QoS Guarantees

- introduction
- call admission
- traffic specification
- link-level scheduling
- call setup protocol
- reading: Tannenbaum, 393-395, 458-471

  Ch 6 in Ross/Kurose

Motivation

Certain applications require minimum level of network performance:

- Internet telephone, teleconferencing: delays > 500ms impair human interaction
- session guaranteed QoS or is blocked (denied admission to network)
- starting to look like telephone net!

Fundamental mismatch between QoS and packet switching

- **packet switching:** statistically share resources in hope that sessions' peak demands don't coincide
  - 100% certain guarantees require accounting for worst case behavior (no matter how unlikely)
- admitting/denying session on worst case demands equivalent to circuit switching!

Internet service classes

Current service model: "best-effort"

- send packet and hope performance is OK

Next generation Internet service classes:

guaranteed service: "provides firm (mathematically provable) bounds on end-to-end datagram queuing delays. This service makes it possible to provide a service that guarantees both delay and bandwidth."

controlled load: "a QoS closely approximating the QoS that same flow would receive from an unloaded network element, but uses capacity (admission) control to assure that this service is received even when the network element is overloaded."

best effort: current service model

ATM service classes

ATM service classes:

CBR: constant bit rate, guaranteed bandwidth, constant end-end delay

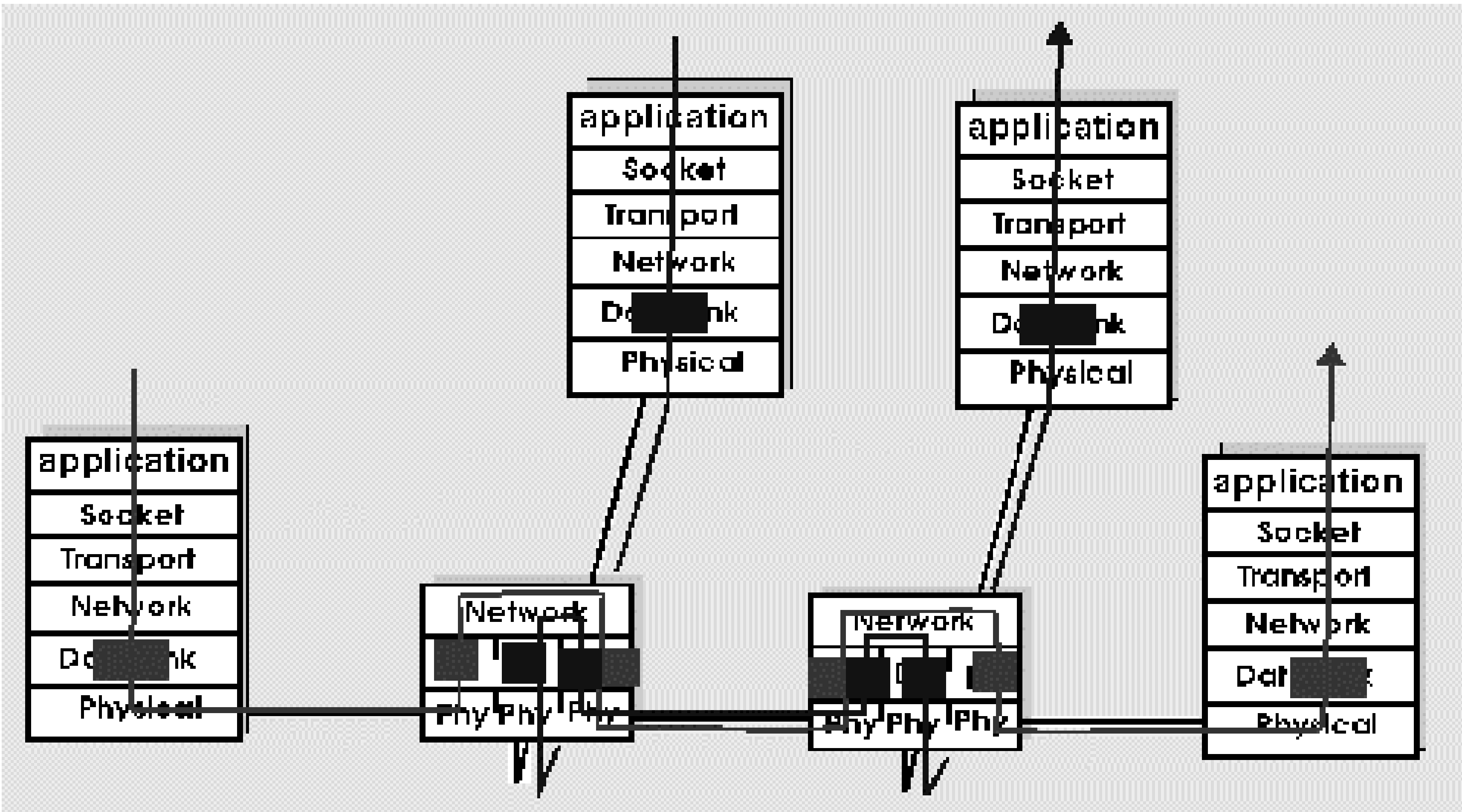
ABR: guaranteed minimum cell rate (bandwidth), more possible if available (congestion control via RM cells)

UBR: unspecified bit rate, no congestion control

Comparing Internet and ATM service classes:

- how are guaranteed service and CBR alike/different?
- how are controlled load and ABR alike/different?

Radical changes required!



Question: what new concepts, mechanisms needed to insure that packets from source see a given QoS?

The call admission problem

Network must decide whether to "admit" offered call (session)

Current networks: all calls accepted, performance degrades as more calls carried

Question: can requested QoS be met while honoring previously made QoS commitments to already accepted calls?

Questions to be answered:

- how much traffic will be injected by call into net, how should that traffic be described (is rate (pkts/sec) enough)?
- what if call sends more traffic than it claimed it would?
- QoS requirement is end-to-end, how to break into per-hop requirements?
- what resources will be needed (bandwidth, buffering) to meet QoS requirement?
- how to reserve resources for call at each hop: call setup protocol
  - current networks: routers play no role in call admission

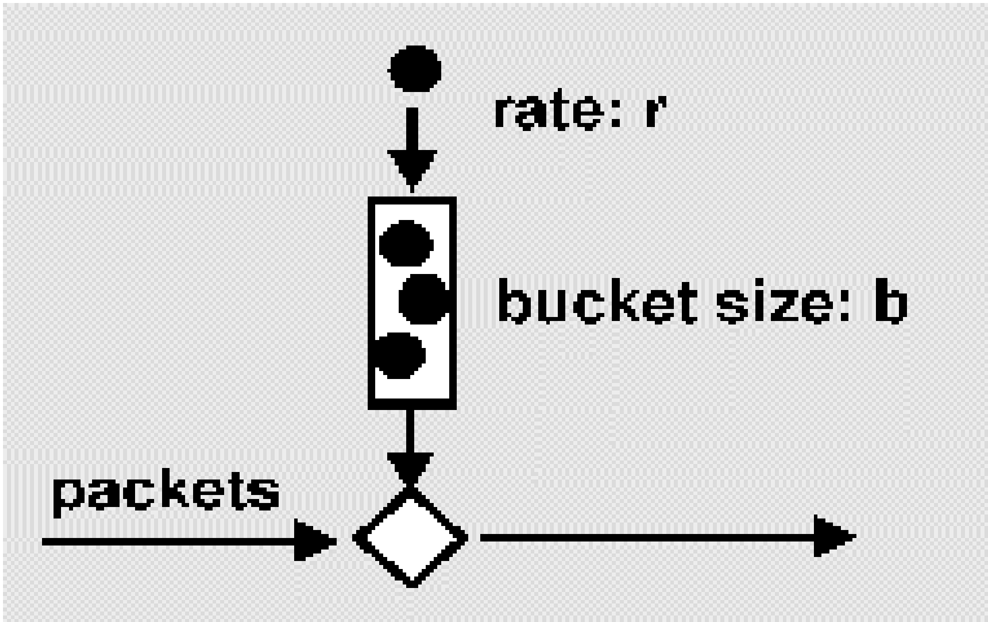
Answers not yet known!

Specifying traffic: Tspec

call must describe traffic that it will inject into net

leaky bucket proposal: traffic entering net filtered by leaky bucket regulator:

- b: maximum burst size
- r: average rate
- amount of traffic entering over any interval of length t, less than b + rt



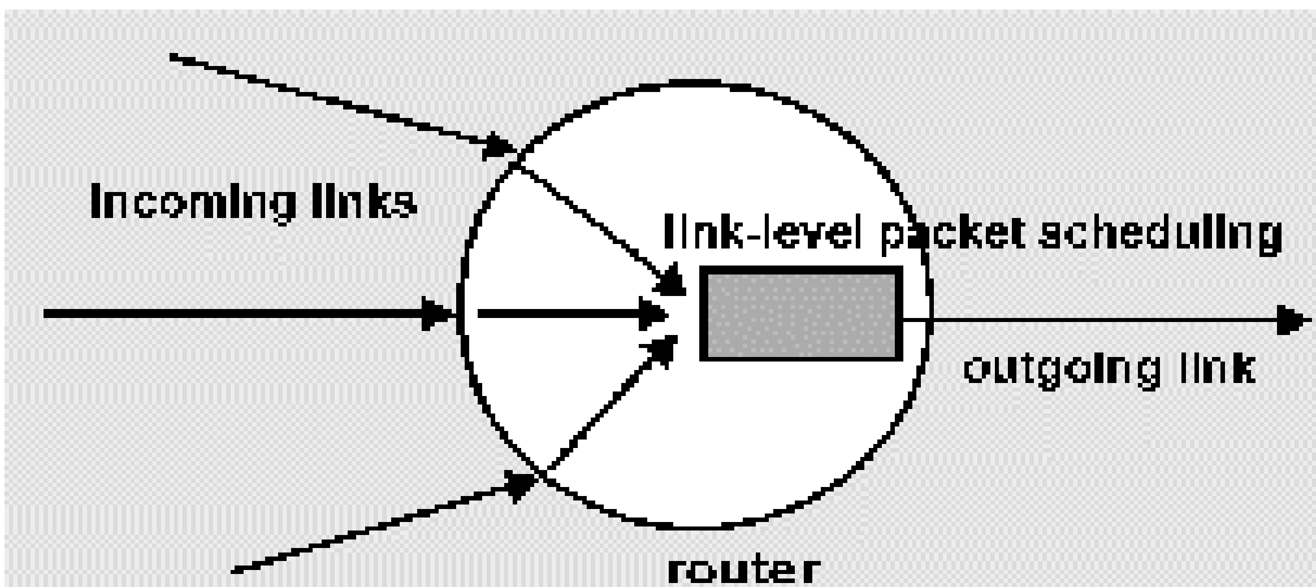
Possible token bucket uses: shaping, policing, marking

- delay pkts from entering net (shaping)
- drop pkts that arrive without tokens (policing function)
- let all pkts pass thru, mark pkts: those with tokens, those without
- network drops pkts without tokens in time of congestion (marking)

Link Layer: critical QoS component

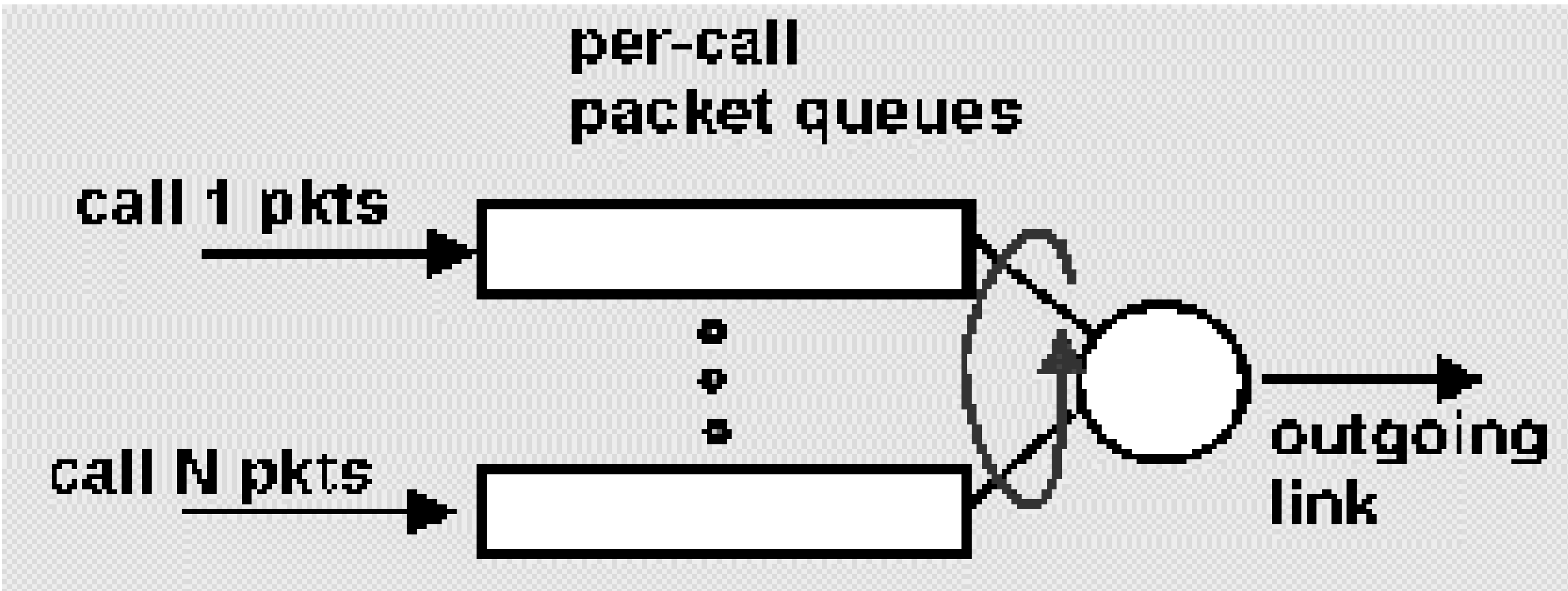
Buffering and bandwidth: the scare resources

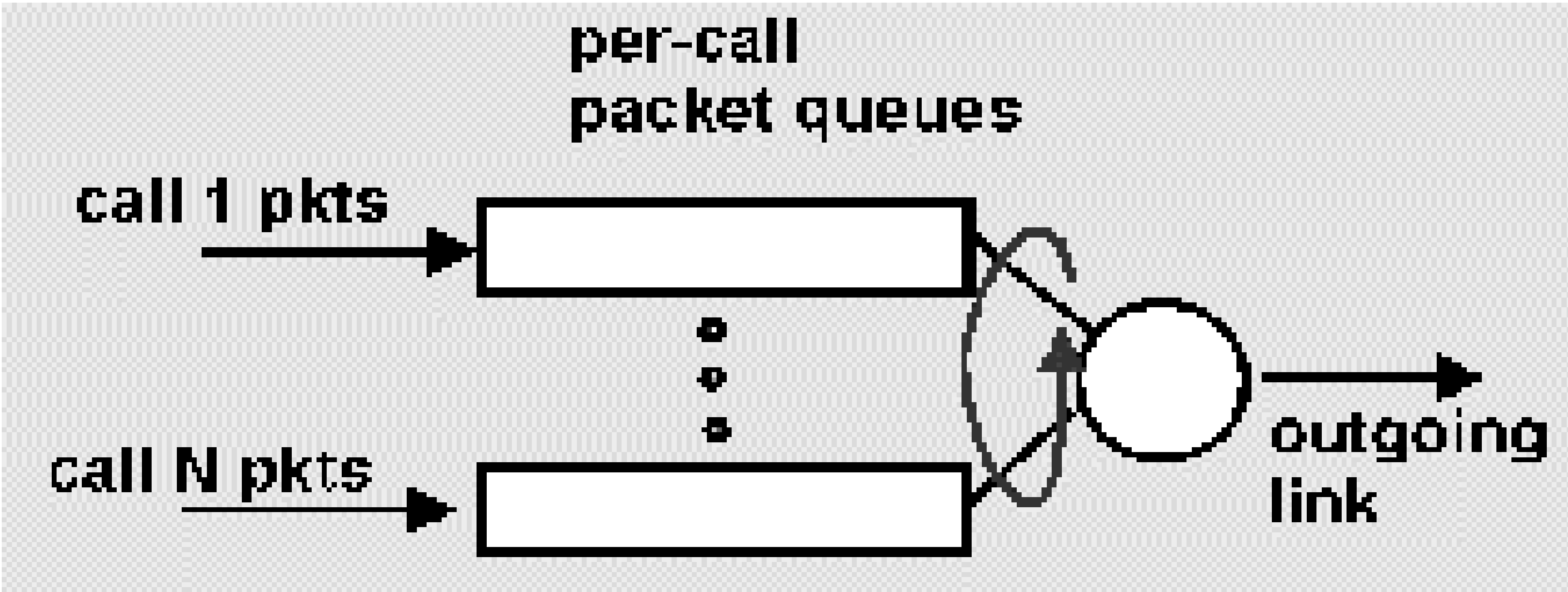
- cause of loss and delay
- packet scheduling discipline, buffer management will determine loss, delay seen by a call
  - FCFS
  - Weighted Fair Queuing (WFQ)



Link-level Scheduling: WFQ

Conceptual view:





Round-robin service:

- assume fixed length pkts, N sessions
- session i gets to send 1 pkt each "turn"
- if session i has no pkt, i+1 gets chance

WFQ: Nice Features

Fair share: every session gets minimum amount of guaranteed bandwidth

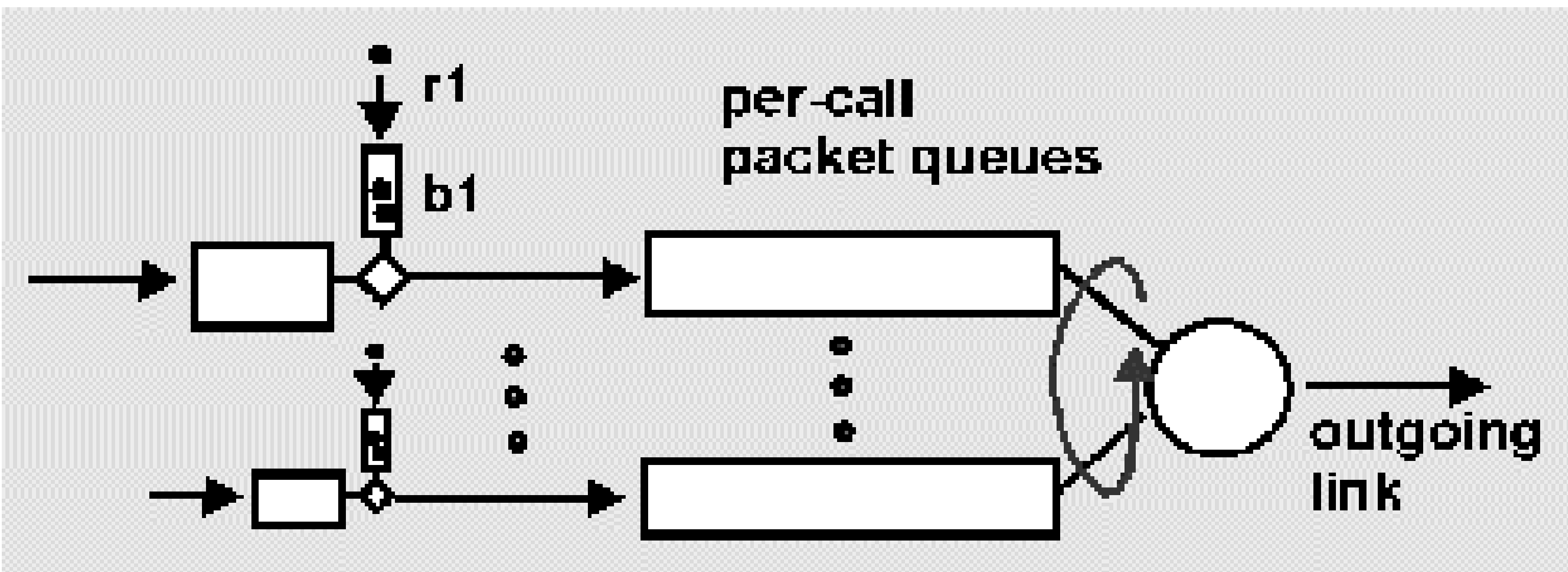
- equal share: 1/N of outgoing link capacity, C
- proportional share: each call i has number f_i
- i's share of C: f_i/ sum(all j with queued data, f_j)

Protection: WFQ *separates* handling of different sessions

- misbehaving or greedy session only punishes itself
- compare with consequences of greedy session under global FCFS!

WFQ + token bucket = delay guarantees

Simple scenario: leaky bucket controlled sources feeding WFQ multiplexor



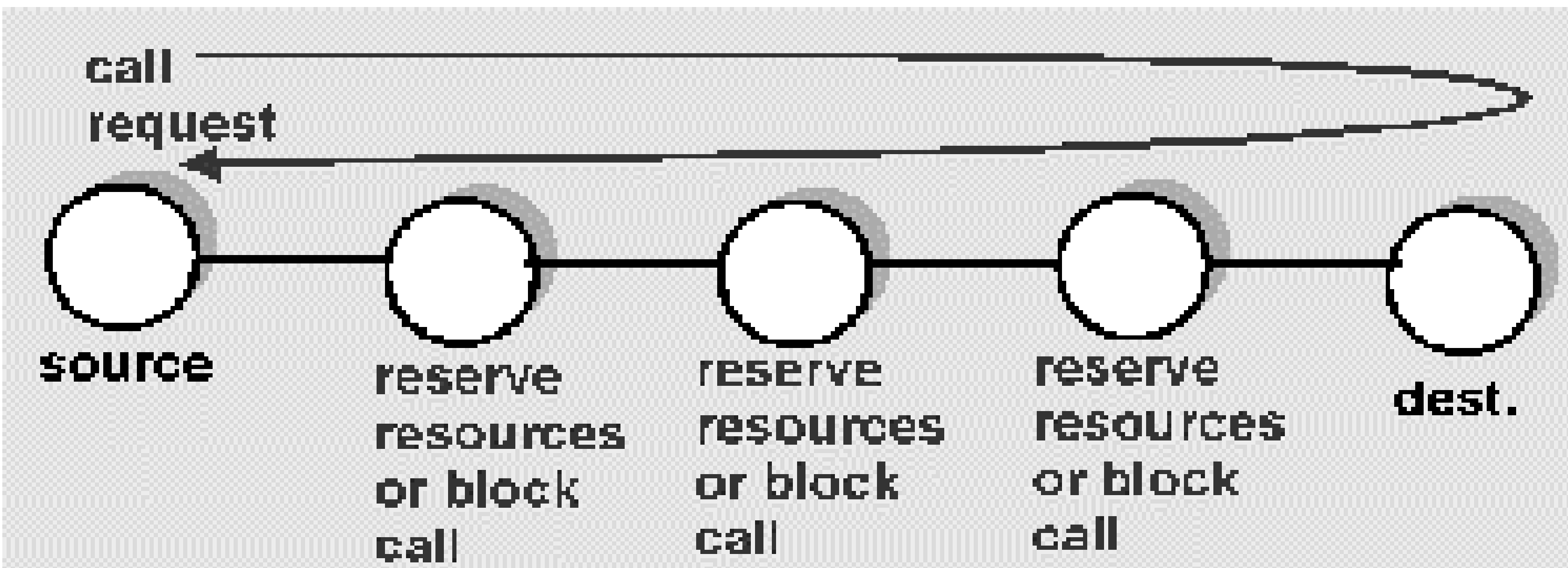
Recall: f_1/sum(all j, f_j) * C: minimum guaranteed bandwidth for session 1

- amount of session 1 traffic < r_1*t + b_1
- burst of size b_1 arrives, empties bucket, enters queue 1
- time until last packet served (maximum pkt delay) is b_1 / (f_1 C)
- queue length decreases from b_1 (assuming r_1 < f_1/sum(all j, f_j) * C)

Analysis can be extended to networks of multiplexors

Reserving Resources

- call setup protocol needed to perform call admission, reserve resources at each router on end-end path



- Q.2931: ATM call setup protocol
- sender initiates call setup by passing Call Setup Message across UNI boundary (i.e., into network)
- network immediately returns Call Proceeding indication back
- network must:
  - choose path (routing)
  - allocate resources along path (note: QoS and routing coupling)
- network returns success/failure connection indication to sender

RSVP: Internet Resource Reservation Protocol

Considerations for an Internet reservation protocol:

- multicast as a "first-class citizen"
  - large numbers of heterogeneous receivers
  - heterogeneity in available bandwidth to receiver
  - heterogeneity in receiver QoS demands (differing delay requirements)

Receiver-orientation: let receivers drive reservation process

- scalability
- heterogeneity

Soft state: call state (reservations) in routers need not be explicitly deleted

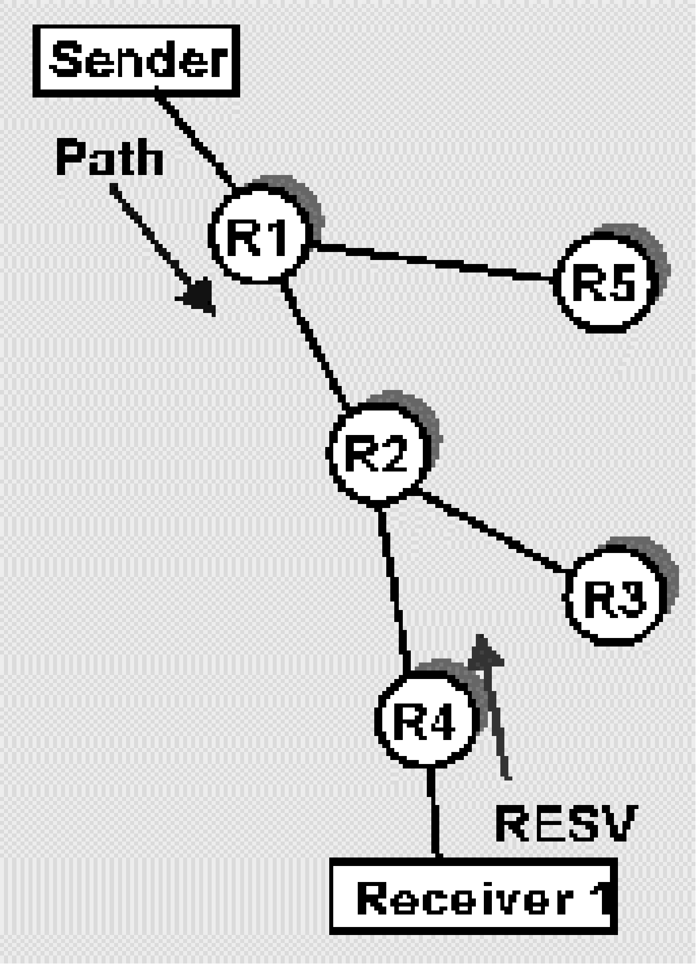
- will go away if not "refreshed" by receivers

Call Setup in RSVP

Sender sends Tspec out on multicast tree in PATH message

Receiver sends RESV message back up tree:

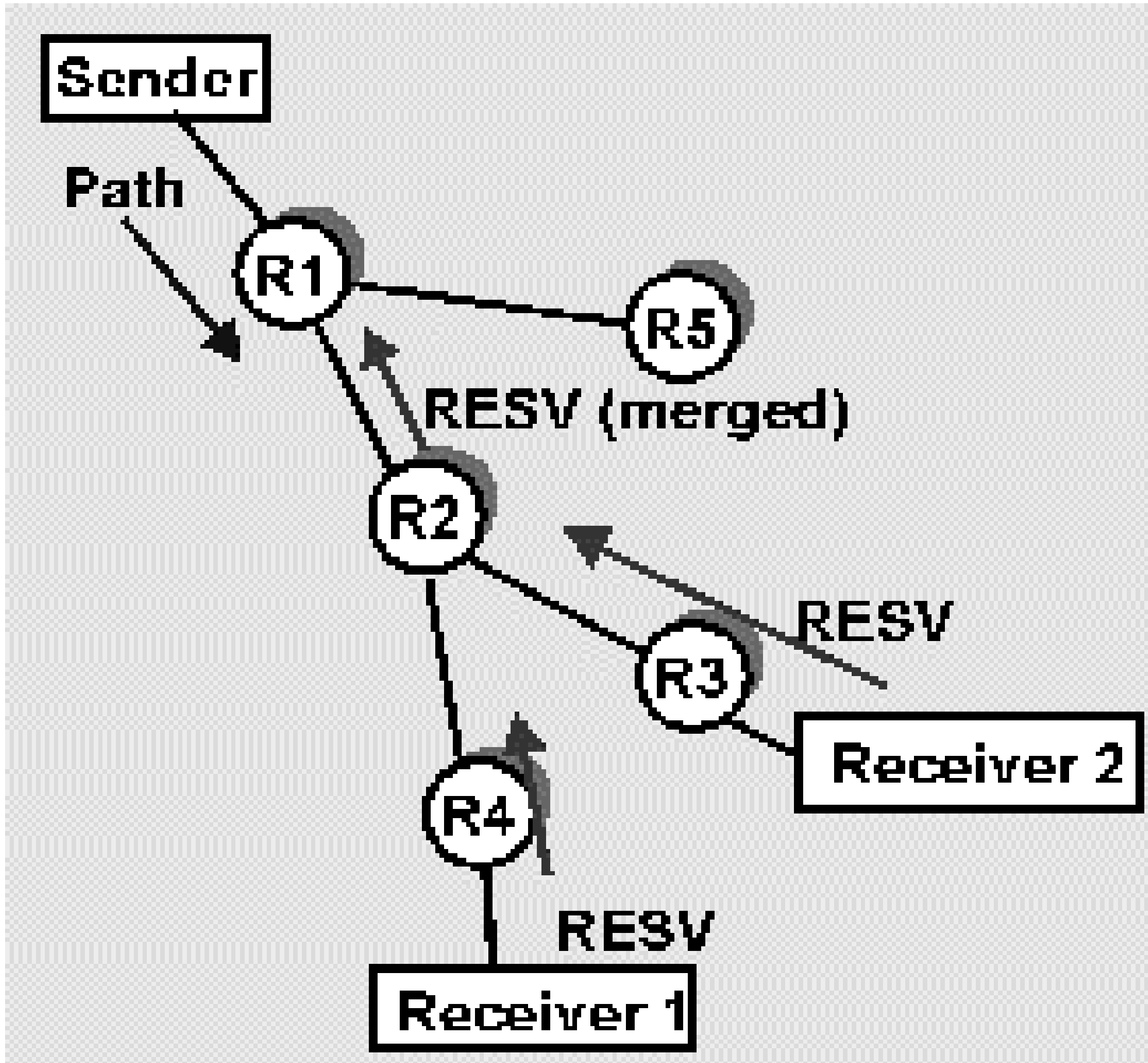
- contains senders Tspec and receiver QoS requirement (Rspec)
- routers along reverse path reserve resources needed to satisfy receiver's QoS



RSVP: Multicast

Multiple receivers:

- may have different QoS requirements
- resource reservations merged as RESV travels upstream
- resources must be allocated to satisfy strictest demands of downstream receivers
  - e.g.: receiver 1 first reserves resources for 100 ms max delay
  - if receiver 2 QoS is 200 ms max delay, no new resources at R2, R1
  - if receiver 2 QoS is 50 ms, more resources needed at R2, R1


Sender
Path
R1
R5
RESV (merged)
R2
RESV
R3
Receiver 2
R4
RESV
Receiver 1

Can handle multiple senders as well: different "styles" of resource reservation

- e.g., reserve enough resources in case all senders simultaneous
- resource enough resources for two simultaneous senders
- can dynamically determine which of N streams is forwarded downstream (switching within the network)