit customers to a fair share, but otherwise use best effort techniques

Architecture 1: Service class per flow



Millions of microflows = millions of service classes (e.g. one per phone call)

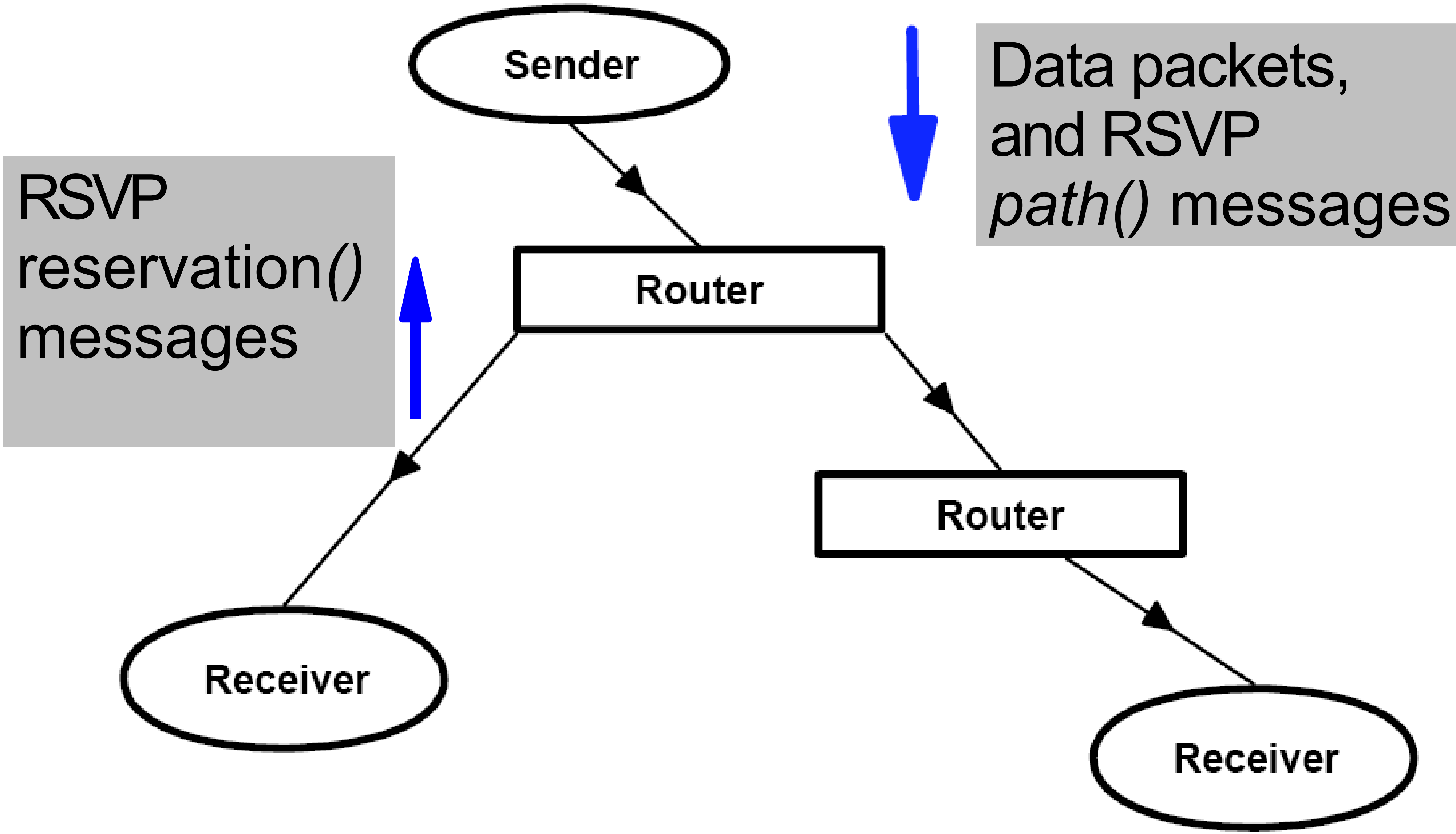
Per-flow approach: Integrated Services and RSVP protocol

- The Integrated Services architecture defines three fundamental types of service:
  - Best Effort. Today's Internet
  - Guaranteed Service. End-to-end delay (one-way trip) will not exceed some value D; throughput will not be less than some value T
  - Controlled Load Service. Loss rate as close to zero as possible, end-to-end delay as close to the minimum as possible
- RSVP protocol is used to request Guaranteed or Controlled Load service end-to-end

RSVP basics

- RSVP = Resource reSerVation Protocol
  - Resources must be reserved all along the transmission path, in every router and switch
- RSVP messages include Tspecs defining:
  - p = peak rate in bytes/sec
  - b = bucket size in bytes
  - r = bucket rate in bytes/sec
  - M = maximum packet size in bytes
  - m = minimum policed unit in bytes
- A flow conforms to the Tspec if the number of bytes sent in time t is $< pt+M$ and $< b+rt$

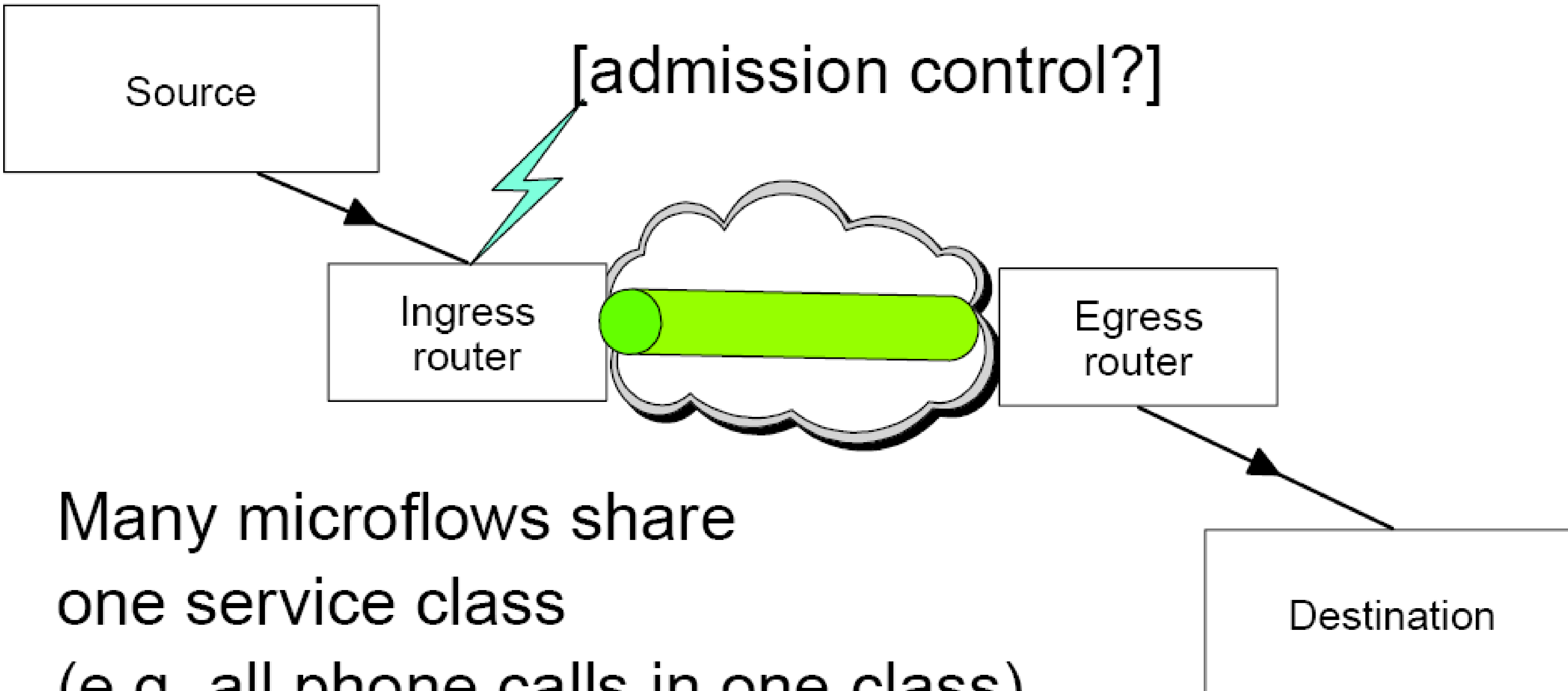
RSVP scenario



Skipping a lot of details...

- After a successful RSVP message exchange, all routers along the path have reserved bandwidth for a particular flow, conforming to the given Tspec
- At Internet scale, every backbone router must
  - hold state for millions of communications
  - classify each packet on
    {source addr, source port, destination addr, dest port, protocol}
    to see if they are Int Serv packets
  - apply controlled load or guaranteed service checks to each Int Serv packet
- Unfortunately, that is quite impossible in practice due to overload and complexity. We need something simpler

Architecture 2: Pre-existing service classes



Many microflows share one service class (e.g. all phone calls in one class)

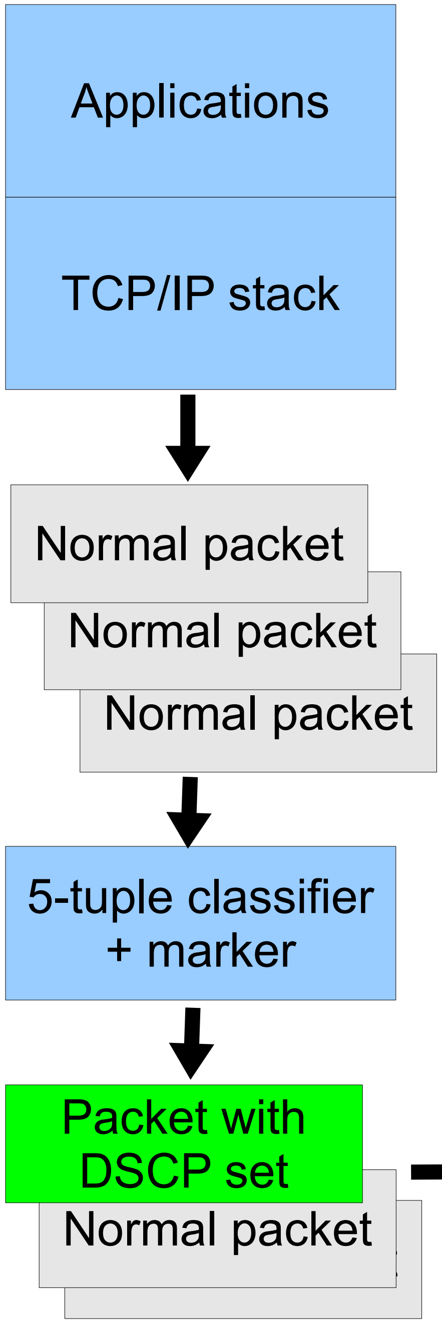
Differentiated services: problems to be avoided

- Avoid per-flow state in backbone routers
  - This suggests aggregating traffic requiring similar QoS into "mega-flows" known as traffic aggregates
  - Put those mega-flows into pre-existing service classes
  - Avoids per-flow state and RSVP signalling
- Avoid expensive per-packet processing
  - This suggests avoiding the 5-tuple classifier and using something simpler such as a flag byte

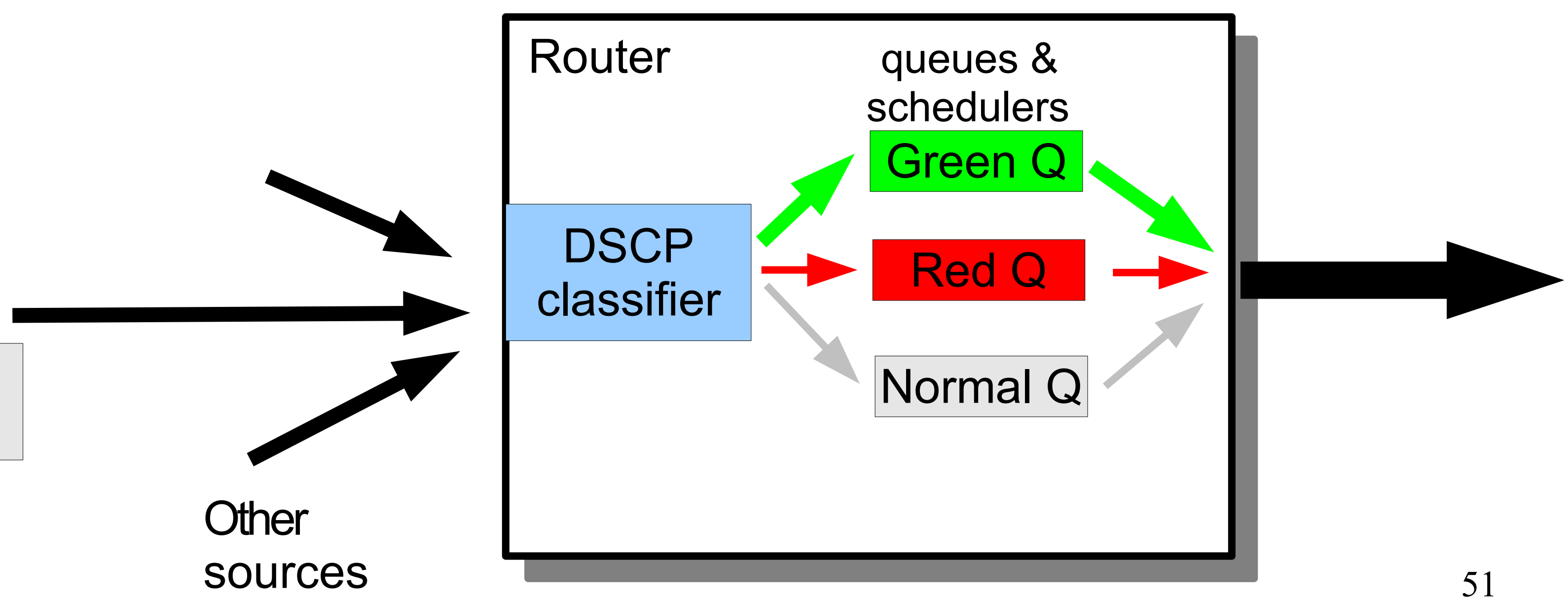
diffserv basic idea

- At input to each router, sort packets into various queues based on a 6-bit differentiated services flag called the DS Field in each IP packet header
  - Think of this as colouring the packet, one colour per queue
- The value in the DS Field is called the DS Code Point (DSCP)
- Queues get different QoS treatments
- Thus each DSCP value effectively asks for a specific queueing/scheduling discipline from a choice of (maximum) 64 classes at each router

Diffserv Overview



- 5-tuple classifier is in source computer or nearby router. Later routers only need to test-and-branch on a 6 bit value
- Queues for different PHBs are configured to provide different grades of service
- A "green" packet will go in the "green" queue at every router



The DS Field in an IP packet

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|         DSCP          |  ECN  |
+---+---+---+---+---+---+---+---+
```

DSCP: diffserv code point
ECN: reserved for explicit congestion notification

Uses TOS octet in IPv4 and Traffic Class octet in IPv6

Diffserv is hop-by-hop

- Each forwarding device (router or IP-aware switch) must interpret the DSCP in each incoming packet
- Each DSCP bit pattern is a code point for a specific forwarding treatment or **per hop behaviour (PHB)**, i.e. a queueing discipline
- The end-to-end network service is the result of concatenating the classifier and N routers implementing the specified PHB
  - Inevitably the service quality is statistical

Diffserv classification

- The source may be its own classifier
  - for example, an IP phone should set a DSCP suitable for voice traffic
- The classifier will probably also be a shaper (on the subscriber side) or policer (on the ISP side)
- Classification policy may depend on an SLA

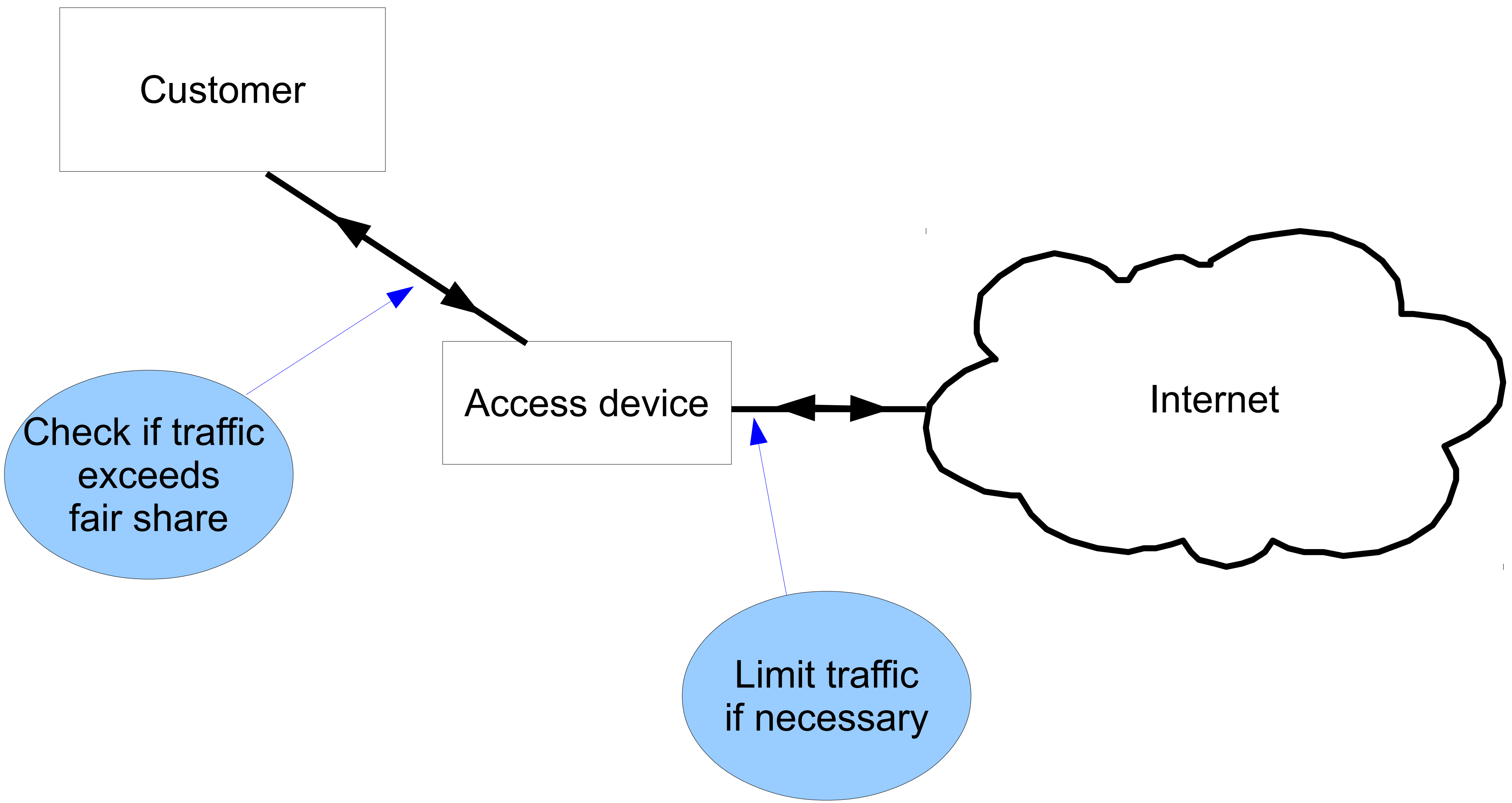
Efficiency of diffserv model

- Slow per-flow jobs done at the edges, where number of users per router is modest:
  - Classification of packets into flows (checking each packet against 5-tuple filters)
  - Traffic shaping/admission control
- Only fast processes in backbone routers:
  - Selecting output queue (PHB) by indexing off a 6-bit code point (64 entry table)
  - No 5-tuple processing
  - No per-flow state

Example PHBs (DSCP values)

- DSCP = 000000 (default)
  - Best Effort (BE) behaviour. This queue gets whatever capacity is left on each link after other queues are empty
- DSCP = 101110
  - Configured to request Expedited Forwarding (EF) behaviour. Queue algorithm **must** provide a specified minimum throughput **and** meet delay and jitter targets **and** minimise loss
- Note that BE service is today's Internet and EF service suits applications such as VoIP
- A number of other PHBs between these extremes are defined
  - Further reading: RFC 2474, 2475, 2597, 3246, 3247, 4594

Architecture 3: Fair share per customer



What is a fair share?

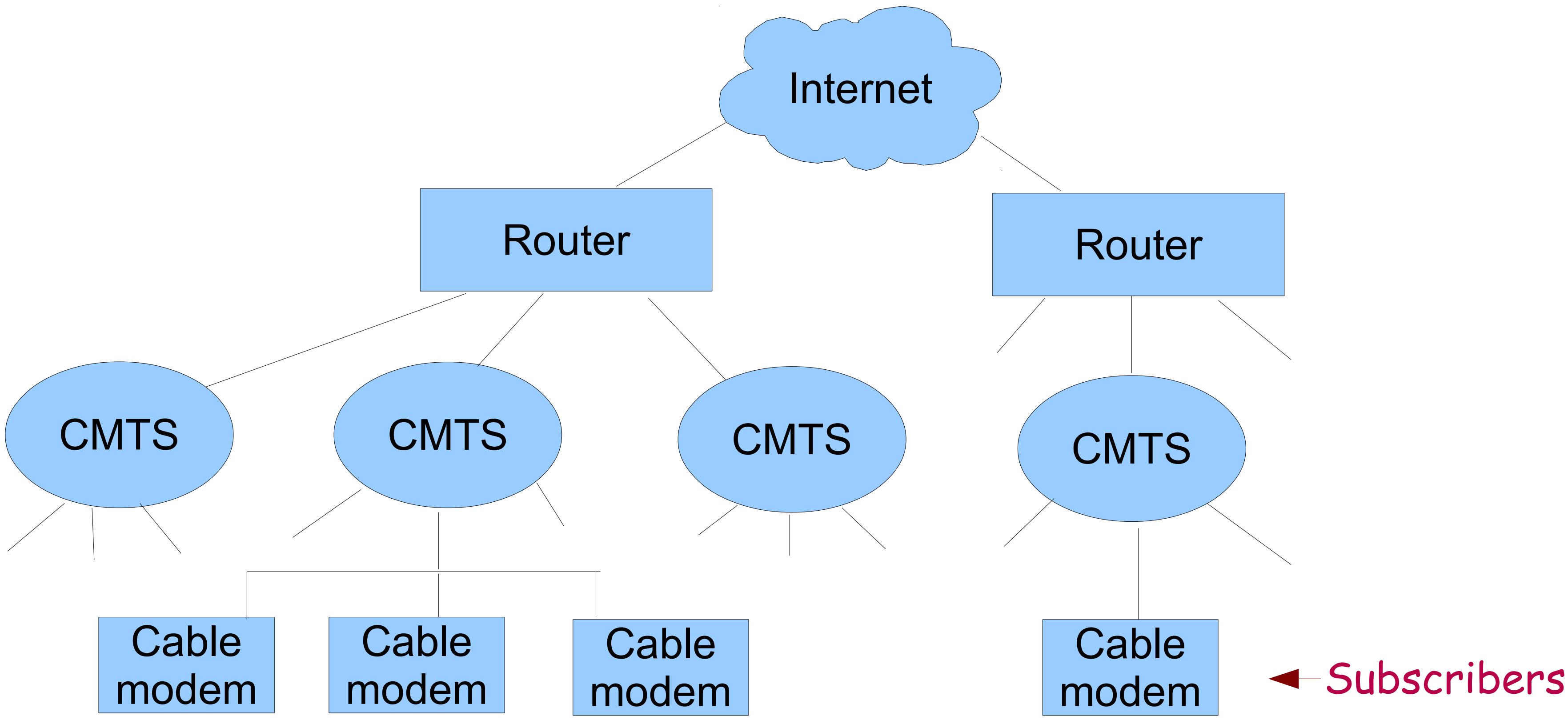
- The best effort service works well if everyone observes TCP-like behaviour (slow down when congestion is detected)
  - short bursts of very heavy traffic are OK
  - continuous very heavy traffic is unfair when network is congested, so must slow down
  - UDP streams or P2P torrents don't slow down
- Blocking individual streams or torrents is not "network neutral" and is complicated to do
- Detecting that a user is generating continuous very heavy traffic is much easier, and is neutral

The Comcast solution (1)

- Comcast is a large ISP in the USA, using Cable TV as the Internet access technology. They got into political trouble by trying to detect and block BitTorrent traffic
  - Involved inspecting contents of individual IP packets (privacy and network neutrality problems)
- New technique is to detect customers exceeding a fair share and limit their traffic
- Can be done at layer 2 without even looking inside the IP packets
  - More efficient than IP packet inspection, and less intrusive

The Comcast solution (2)

- Implemented in the “Cable Modem Termination System” (CMTS) [see RFC 6057]
  - CMTS is a multiport switch providing link-layer connection between subscribers and routers



The Comcast solution (3)

- CMTS logs traffic on each port
  - if the port's total traffic > 70% of physical capacity for 15 minutes, port is marked as "Near Congestion State"
  - then if a subscriber's average traffic > 70% of paid capacity for 15 minutes, subscriber is marked as "Extended High Consumption State"
  - then that subscriber is downgraded from "priority best effort" to "best effort" (i.e., packets will be dropped first when queue in the CMTS is full)
  - subscriber will be upgraded again when traffic < 50% of paid capacity for 15 minutes
- Do you think this is fair? Will it prevent all congestion?

Internet QoS in practice

- Traffic engineering techniques are in widespread use
- Integrated Services with RSVP - very little used
  - (RSVP has been recycled as a TE mechanism for MPLS)
- Differentiated Services - some deployment
  - typically to help VoIP within a company network
- Mostly, operators rely on increasing capacity ("bandwidth") as load increases
  - adjust capacity so that the Internet 'only just works'
  - queueing theory to calculate capacity to meet SLA

Future of QoS?

- Today, there is little use of QoS mechanisms for the consumer Internet or across ISP-ISP boundaries
  - Applications like Skype must rely on best effort service and very clever codecs
  - Numerous ISPs enforce fair shares by one method or another (e.g. “bandwidth” caps, Comcast method)
- Will things change as VoIP and IPTV become more important?
- How will the increase in mobile hosts change things?

Topic 3: Wireless QoS

- What effect does a wireless network have on IP Quality of Service?
  - More generally: what effect do characteristics of a particular type of link have on QoS and performance?

Reminder: What can go wrong with wired links

- Actual bit errors on modern wired links are very rare
  - link-layer retransmit generally not needed and not provided
- Therefore, packet loss and delay are almost exclusively caused by congestion
  - RTT has a definite "speed of light" minimum
  - bit rate has a definite "slowest link" maximum
- Therefore, TCP has evolved mainly to deal correctly with congestion (i.e. slow down when appropriate)
- UDP usage, although more greedy than TCP, is also tuned for congested networks - there's no point sending extra packets

Differences with wireless

- Actual bit errors are common (interference, noise, fade)
  - 802.11 retransmits on *missing* ACK, and CRC error prevents ACK, so bit errors cause retransmission
    - RTT is highly variable during interference/fade
    - A lost ACK means an unnecessary retransmission!
  - 3G UMTS and 802.16 have an *active* retransmission mechanism (ARQ, Automatic Repeat Request)
    - More responsive than "missing ACK" but still varies RTT during interference/fade
- Maximum bit rate is highly variable (link budget)
  - e.g. 802.11: 1 to 54 Mb/s according to where you stand!
- Radio channel is intrinsically simplex, so you often have to wait for the medium (contention, RTS/CTS)

Transport's viewpoint

- In addition to congestion loss (which is essentially randomly distributed in the packet stream), when one or both ends are wireless, TCP and UDP will see
  - bursts of increased RTT (due to packet loss followed by *link layer* retransmission, or contention)
  - sudden very large changes in achievable bit rate (due to link budget changes)
  - occasional moments of silence due to handover between cells/base stations

(We're assuming no change of IP address due to handovers.)

What will TCP do?

- In an established TCP session, increase in RTT directly reduces TCP throughput
- Suppose RTT is 10 ms and bit rate is 54 Mb/s, but a burst of interference on 802.11 can *increase* RTT to 15 ms and *reduce* bit rate to 1 Mb/s
  - TCP will automatically reduce offered load by 33% (inversely proportional to RTT)
  - actual rate will be capped at 1 Mb/s, so TCP may enter congestion control due to lost packets and missing ACKs

What will UDP do?

- In a UDP stream, increase in RTT is really just a one-time delay to the packet stream, so doesn't matter too much (small audio/video glitch)
- Suppose bit rate is 54 Mb/s, but a burst of interference on 802.11 reduces it to 1 Mb/s
  - UDP will not reduce offered rate
  - actual rate will be capped at 1 Mb/s
  - possible data loss (for video)
  - not much impact on VoIP
  - competing TCP sessions will suffer since UDP doesn't slow down

TCP response to asymmetric paths (RFC 3449)

- Examples of asymmetric paths (downlink from server ≠ uplink from client)
  - 802.11 (lower power in mobiles than base station)
  - Satellite downlink, modem uplink
  - 3G UMTS and 4G
  - ADSL or CATV broadband - downlink >> uplink
- In all these cases, downlink bandwidth will be several times uplink, and packet delays may also be larger for uplink
  - Most TCP development work has assumed ~ symmetrical bandwidth and delay, and in particular that ACKs arrive as promptly as data packets

Case 1: TCP ACKs arrive slowly

- Consider an example* of 10 Mb/s downlink (data) and 50 kb/s uplink (ACKs). Capacity ratio is 200:1
- Assume 1000 byte data segments and 40 byte ACKs. Packet size ratio is 25:1
  - Relative capacity in packets is 200/25 = 8:1
  - If we send more than one ACK per 8 data segments, ACK channel will saturate
- In any case, ACKs arrive at a max. rate corresponding to the uplink, so TCP will clock its sending rate accordingly, and under-use the downlink

*These numbers could apply to a VSAT satellite connection with a modem return path

Case 2: TCP ACKs are lost

- Same example of 10 Mb/s downlink (data) and 50 kb/s uplink (ACKs), 1000 byte data segments and 40 byte ACKs. Now assume uplink is overused, so many ACKs are dropped
  - If TCP sender receives only one ACK in *k*, it transmits data in bursts of *k* (or more) segments, because each ACK shifts the sliding window by at least *k.* Burstiness causes loss
  - TCP senders increase their *cwin* by counting the number of ACKs they receive and not by how much data is actually acknowledged by each ACK. Lost ACKs imply a slower rate of growth of the *cwin*, which degrades performance
  - TCP's Fast Retransmission and Fast Recovery algorithms are less effective when ACKs are lost

More about RTT impact

- The observed RTT and its variability are used by TCP to compute the RTO (retransmission timeout - how long to wait for a missing ACK), e.g.

$$RTO = RTT + 4\text{x}RTTVAR$$

- The high RTTVAR on a wireless network can quickly inflate RTO to several seconds
  - This amplifies the performance impact of lost packets or lost TCP ACKs

Making TCP handle asymmetry

- Send delayed ACKs, but not by more than 500 ms or 1 segment
- Use largest possible segment size (after MTU discovery)
  - but beware, larger packets are more subject to bit errors
- Use VJ header compression (RFC1144). This greatly reduces the size of ACKs when losses are infrequent
- Messing with the ACK stream. Various techniques (the simplest is random discard of ACKs) have been tried. All have side-effects, none are recommended
- ACK flow management. Use queue management on uplink (diffserv-like) to ensure smooth flow of ACKs

TCP and 3G UMTS (RFC 3481)

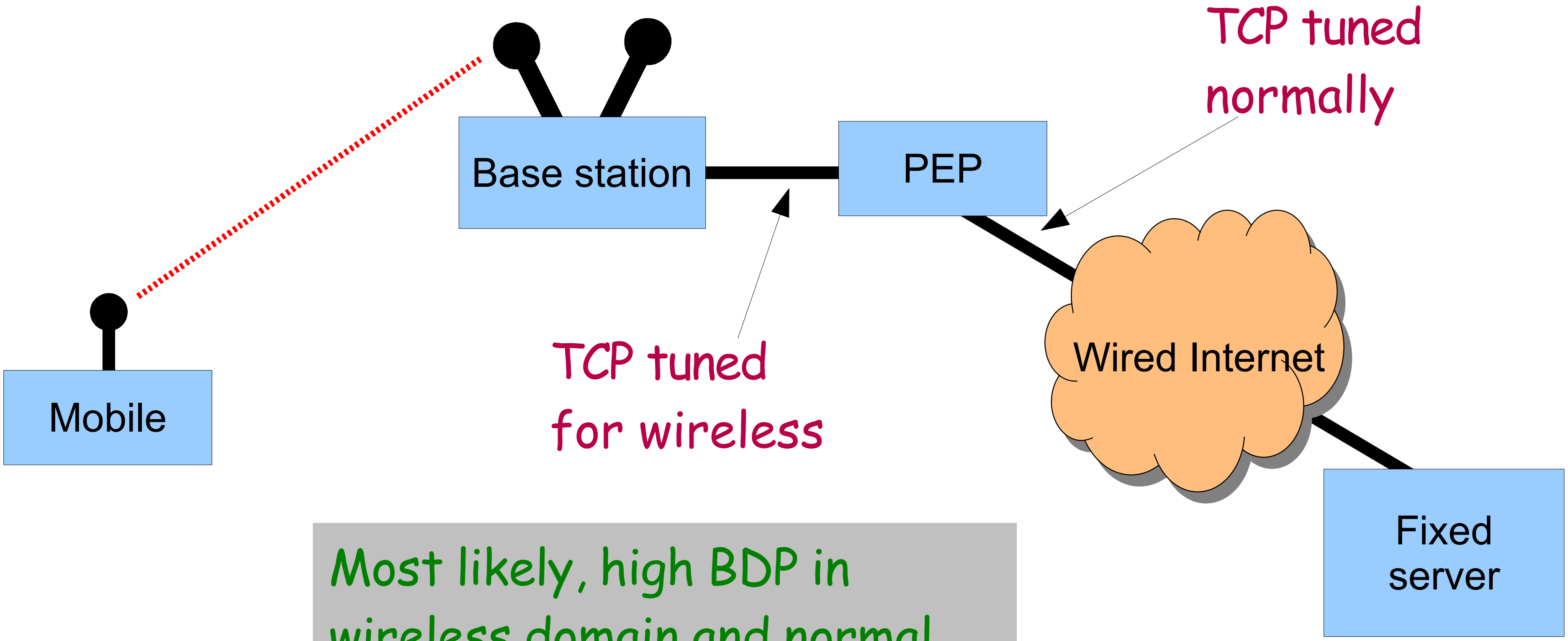
- RTT in 3G is between a few hundred ms and 1 second
  - Same sources of variability as 802.11 (interference, fade, ARQ retransmission), plus delay spikes during cell handover
  - Cell handover can also cause systematic change in RTT (e.g. moving from a quiet cell to a very busy one as you approach downtown)
    - → high RTT variability → large RTO → idle link
  - Asymmetry effects also present (e.g. 64 kb/s uplink and 384 kb/s downlink)
  - Bandwidth-delay product (BDP) of around 10-24 KB (but how do you calculate BDP with an asymmetric link??)
    - (Compare BDP for a 10 Mb/s,100 ms link = 125 KB)
    - In any case, relatively small TCP window and potentially large RTO → poor throughput

TCP config for 3G

- Ensure receive buffer allows for adequate window (2xBDP = 50kB)
- Set initial *cwin* to ~4kB
- Use largest possible MTU
- Use "limited transmit" (RFC 3042) - send pending packets in response to duplicate ACK, instead of waiting for RTO
- Use TCP timestamps (RFC 1323) to improve RTT estimate
- Use SACK (selective ACK) to avoid go-back-N waste
- Disable VJ header compression *iff* loss rate is high

Performance Enhancing Proxy

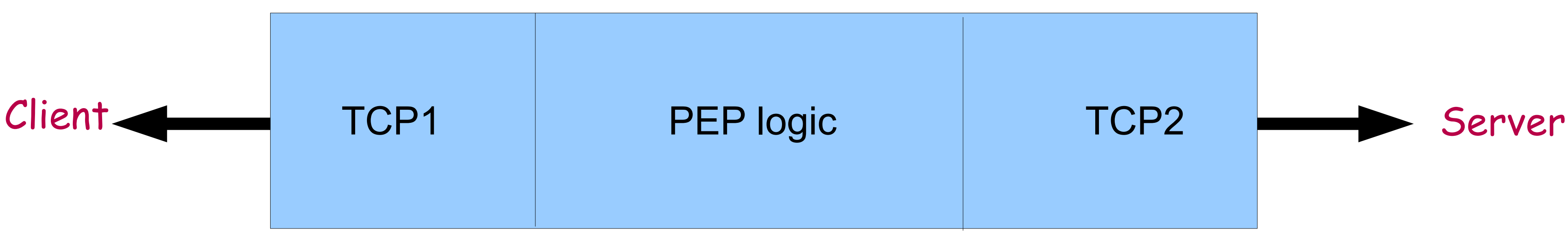
- Idea: mitigate TCP performance problems using a magic box



Most likely, high BDP in wireless domain and normal BDP in wired domain.

PEP mechanisms (1)

- Symmetric vs asymmetric - are streams in both directions handled the same?
  - typically not, if PEP is at wireless/wired boundary
  - in any case, ACK stream and data stream are different
    - (In theory, TCP ACKs are piggy-backed on inverse data, but in practice, most TCP sessions are mainly data one way and mainly ACKs the other)
- Split connection - does the PEP completely terminate one TCP session and create another?



PEP mechanisms (2)

- ACK spacing - smooth out the flow of TCP ACKs
- Local TCP ACKs - generate ACKs inside the PEP, to speed up slow start for the high BDP side
- Local retransmit - detect missing TCP ACKs, and retransmit data from buffer in PEP
  - Split connection → local ACKs and local retransmit
- TCP ACK Filtering and Reconstruction
  - For highly congested uplinks where many ACKs are lost
- Aggressive compression on the slow (wireless) side
  - Will need (de)compression code in mobile node too
- Hide temporary disconnects
  - Buffer data, and indicate zero window size to wired sender

Do you think PEPs are safe?

Do you think PEPs are safe?

- What if a split PEP acknowledges data that are part of a two-phase commit for a banking transaction, but the mobile node goes off the air before those data actually reach it?
- Did the banking transaction complete or not?
- ***Where is your money?***
- There are reasons why we care about the end-to-end principle in distributed systems design

Summary on Wireless QoS

- Wireless mobility adds new QoS and performance problems (asymmetry, temporary disconnects, highly variable RTT)
- There's not much UDP can do about it
- TCP can be tuned and configured to minimise the impact
- Performance Enhancing Proxies bring their own problems
- Now that millions of mobiles have Internet access, these problems are very important