Rate Control in RPR – Lessons Learned from Java Simulations

Stein Gjessing
Simula Research Laboratory

Contents

- Granularity of rate control
  - Fine grained vs bursty
- 1. Simple version
- 2. Multiple (N) VOQs / choke points version
  - when N+1 is the number of stations

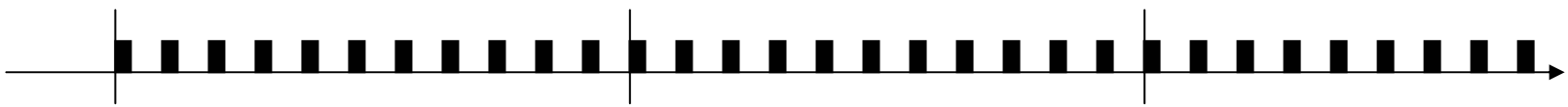
Bursty rate control

- Gandalf today:
  - OK to send if (**myUsage < allowUsage**)
  - myUsage is increased when packet sent
- Result, e.g.:

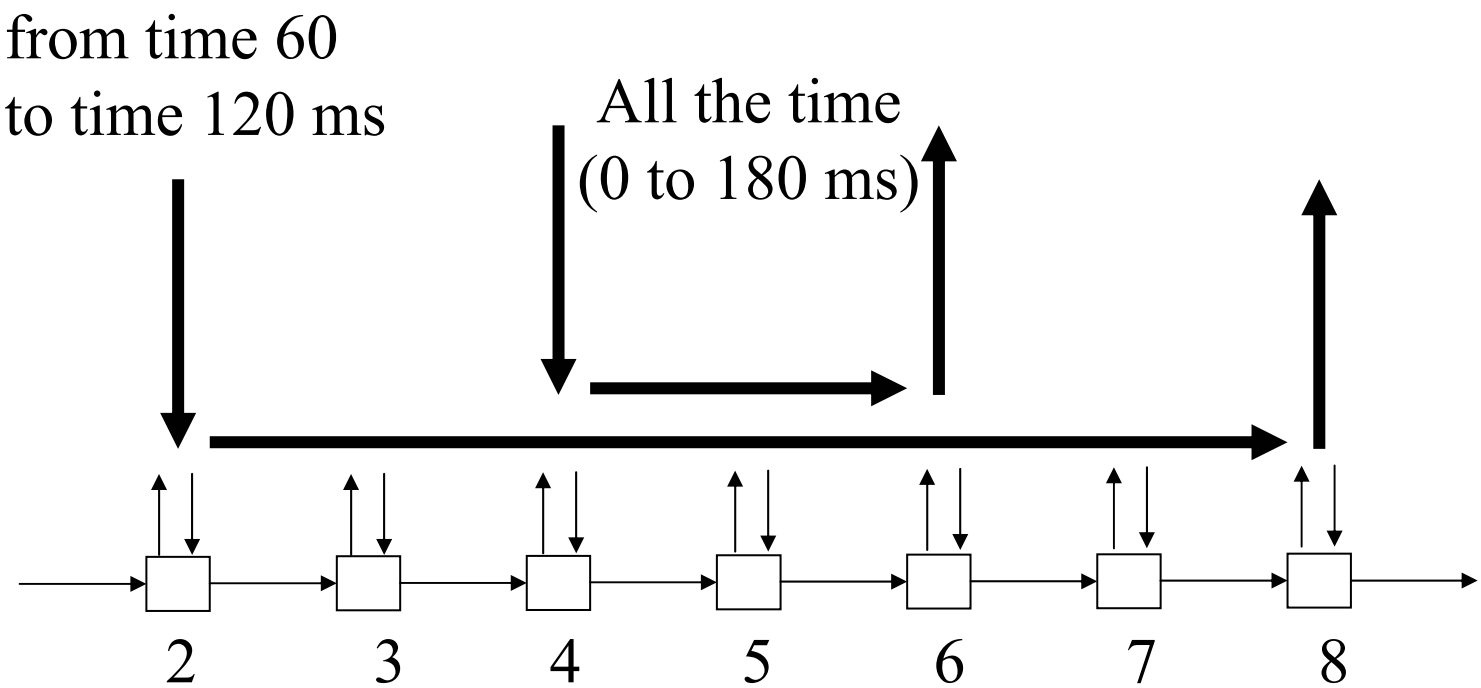
  100 microsec intervals:

Fine grained rate control

- My new Java code:
  - Send at all packets at **allowUsage / maxRate** rate
- Result:



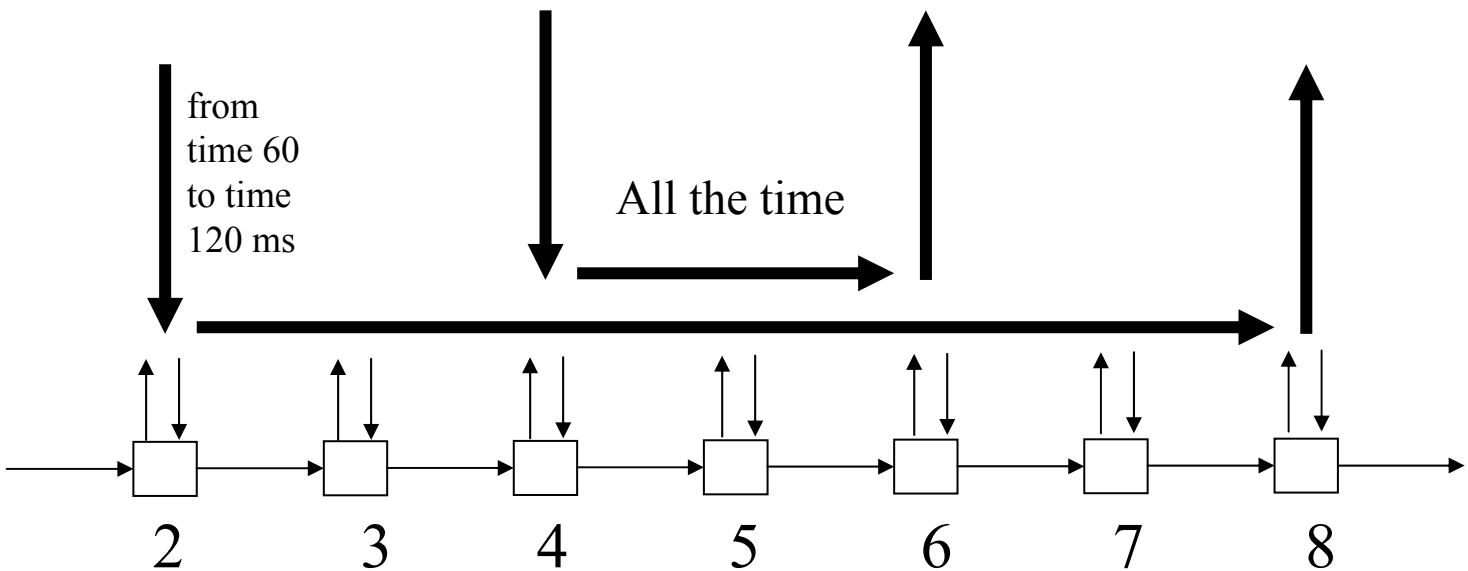
Experiment – new upstream flow



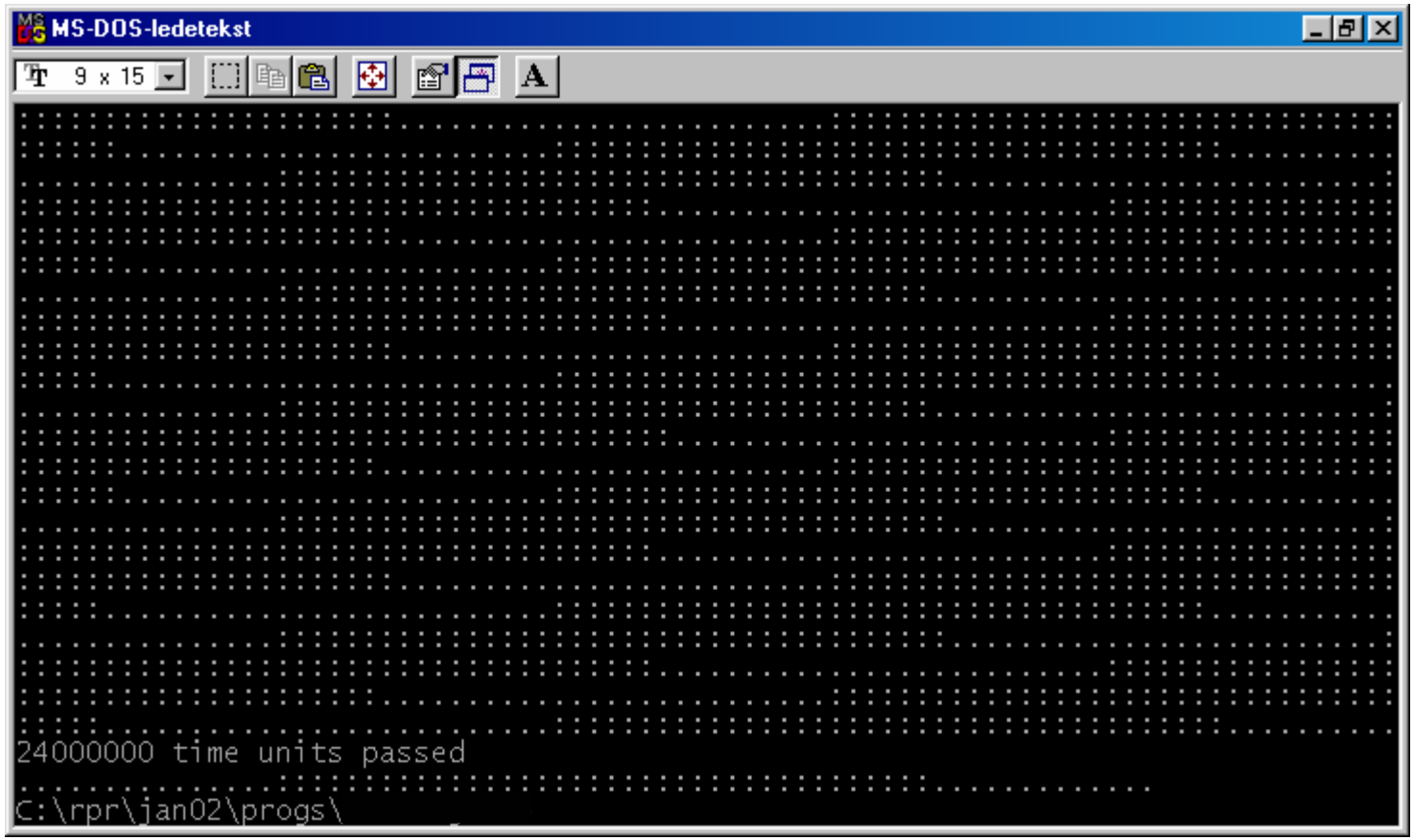
When the flow from 2 starts, it will congest station 4.
Station 4 sends upstream congestion notification to station 2.

Simulation results

- Sequence of packets passing station 5
  - 500 bytes packets
- Number of packets received per unit time (100 microsec) at stations 6 and 8



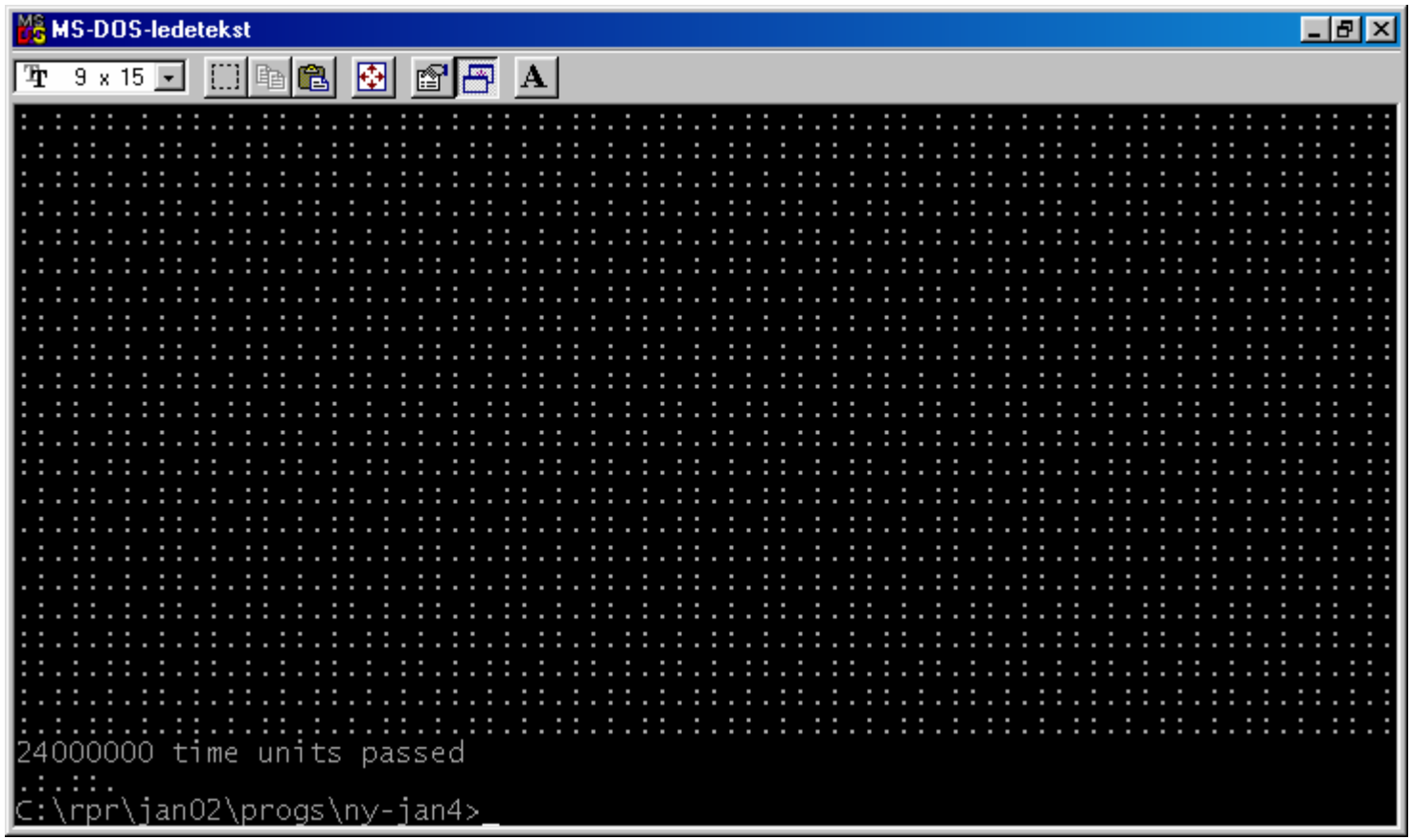
Packets passing station 5 – bursty rate control



: packet from station 2

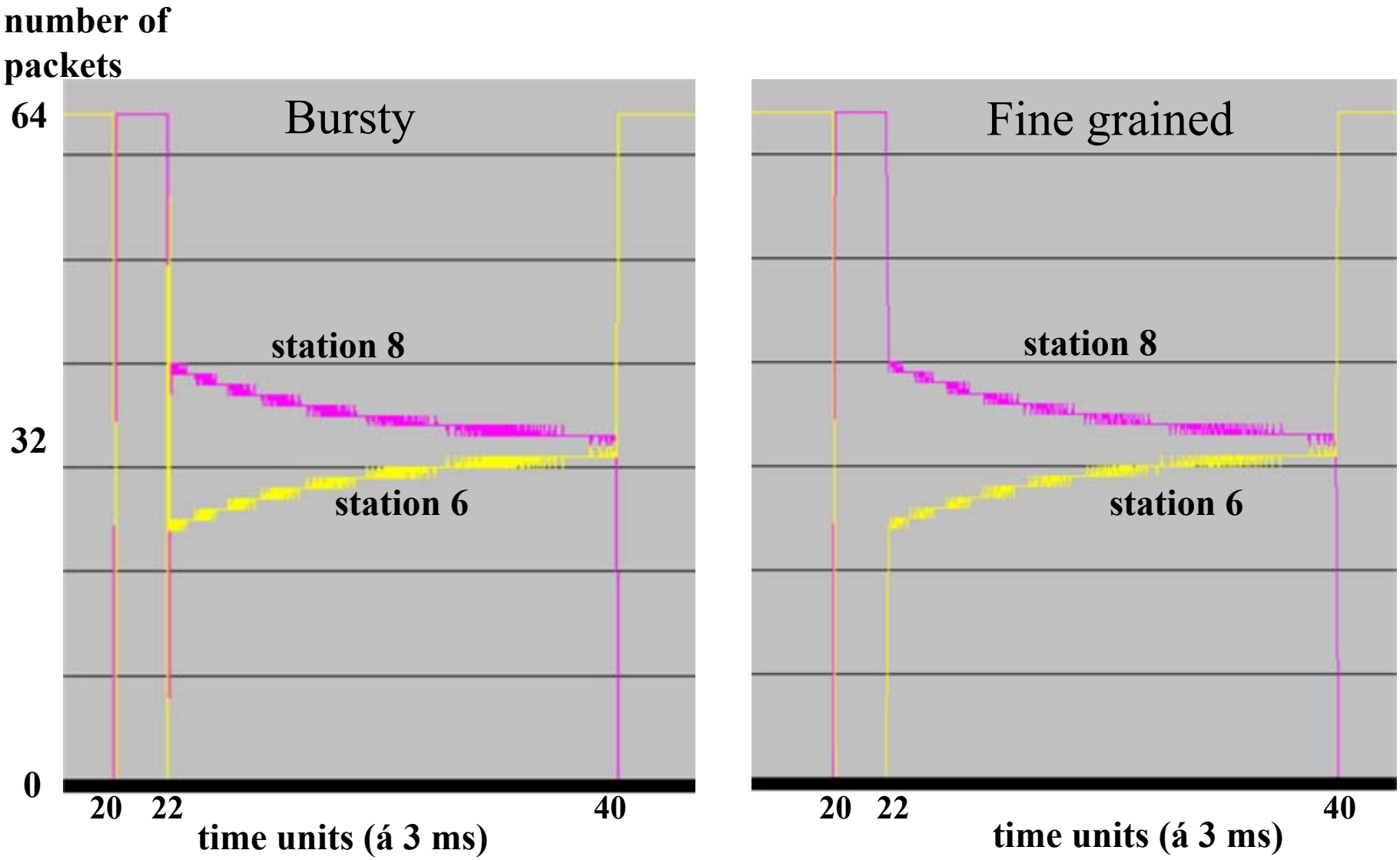
. packet from station 4

Packets passing station 5 – with fine grained rate control

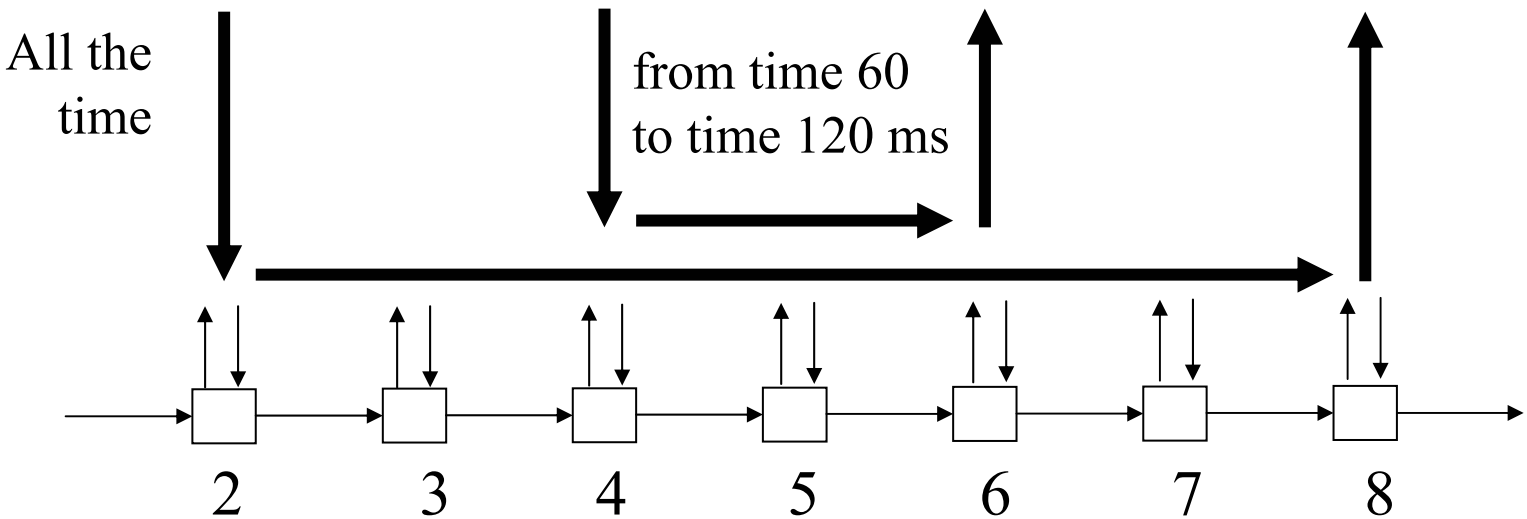


: packet from station 2 . packet from station 4

Number of packets received per 100 microsec.



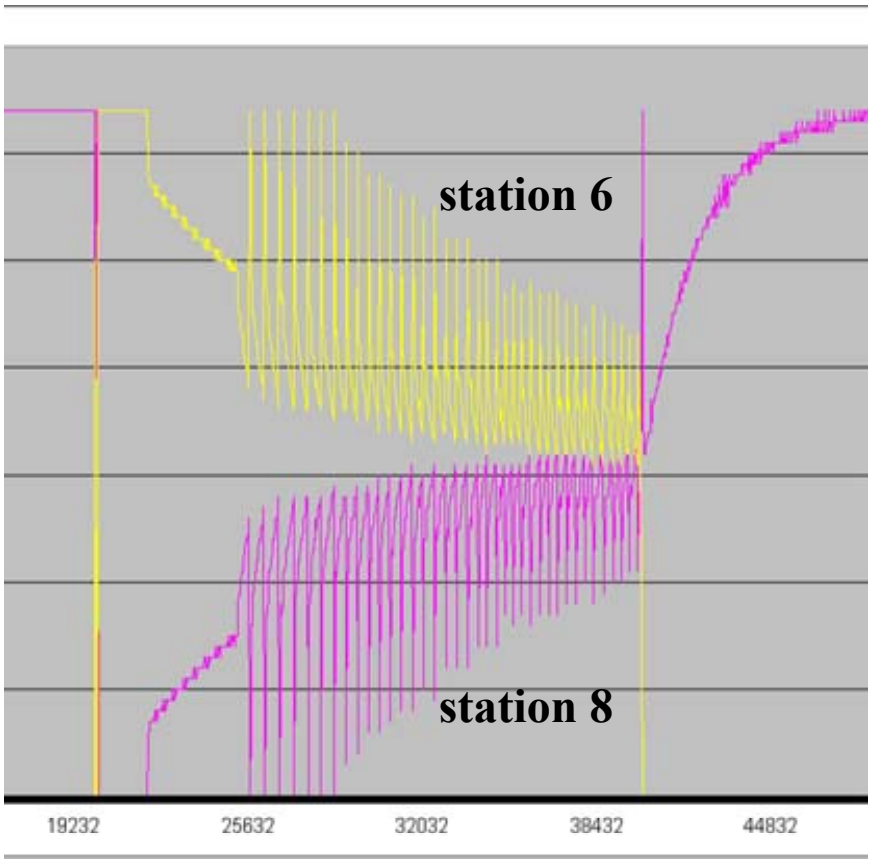
New experiment - new downstream flow



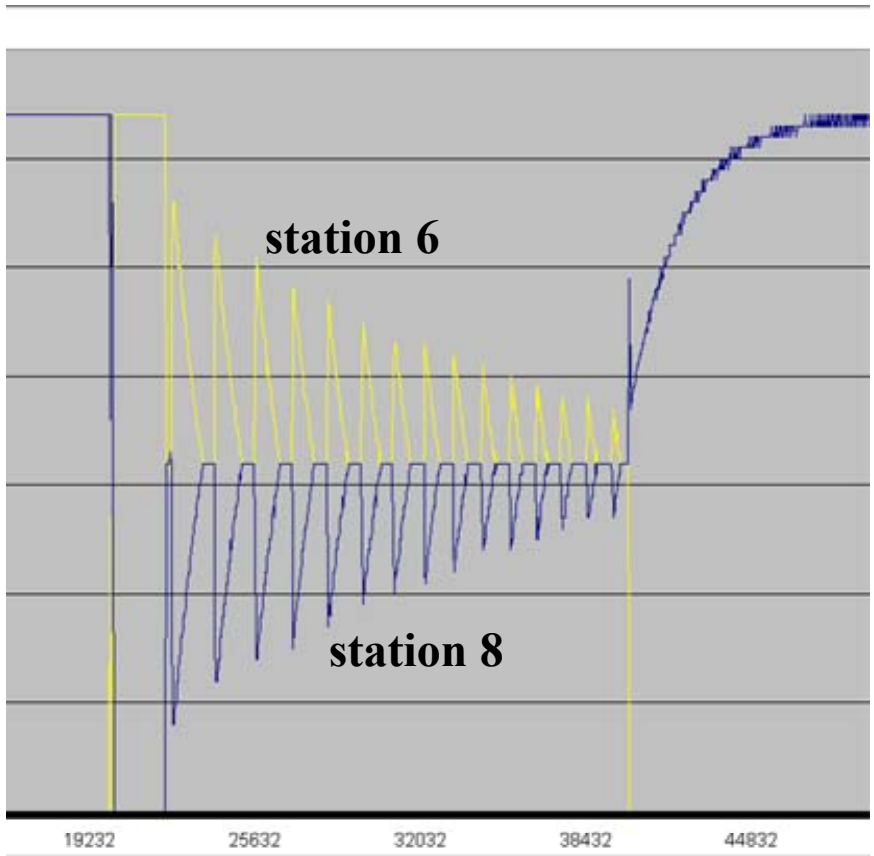
When the flow from 4 starts, station 4 will be congested.
Station 4 sends upstream congestion notification to station 2.

- Same pattern as before for packets passing by staion 5 (not shown)

New downstream flow



Bursty rate control



Fine grained rate control

Fine grained rate control makes congestion discovery more precise!

Choke points and fine grained rate control

- I have implemented N VOQ's with N possible choke points and fine grained rate control
  - (The full ring is N+1 stations)
- The allowed rate at each downstream link is
  allowUsage[i] / maxRate (or maxRate[i])
- No extra state needed for fine grained rate control (except whatever is needed in order to send smoothly at this rate)

VOQ's, Choke Points and fine grained rate control

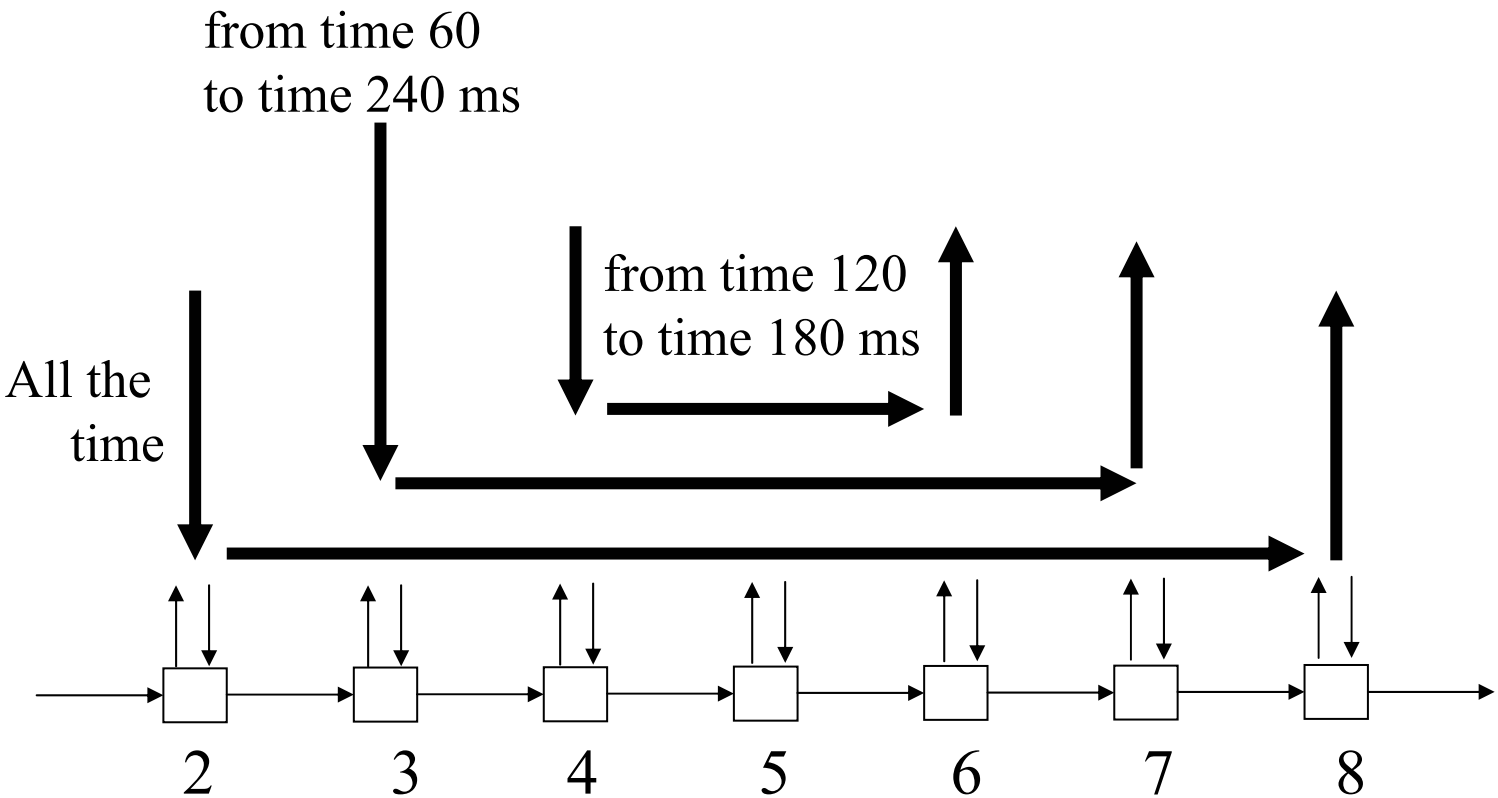
Two different implementations possible:

- Client notifies MAC on the shortest distance it wants to send a packet, and MAC notifies client when the rate counters have reached 0 for all stations up to the chosen destination station (implementation easier than it sounds because needs to keep track of the link with the minimum rate only)
  or
- MAC notifies clients whenever this station is allowed to send a packet to a new destination further downstream. Several notifications may then be sent to the Client with increasing distance to receiver.

New experiment

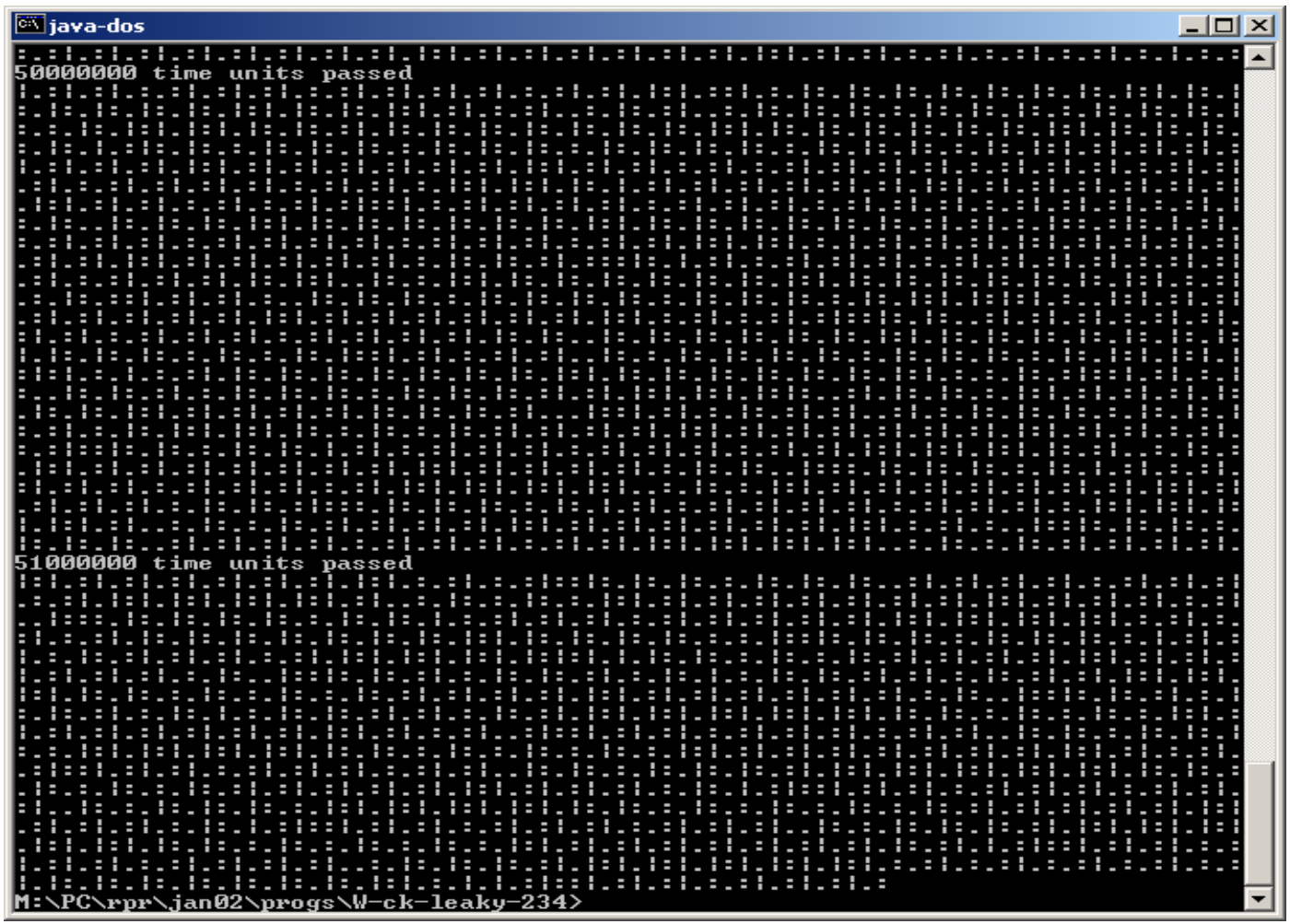
- Three flows (500 byte packets)
  - 2 to 8 sends all the time
  - 3 to 7 sends from time 60 to 240 ms.
  - 4 to 6 sends from time 120 to 180 ms.
- Station 2 will experience an increase in number of choke points and their rate
- However in this example VOQ's are not used (only choke points)

New experiment – three flows



Packet passings station 5

Fine grained rate control does not give bursts

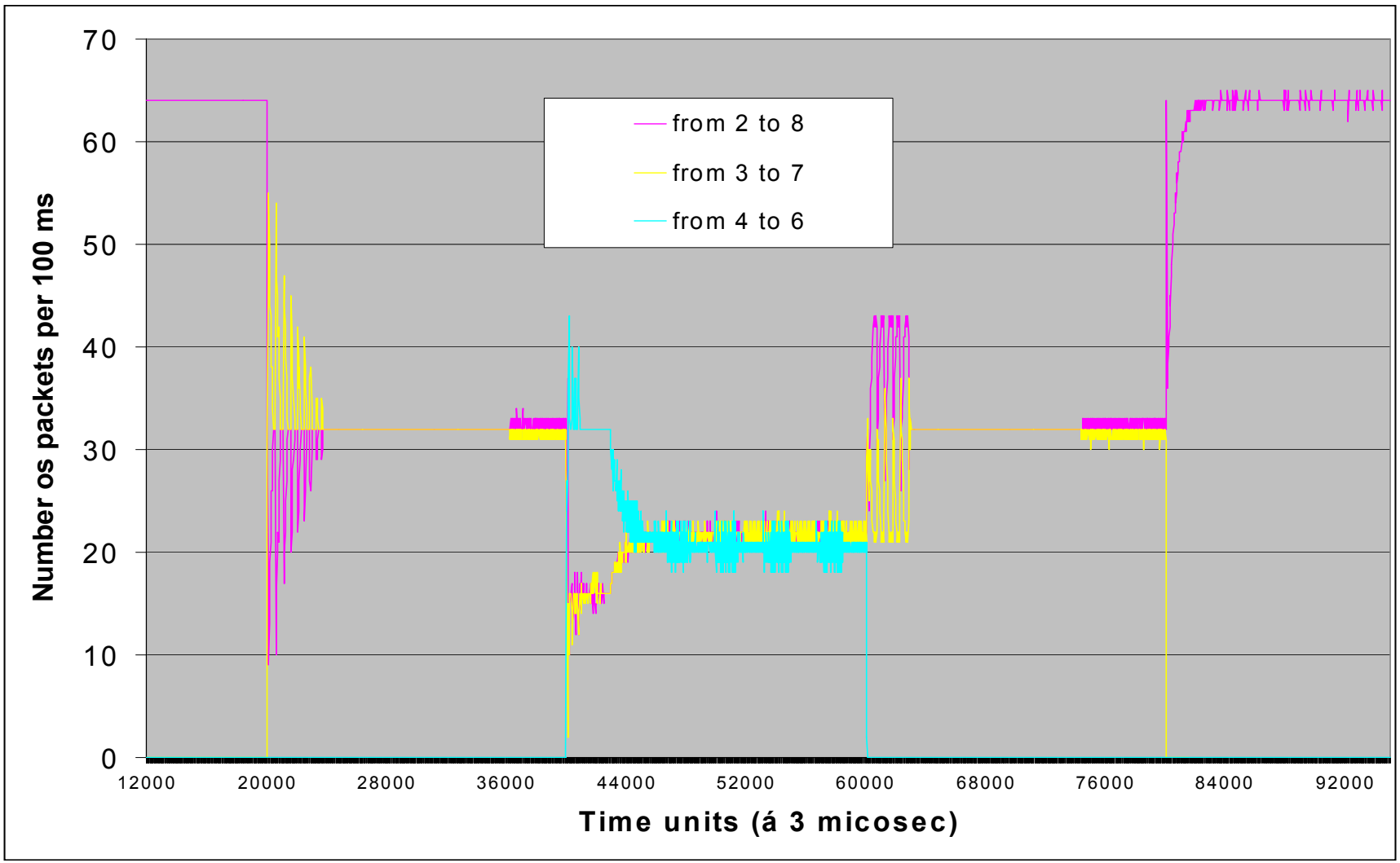


- from 2
- from 3
- from 4

2.5 Gbit/s

One time unit is three ns.

Packets received



Conclusion

- Fine grained rate control is easy to implement
  - Also together with choke points and VOQs
  - Rate is **allowedUsage[i] / maxRate**
  - A packet has to ”obey the rates” of all links it passes
- Fine grained rate control makes congestion detection more precise
- Thesis: Fine grained rate control smooths Internet traffic and decreases buffer needs.