We Don't Need No Control Plane

Michael Welzl, Kashif Munir

Michael Welzl
DPS NSG Team http://dps.uibk.ac.at/nsg
Institute of Computer Science
University of Innsbruck

AGNM 2006
Dublin, Ireland
26-27 October 2006

Proposed architecture

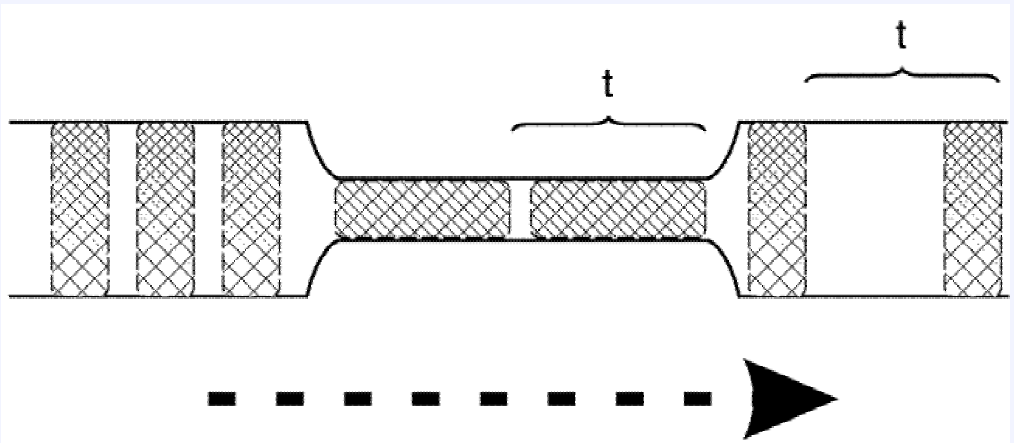
- Goal: efficient per-flow QoS without signaling to routers
  - ultimate dream (*very* long-term goal): without any router involvement!
    (99% instead of 100% reliable guarantees)
- Idea: use traditional coarse-grain QoS (DiffServ) to differentiate between
  - long-lived bulk data transfer with advance reservation (EF) and
  - everything else (= SOAP etc. over TCP) (best effort)
- Allows us to assume isolated traffic; planned to drop this requirement later
- Because data transfers are long lived, apply admission control
  - Flows signal to resource broker (RB) when joining or leaving the network
- Mandate usage of one particular congestion control mechanism for all flows in the EF aggregate
  - Enables efficient resource usage because flows are elastic

Key ingredients of our QoS soup

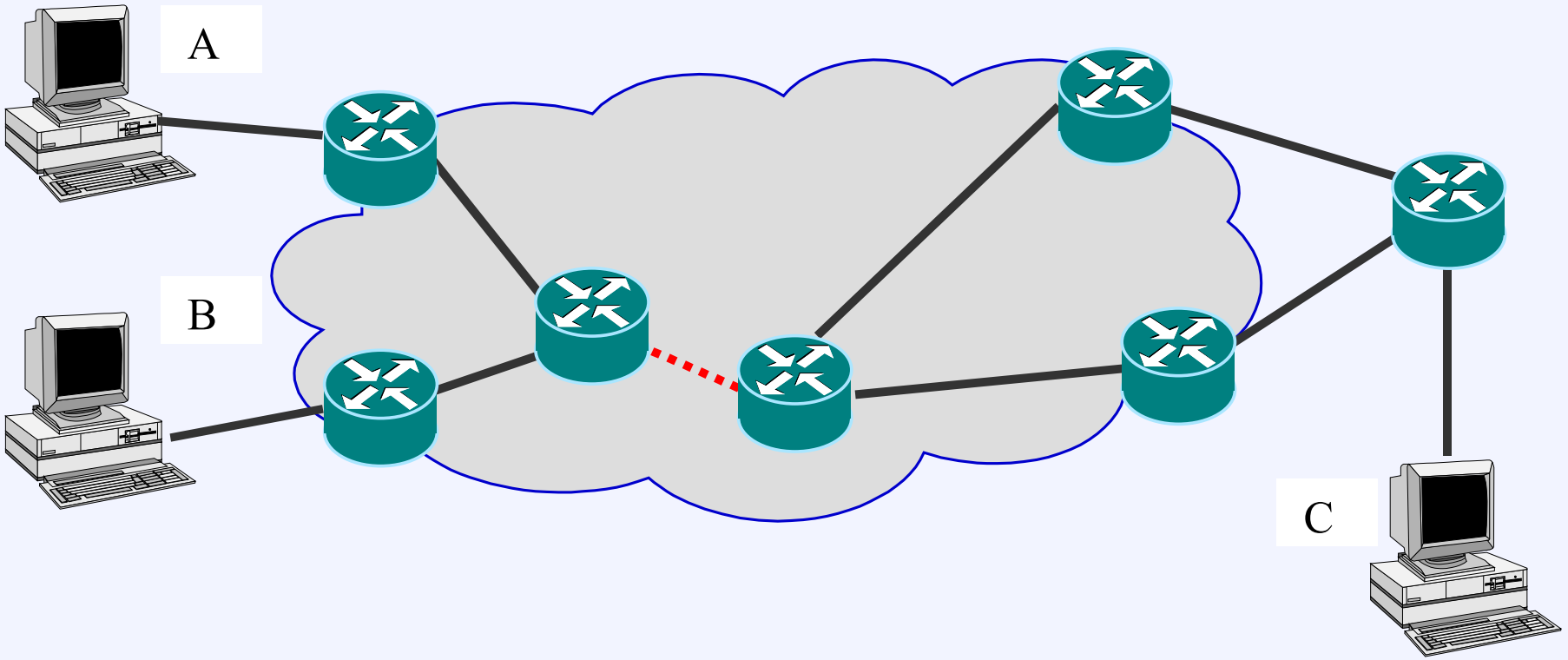
- Link capacities must be known, paths should be stable (capacity information should be updated upon routing change)
- Shared bottlenecks must be known
- Bottlenecks must be fairly shared by congestion control mechanism irrespective of RTT (max-main fairness required, i.e. all flows must increase their rates until they reach their limit)
- No signaling to routers = no way to enforce proper behavior ⇒ there must be no cheaters
  - User incentive: fair behavior among cooperating nodes among which Grid application is distributed
  - Unfair behavior between Grid apps 1 and 2 in same Grid neglected (usually acceptable, as used by same Virtual Organization)

Link capacities must be known

- Can be attained with measurements
- Working on permanently active, (mostly) passive measurement system for the Grid that detects capacity with packet pair
  - send two packets p1 and p2 in a row; high probability that p2 is enqueued exactly behind p1 at bottleneck
  - at receiver: calculate bottleneck bandwidth via time between p1 and p2
  - e.g. TCP: "Delayed ACK" receiver automatically sends packet pairs
    ⇒ passive TCP receiver monitoring is quite good!
  - exploit longevity - minimize error by listening for a long time



Shared bottlenecks must be known



- Simple basis: distributed traceroute tool
  - enhancement: traceroute terminates early upon detection of known hop
- Handle "black holes" in traceroute
  - generate test messages from A, B to C - identify signature from B in A's traffic
  - method has worked in the past: "controlled flooding" for DDoS detection

Congestion Control mechanism must be max-min fair

- Was once said to be impossible without per-flow state in routers
  - not true; XCP and some others
  - but these explicit require router support...

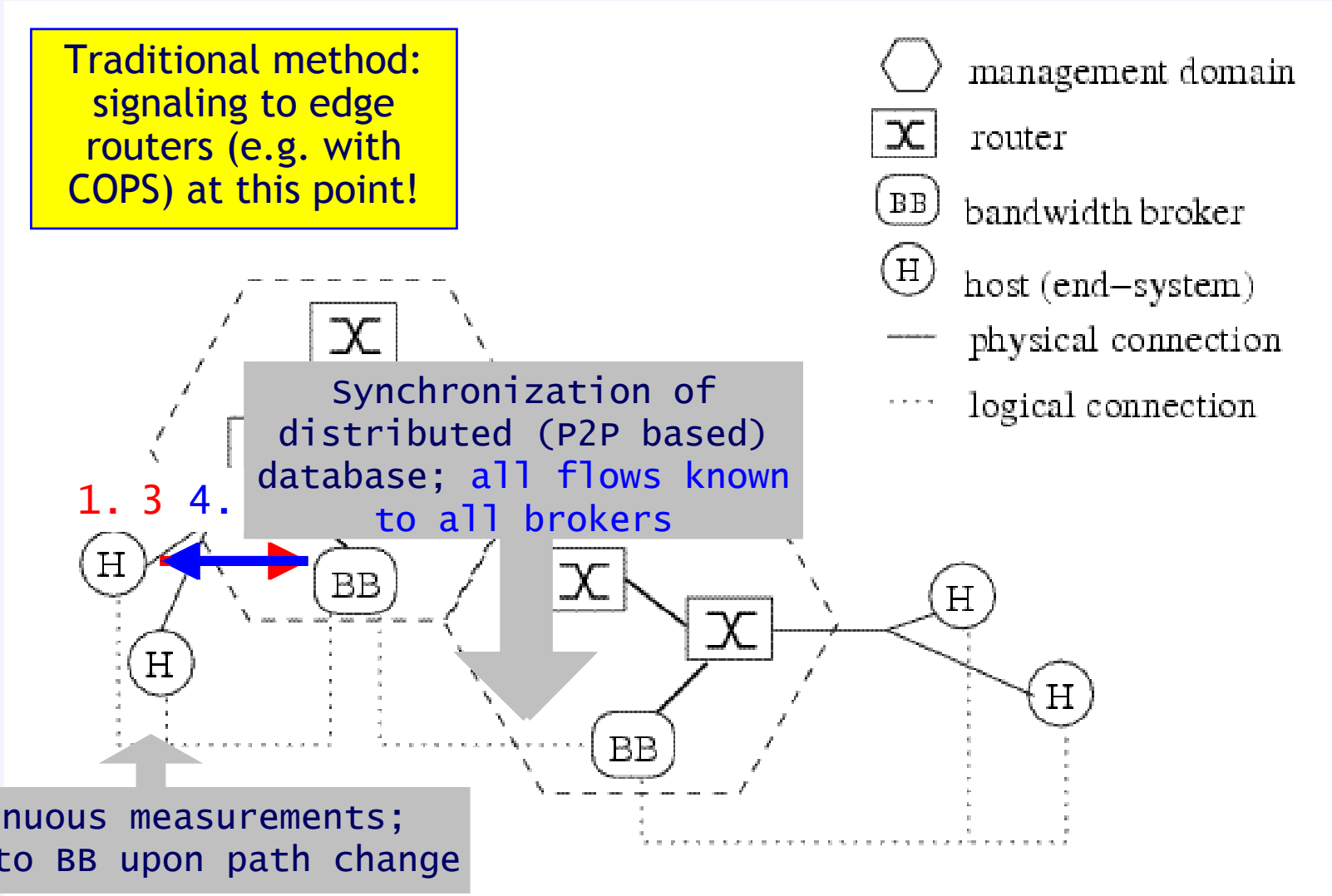
- Main problem: dependence on RTT
  - three good indications that this can be removed without router support

1. CADPC/PTP (my Ph.D. thesis)...
   - max-min fairness based on router feedback, but only capacity and available bandwidth (could also be obtain by measuring)
2. Result in old paper on phase effects by Sally Floyd
3. TCP Libra

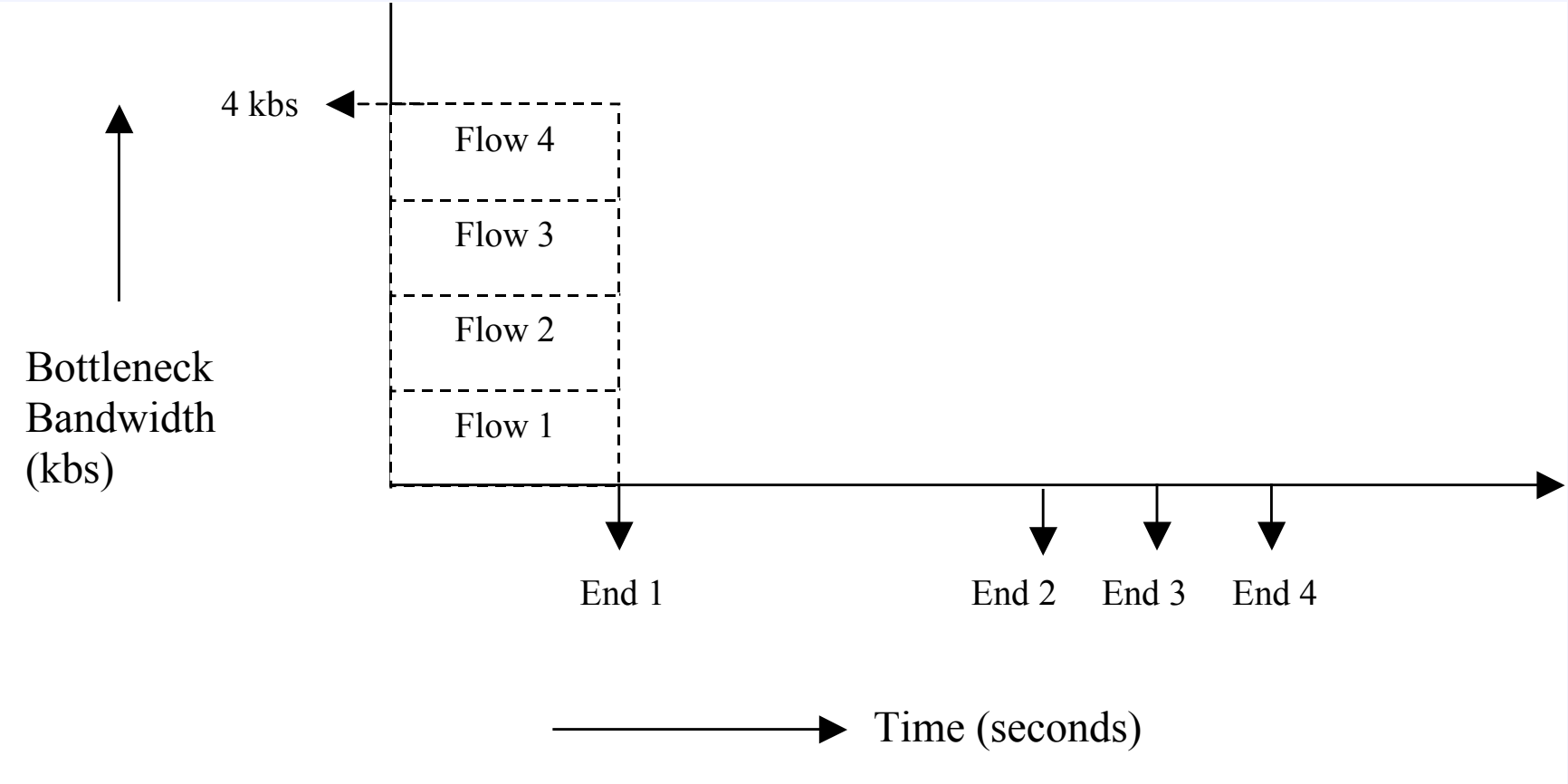
} increase/decrease factors are f(RTT)

- Problem: efficiency - no max-min fair "high-speed" CC mechanism without router support
  - now: plan to change existing one based on knowledge from above examples

Per-flow QoS without signaling to routers

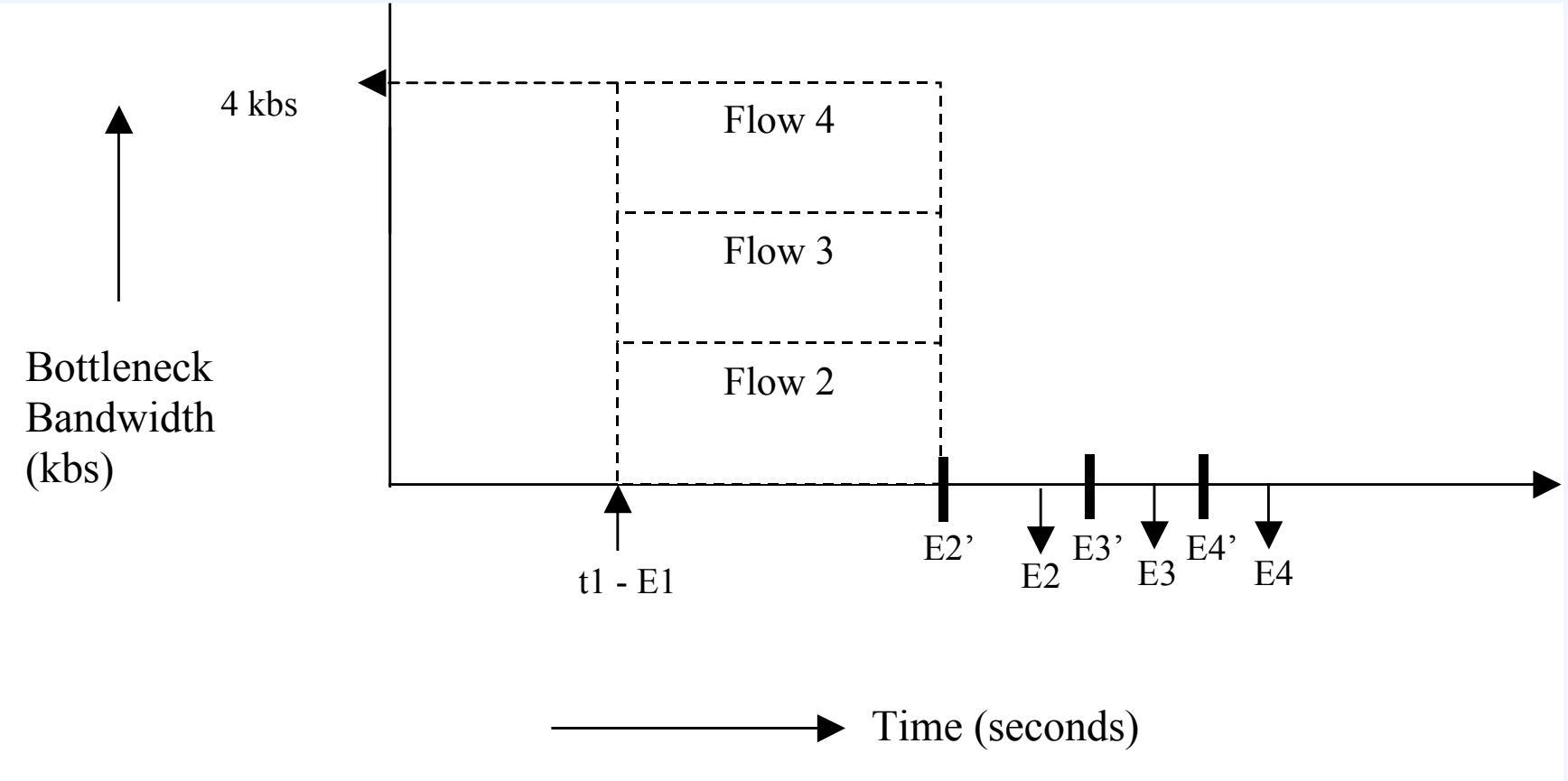


Efficiency via elasticity



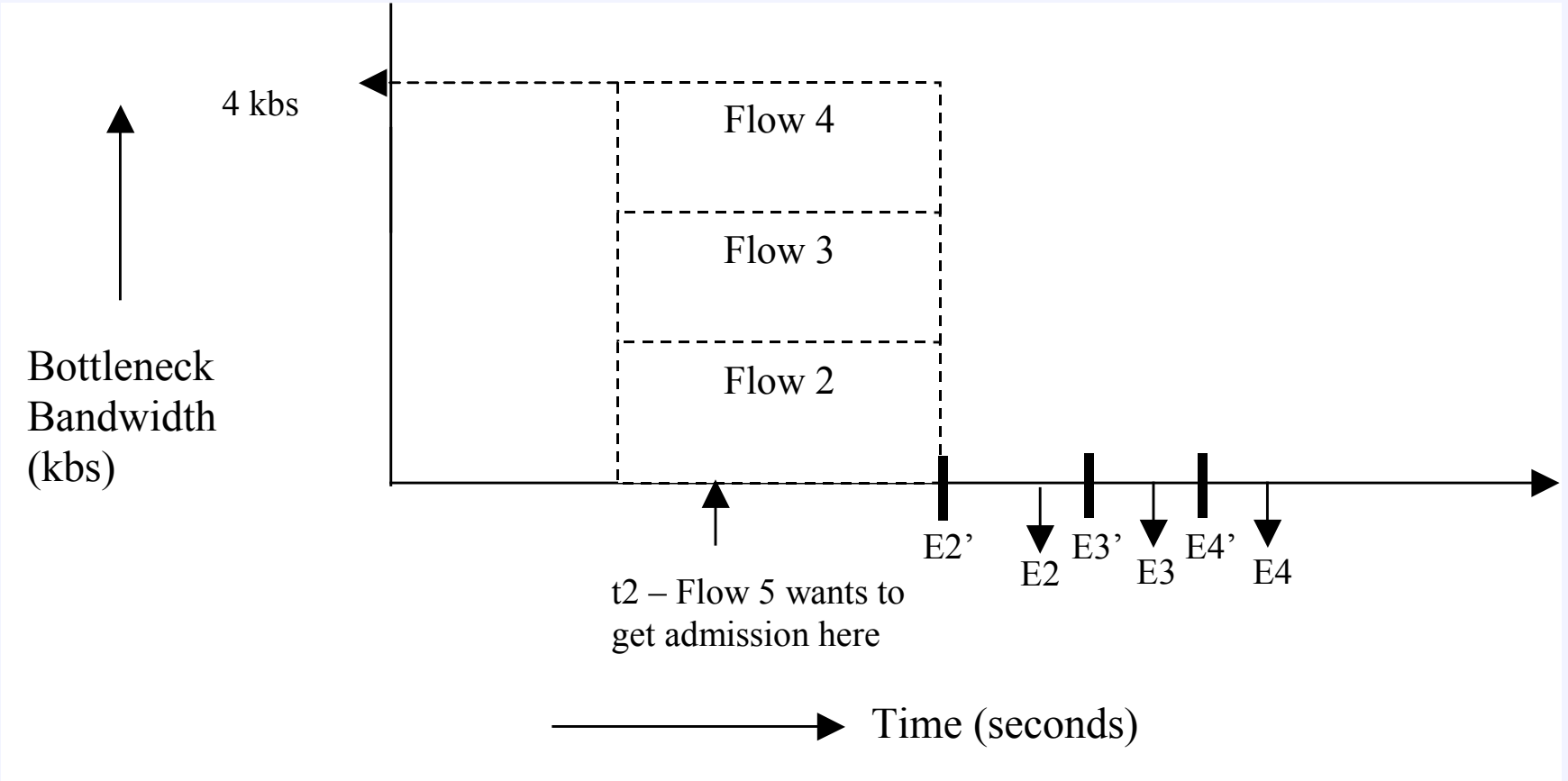
- QoS guarantees in Grid: „File will be transferred within X seconds"
  ⇒ enables flexible resource usage

Efficiency via elasticity /2



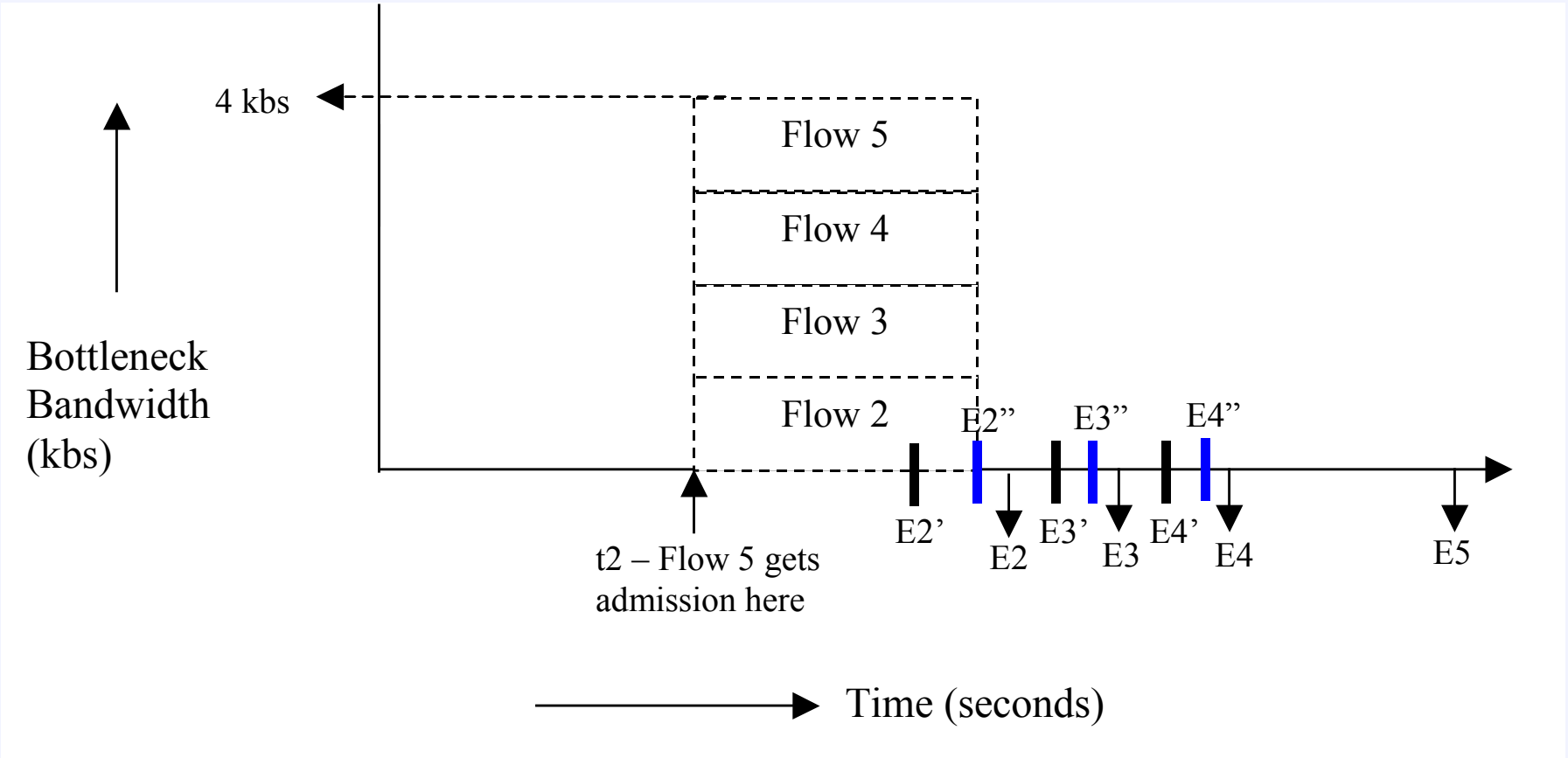
- Flow 1 stopped, flows 2-4 automatically increase their rates
  - leading to earlier termination times E2'-E4'; known to (calculated by) BB

Efficiency via elasticity /3



- Flow 5 asks BB for admission
  - BB knows about current rates and promised E2-E4, grants access

Efficiency via elasticity /4

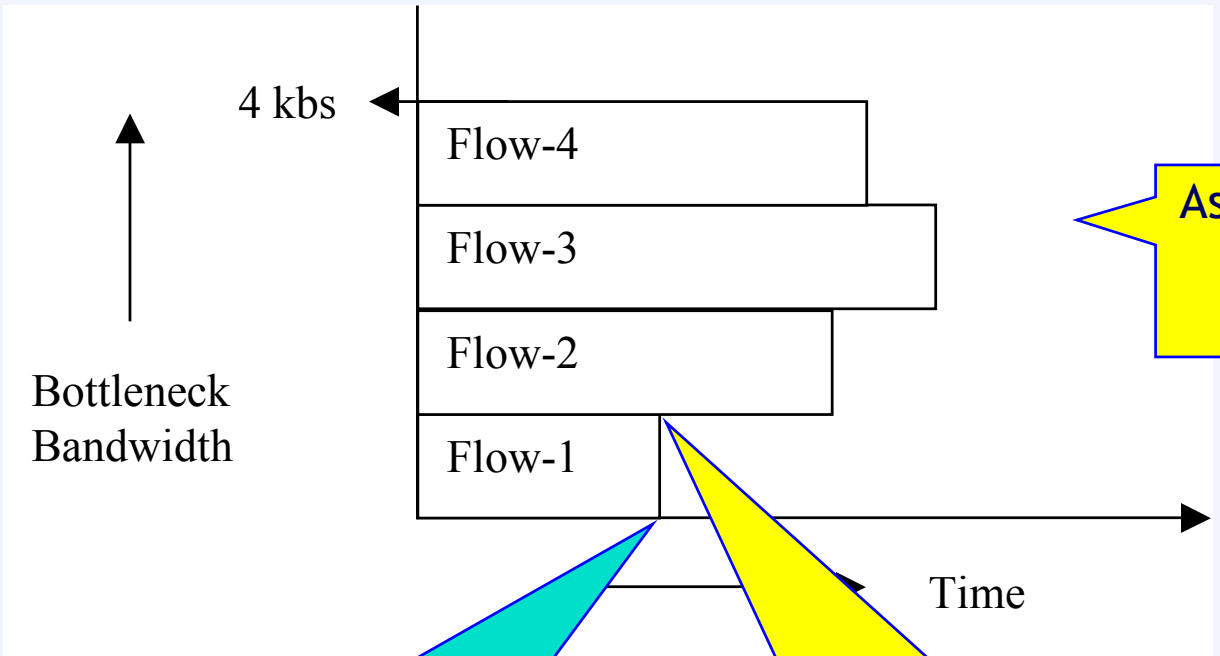


- Flow 2 terminates in time
  - Flows 3-5 will also terminate in time

Additional flow admitted and earlier termination times than promised!

Elasticity without Congestion Control?

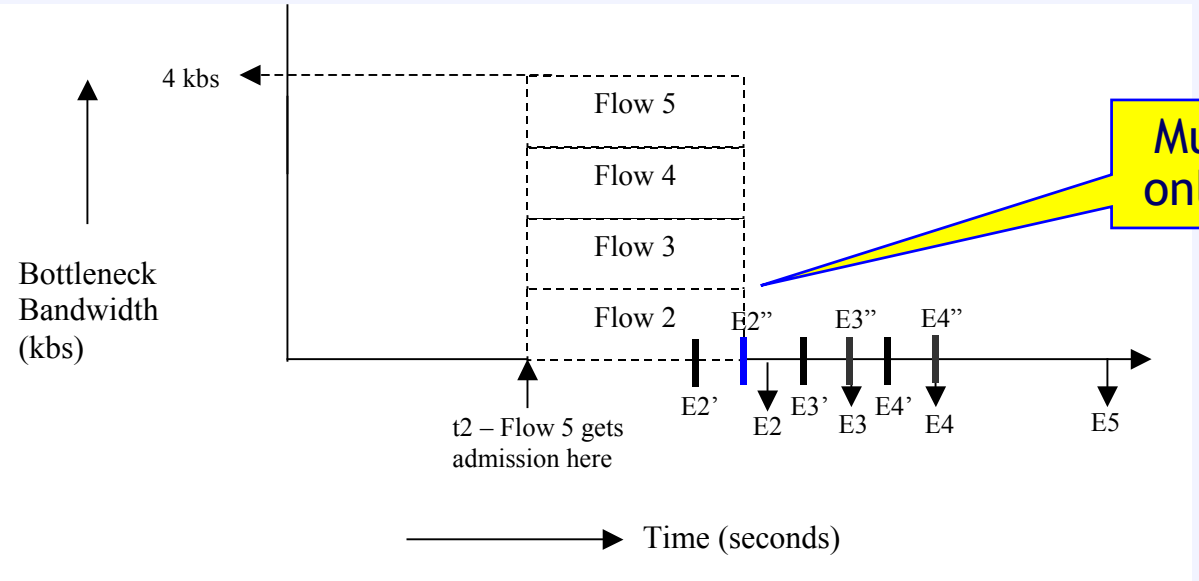
- Significant amount of additional signaling necessary



Additional considerations

- How to assign different rates to different flows?
  - max-min fairness: if a sender "acts" like two, it obtains twice the rate
  - consider rate consisting of slots (e.g. 1 kbit/s = 1 slot)
  - flows can consist of several slots
  - let congestion control mechanism operate on slots

- Possibility: admit new flows even in scenario below



Must introduce unfairness: only flow 2 can reduce rate

Disadvantage: more signaling again!

Difficult & distant future work

- Drop requirement of traffic isolation via DiffServ
  - constantly obtain and update conservative estimate of available bandwidth using packet pair (works without saturating link)
  - ensure that limit is never exceeded; "condition red" otherwise!
  - Some open questions...
    - does this require the CC mechanism to be TCP-friendly?
    - condition red: reduce slots, or let flows be aggressive for a short time?
- How to handle routing changes
  - will be noticed, but can reduce capacity ⇒ break QoS guarantee
  - condition red; can happen in worst case, but to be avoided at all cost
  - mitigation methods
    - very conservative estimate of available bandwidth; leave headroom
    - tell senders to reroute via intermediate end systems
- Bottom line: lots of complicated issues, but possible to solve them

Thank you!

Questions?