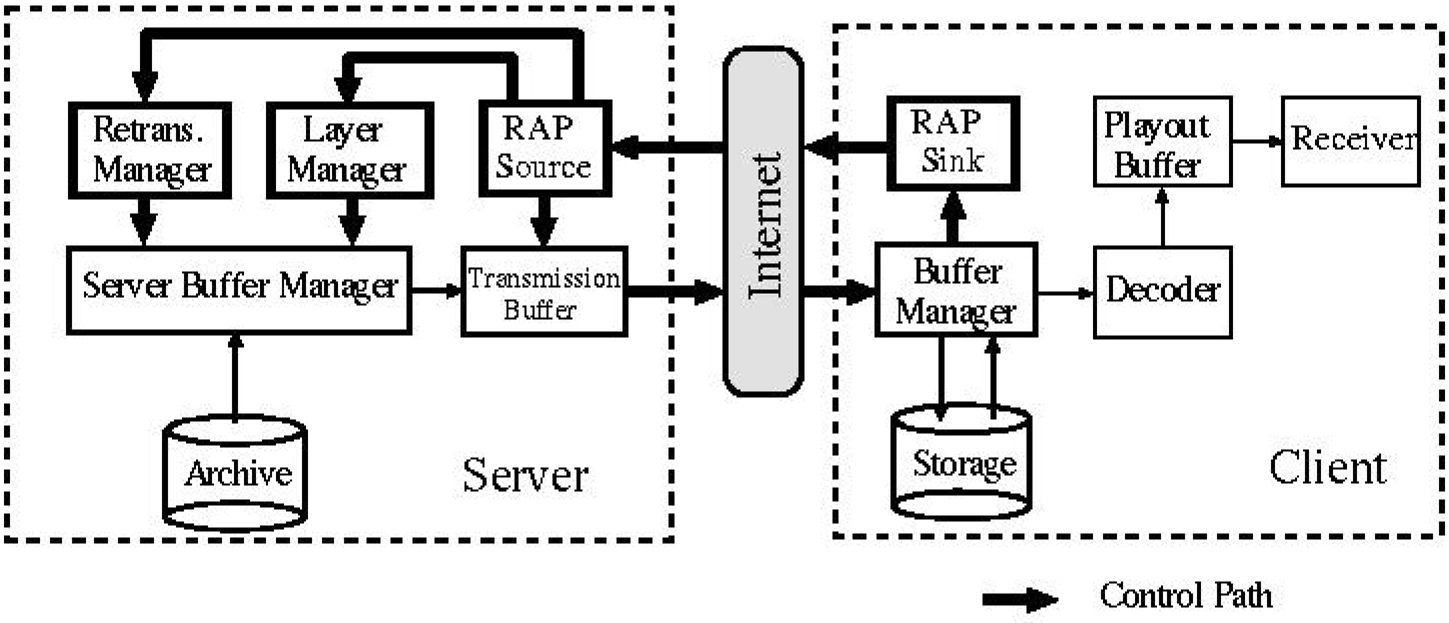
RAP: End-to-end Rate Based Control for Real Time Streams in the Internet

R. Rejaie et al, Infocom 99

- **RAP: Rate Adaptation Protocol**
- **Applications: web server or VOD server streaming**
- **Uses UDP and RTP**
- **Mimics TCP's AIMD behavior**
- **Main design goal: friendliness to TCP**



- **End to end, application level implementation**
- **Layer encoded, stored real time stream**
- **Source adapts rate by adding/removing layers based on ETE feedback**

RAP Mechanism

- Receiver individually ACKs packets
- It delivers packets to playout buffer even if received out of order
- Each ACK carries sequence #
- Sender estimates round trip time SRTT from ACKs
- Sender keeps packet timers and checks for potential timeouts

RAP (cont)

Increase/decrease (AIMD) alg:

- Rate adjusted each SRTT
- No loss: add one more packet in SRTT
- Loss detected: reduce # of packets per SRTT by ½ (like in Reno)
- "Cluster loss" (ie, many consecutive packets lost): react only to first loss – similar to TCP SACK behavior

RAP (cont)

- RED (Random Early Drop) used to limit the burst loss occurrence
- RED improves TCP-friendliness: it allows TCP to catch up with RAP

Simulation experiments

NS-2

TCP flows: FTP

Resources are scaled up proportionally to # of users.

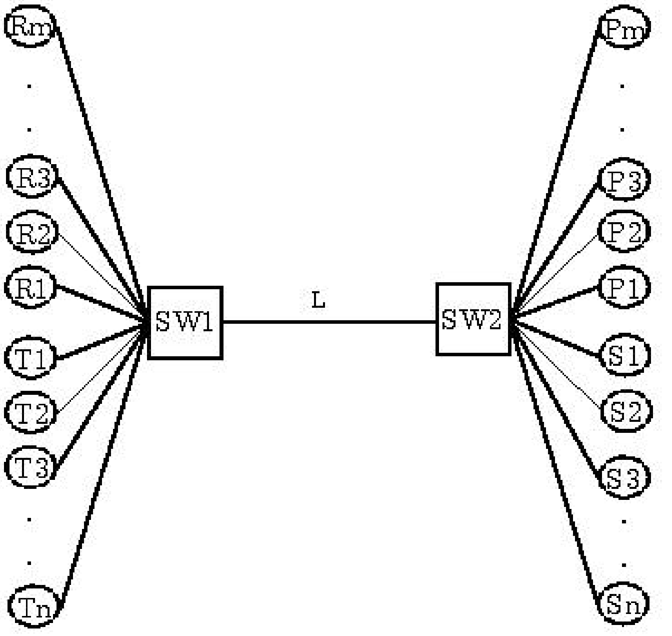
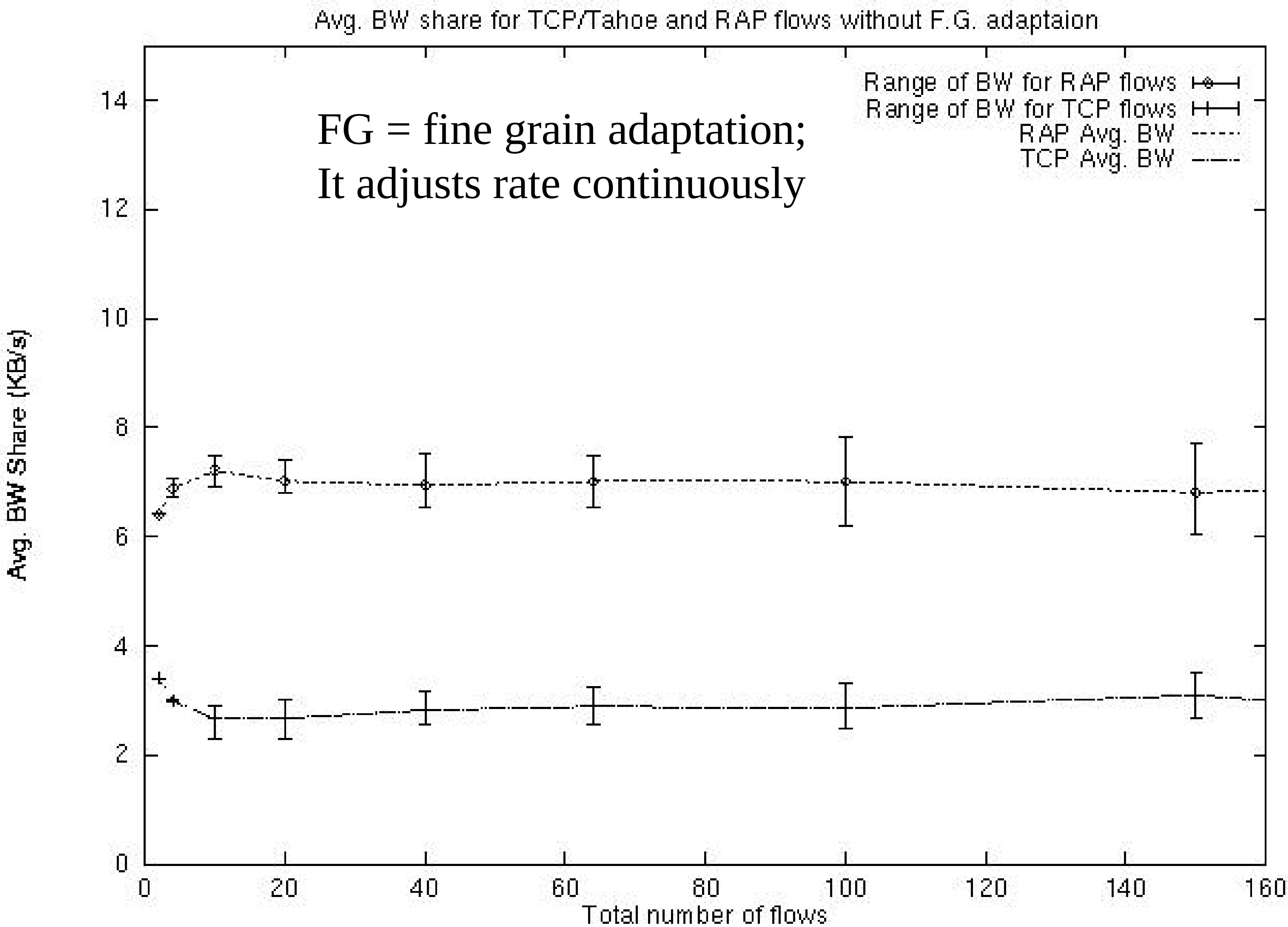


Table 1

Packet Size	100 Byte
ACK Size	40 Byte
Bottleneck Delay	20 ms
B/W per Flow	5 KByte/s
B/W of Side Links	1.25 MByte/s
Tot. Delay of Side-Links	6 ms
Simulation Length	120 sec
TCP Maximum Window	1000
TCP Timeout Granularity	100 ms

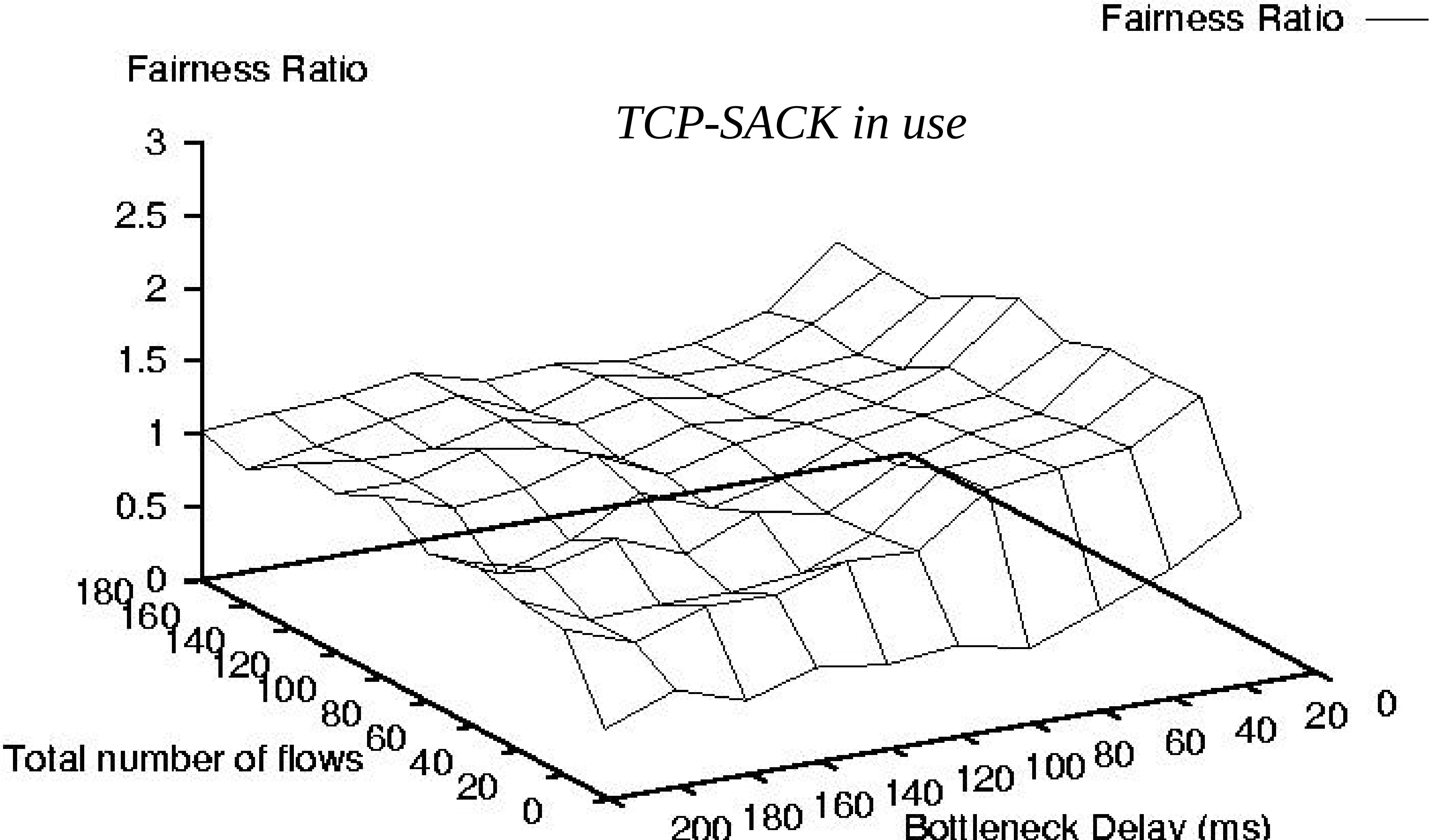

Avg. BW share for TCP/Tahoe and RAP flows without F.G. adaptaion
Range of BW for RAP flows
Range of BW for TCP flows
RAP Avg. BW
TCP Avg. BW
FG = fine grain adaptation;
It adjusts rate continuously
Avg. BW Share (KB/s)
Total number of flows
0
2
4
6
8
10
12
14
0
20
40
60
80
100
120
140
160

TCP friendliness

- RAP not very friendly to Tahoe!
- Tahoe suffers from frequent time outs and slow-start episodes
- Other TCP versions fare better
- In following experiments, TCP SACK is used

Fairness Ratio across the parameter space without F.G. adaptaion

TCP-SACK in use



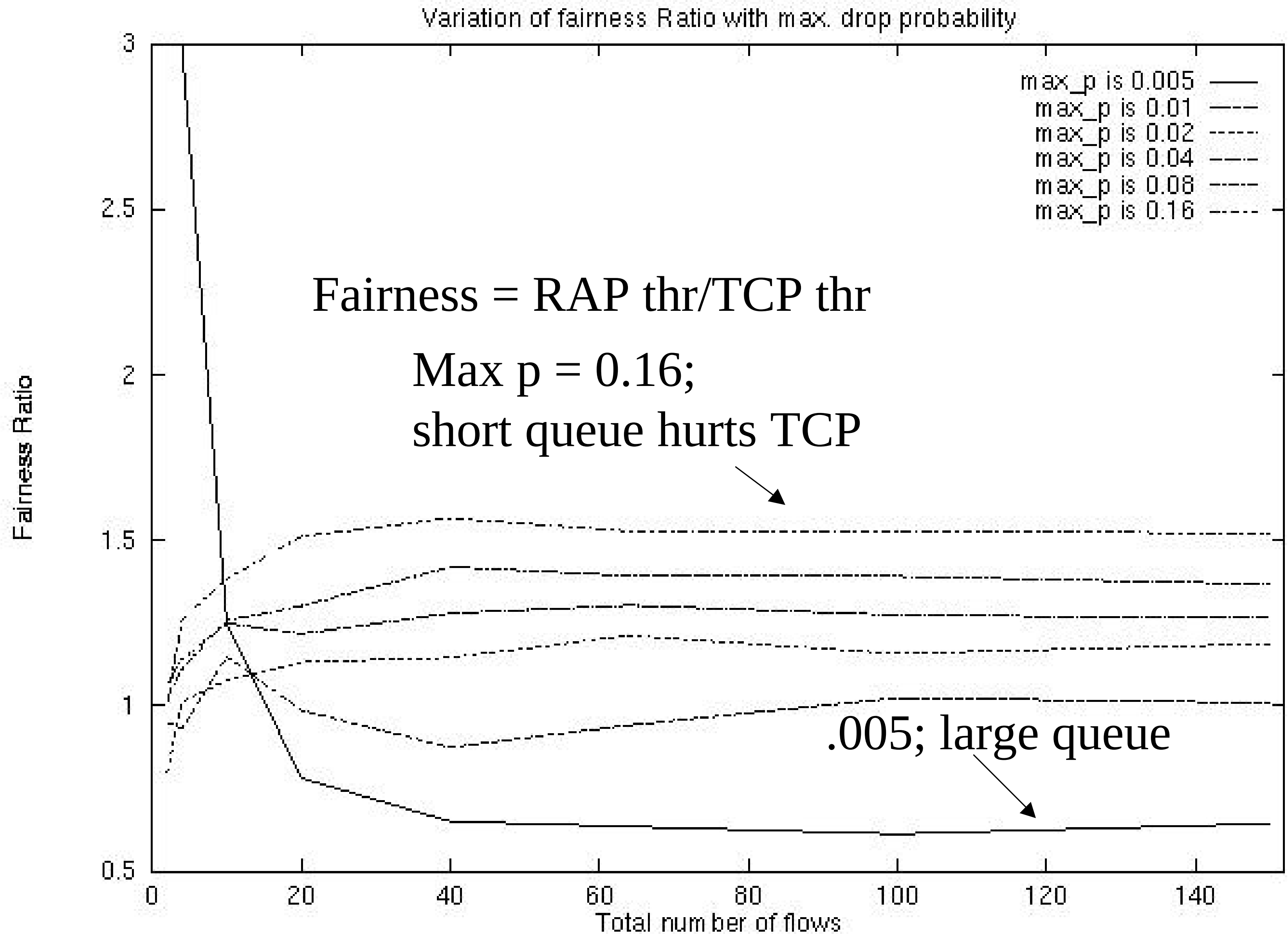
TCP friendliness

- Half flows are RAP, half are TCP
- Fairness ratio = avg RAP thrpt/avg TCP thrpt
- RAP fair to TCP SACK (except for small # of flows and small round trip delays)

RED Gateways

Table 2

Min. Threshold	5 Packets
Max. Threshold	0.5 * Buffer
Bottleneck B/W	5 KByte/s * No. of Flows
Bottleneck Delay	20 ms
Buffer Size	12 * RTT * Bottleneck B/W
q_weight	0.002

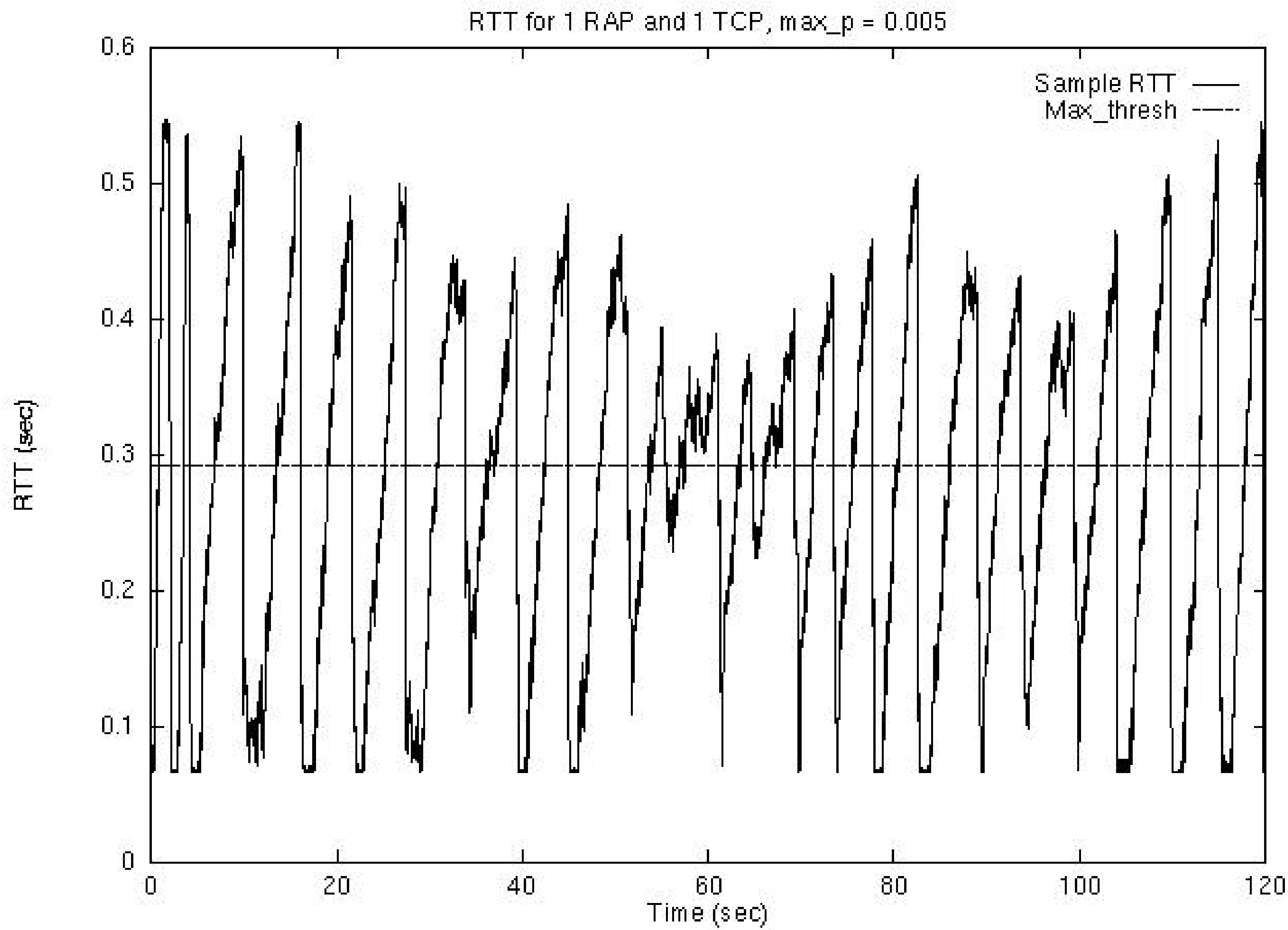

Variation of fairness Ratio with max. drop probability
max_p is 0.005
max_p is 0.01
max_p is 0.02
max_p is 0.04
max_p is 0.08
max_p is 0.16
Fairness = RAP thr/TCP thr
Max p = 0.16;
short queue hurts TCP
.005; large queue
Fairness Ratio
Total number of flows
3
2.5
2
1.5
1
0.5
0
20
40
60
80
100
120
140

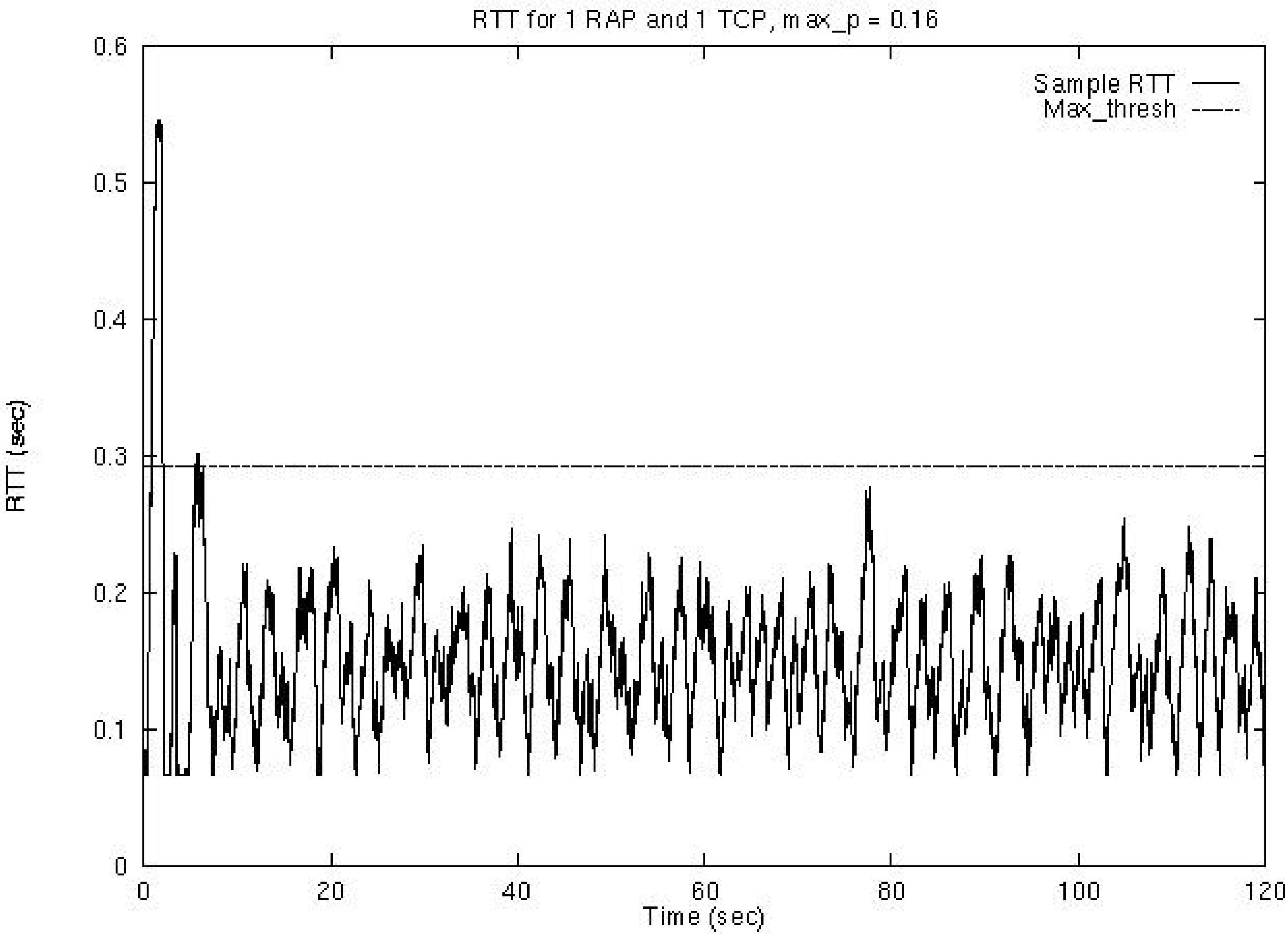
RAP behavior with RED

If max p is too small (eg .005):

Avg queue size grows large, packets are tail-dropped; system has large queue fluctuations

- With small # of flows, the period is large; TCP recovers less easily than RAP
- With large # of flows, the period is shorter; TCP flows are hit less than the evenly spaced RAP flows; RAP performs poorly!


RTT for 1 RAP and 1 TCP, max_p = 0.005
Sample RTT
Max_thresh
RTT (sec)
Time (sec)
0
0.1
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100
120


RTT for 1 RAP and 1 TCP, max_p = 0.16
Sample RTT
Max_thresh
RTT (sec)
Time (sec)
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20
40
60
80
100
120

RAP behavior with RED (cont)

- As max p is increased, more packets are random dropped and the queue becomes more stable
- Buffer utilization (and throughput) are lower
- TCP congestion window becomes small, and RAP takes advantage of it, “stealing’ the avail bandwidth from TCP flows

Conclusions

- RAP is reasonably TCP friendly in large scale nets (many flows) and large windows
- With a few flows, the scheme becomes unfair
- RED improves fairness; but parameters must be properly tuned; bed RED param selection (eg, max p = .005) can harm real time traffic!