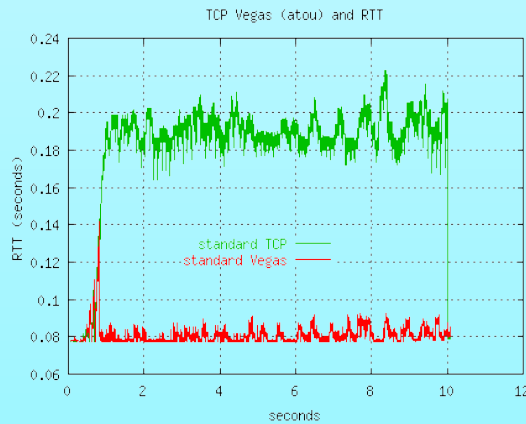


TCP Reno vs. Vegas



INFOCOM 1999

Paper by:

Jeonghoon Mo
Richard La
Venkat Anantharam
Jean Walrand

Discussion Leader:
Jack Meier

Presented by:

Manfred Georg

1

Overview

- TCP Reno
- TCP Vegas
- Fairness
- Simulation
- Conclusion

TCP Reno

- Slow Start
- Congestion Avoidance (AIMD)
- Packet Loss
- Window size and Network Buffering

TCP Vegas

- Slow Start
- Congestion Avoidance
 - Round Trip Delay
 - Alpha and Beta
- Rerouting
- Persistent Congestion

Vegas Algorithm

- $\text{Expected} = \text{CWND} / \text{BaseRTT}$
- $\text{Actual} = \text{CWND} / \text{RTT}$
- $\text{Diff} = (\text{Expected} - \text{Actual}) \text{BaseRTT}$

CWND = 60 packets

BaseRTT = 20 ms

RTT = 30 ms

Expected = 3000 packets/s

Actual = 2000 packets/s

Diff = 20 packets

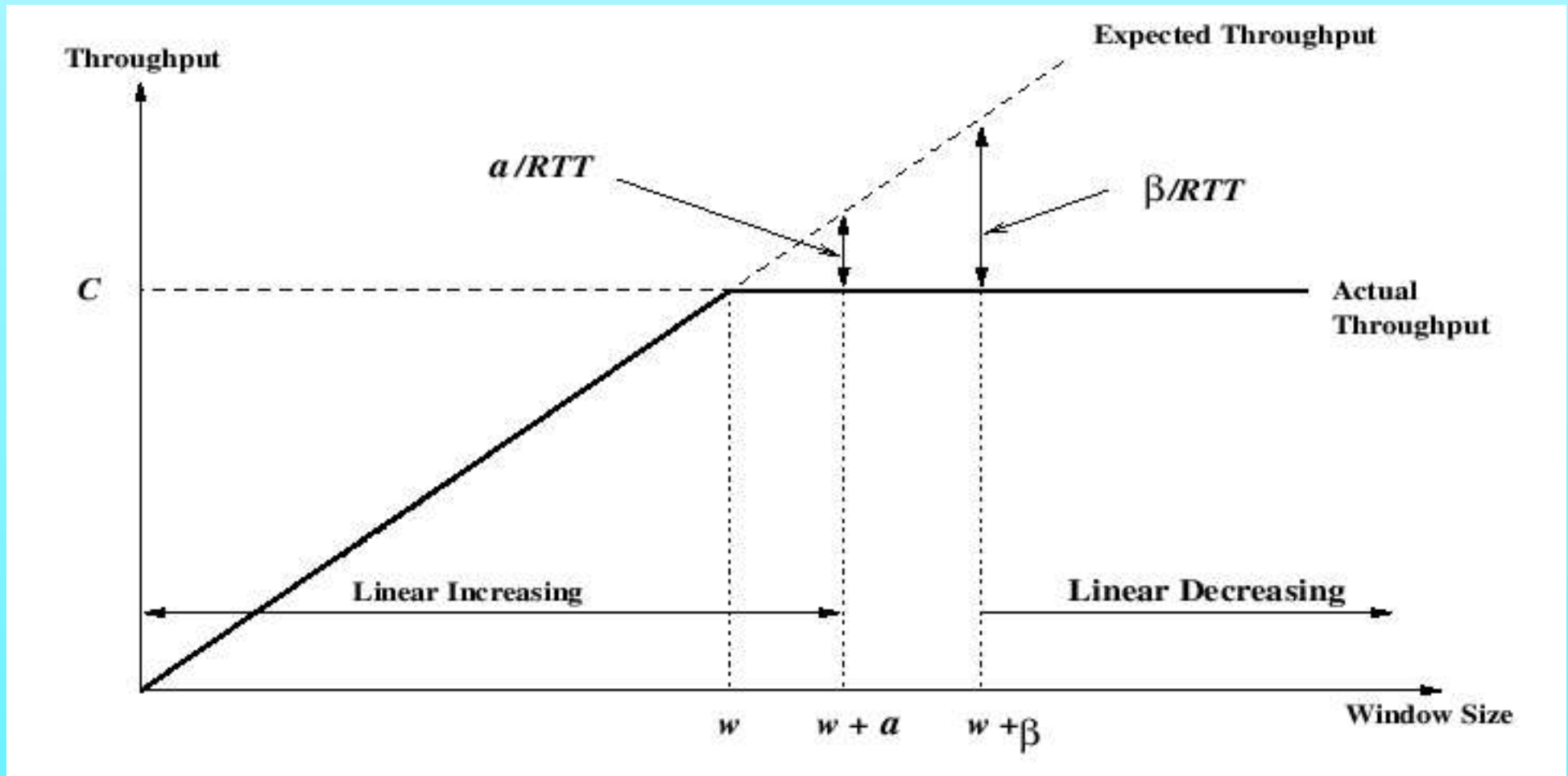
Vegas Algorithm

- $\text{Expected} = \text{CWND} / \text{BaseRTT}$
- $\text{Actual} = \text{CWND} / \text{RTT}$
- $\text{Diff} = (\text{Expected} - \text{Actual}) \text{BaseRTT}$

$$\text{CWND} = \begin{cases} \text{CWND} + 1 & \text{if Diff} < \text{alpha} \\ \text{CWND} - 1 & \text{if Diff} > \text{beta} \\ \text{CWND} & \text{otherwise} \end{cases}$$

Vegas Algorithm

- $\text{Diff} = (\text{Expected} - \text{Actual}) \text{BaseRTT}$



Rerouting Vegas

- No Explicit Feedback
- Propagation Delay Changes
 - Shorter Delay Detected
 - Longer Delay Interpreted as Congestion
- Occasionally update BaseRTT
 - BaseRTT = min RTT since last update
 - If (RTT < BaseRTT) update BaseRTT = RTT
- Cause Persistent Congestion?

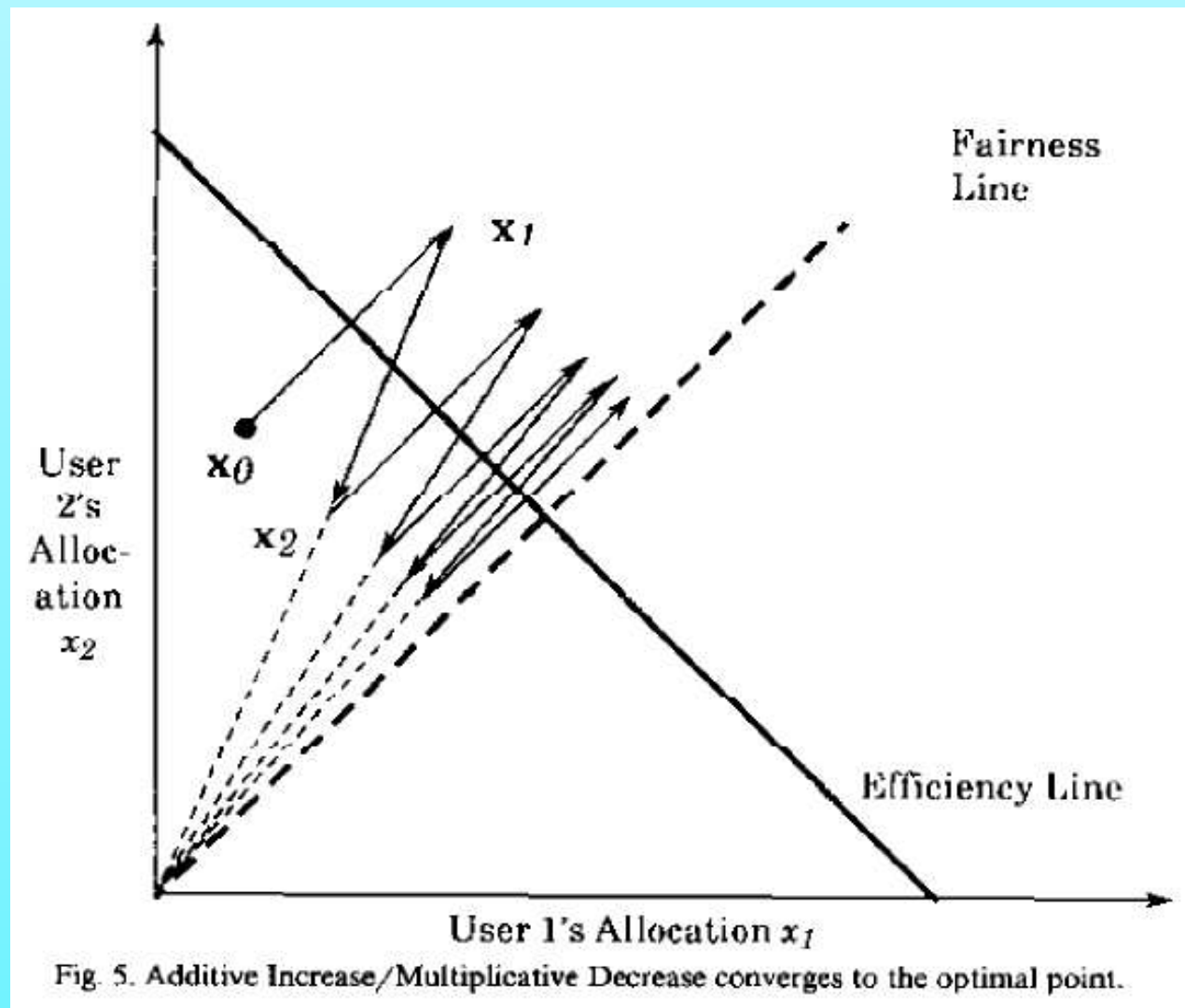
Persistent Congestion

- New Connection Increases Congestion
- BaseRTT updating
 - Temporarily causes congestion
 - Other connections back off
 - All connections get accurate estimate of BaseRTT

Fairness

- Reno vs. Reno
- Vegas vs. Vegas
- Reno vs. Vegas

Reno Fairness



D. Chiu and R. Jain,
Journal of Computer Networks and ISDN Systems, 1989

Vegas Fairness

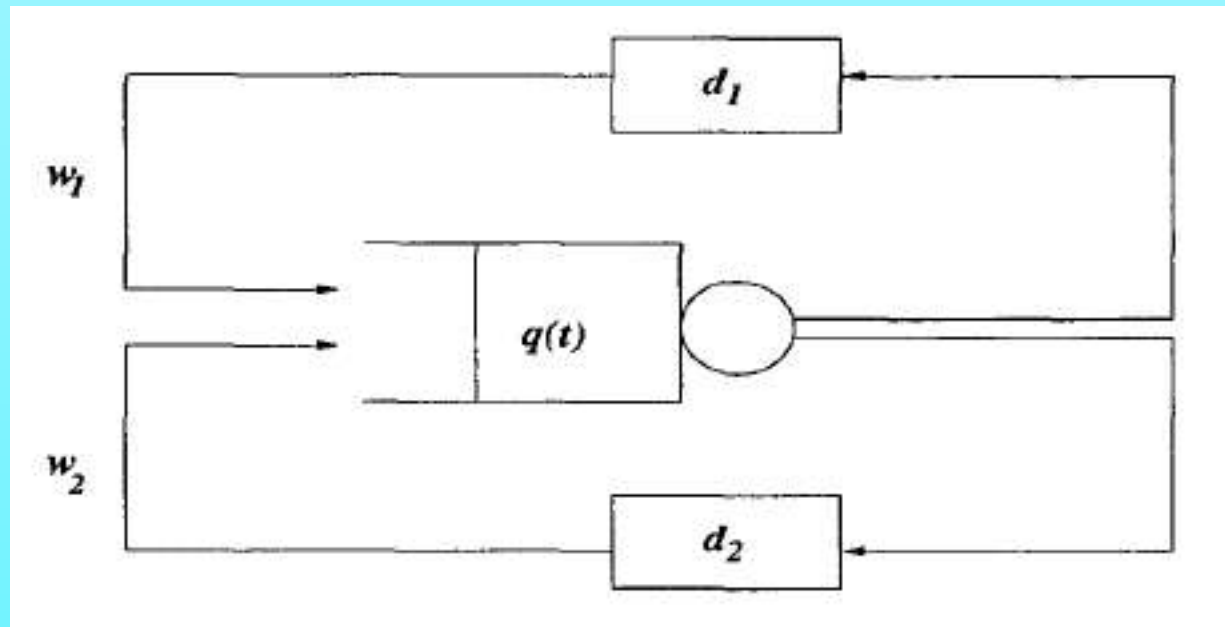
$z(t)$ = total backlog

$e_i(t)$ = ACK received rate

$q_i(t)$ = backlog in queue

$w_i(t)$ = window size

$q_i(t)$ between α and β



Vegas Fairness

$z(t)$ = total backlog

$e_i(t)$ = ACK received rate

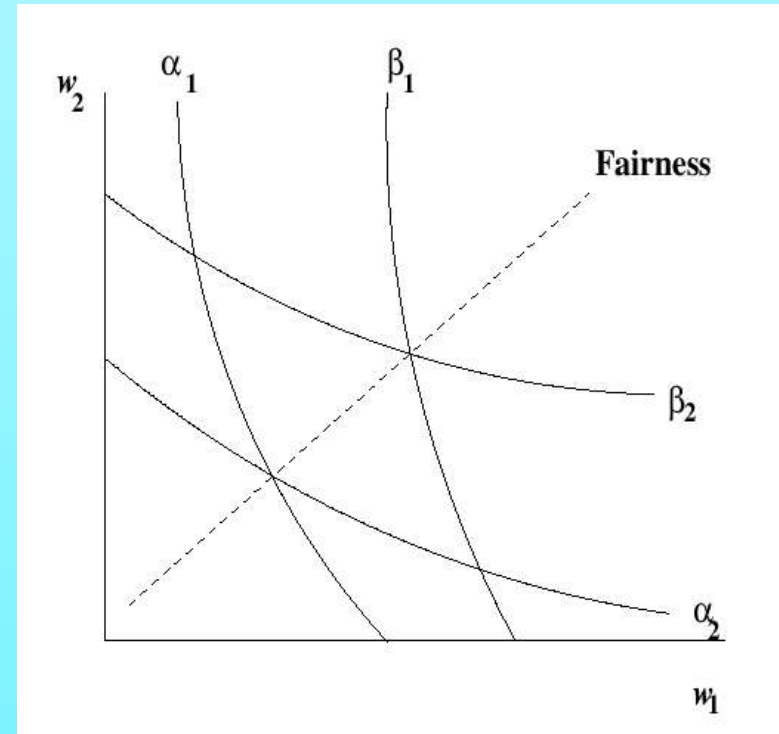
$q_i(t)$ = backlog in queue

$w_i(t)$ = window size

$$w_i(t) = q_i(t) + e_i(t) d_i$$

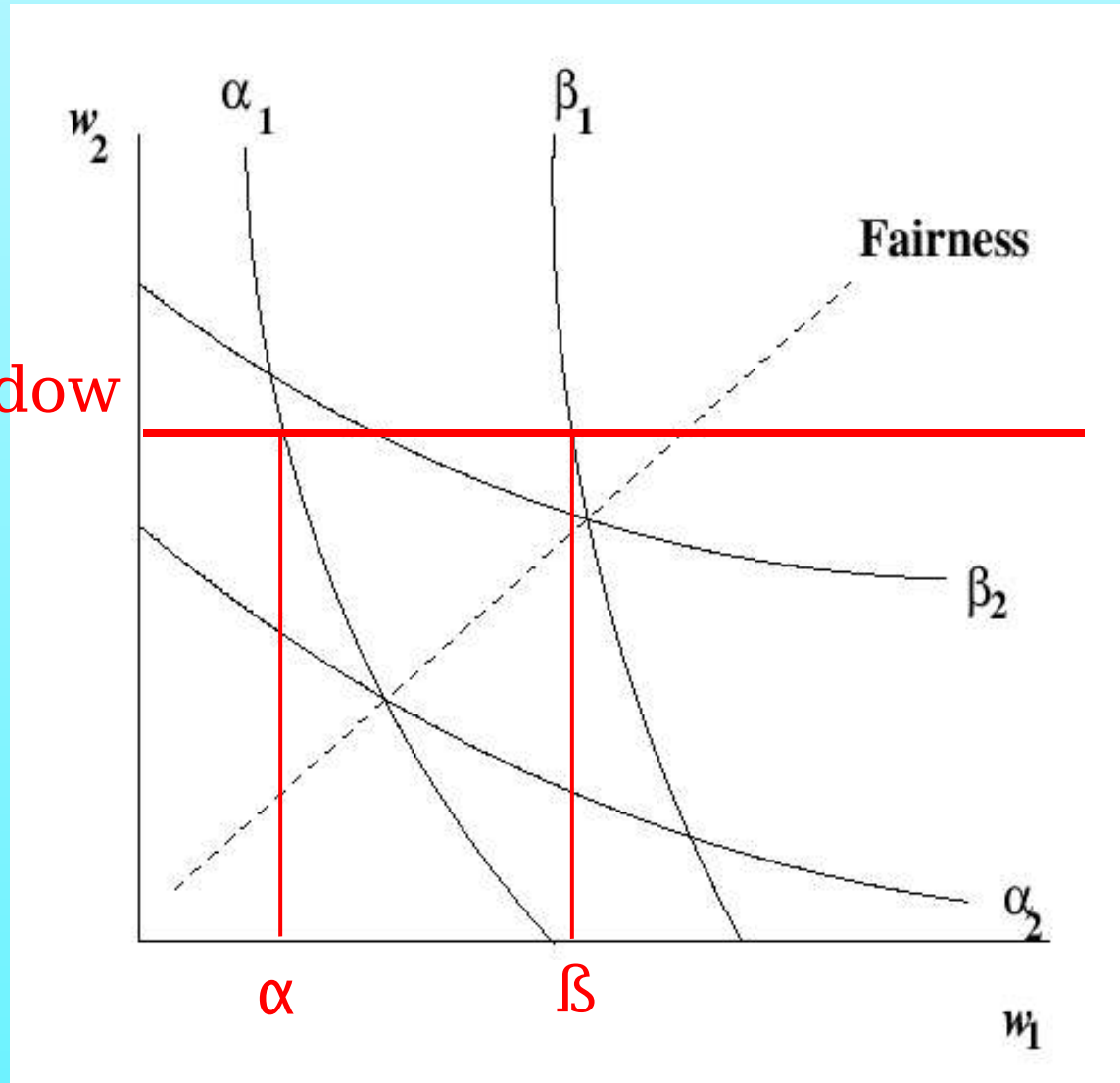
$$e_i(t) = w_i(t) / (z_i(t) + d_i)$$

$$e_1(t) + e_2(t) = 1$$

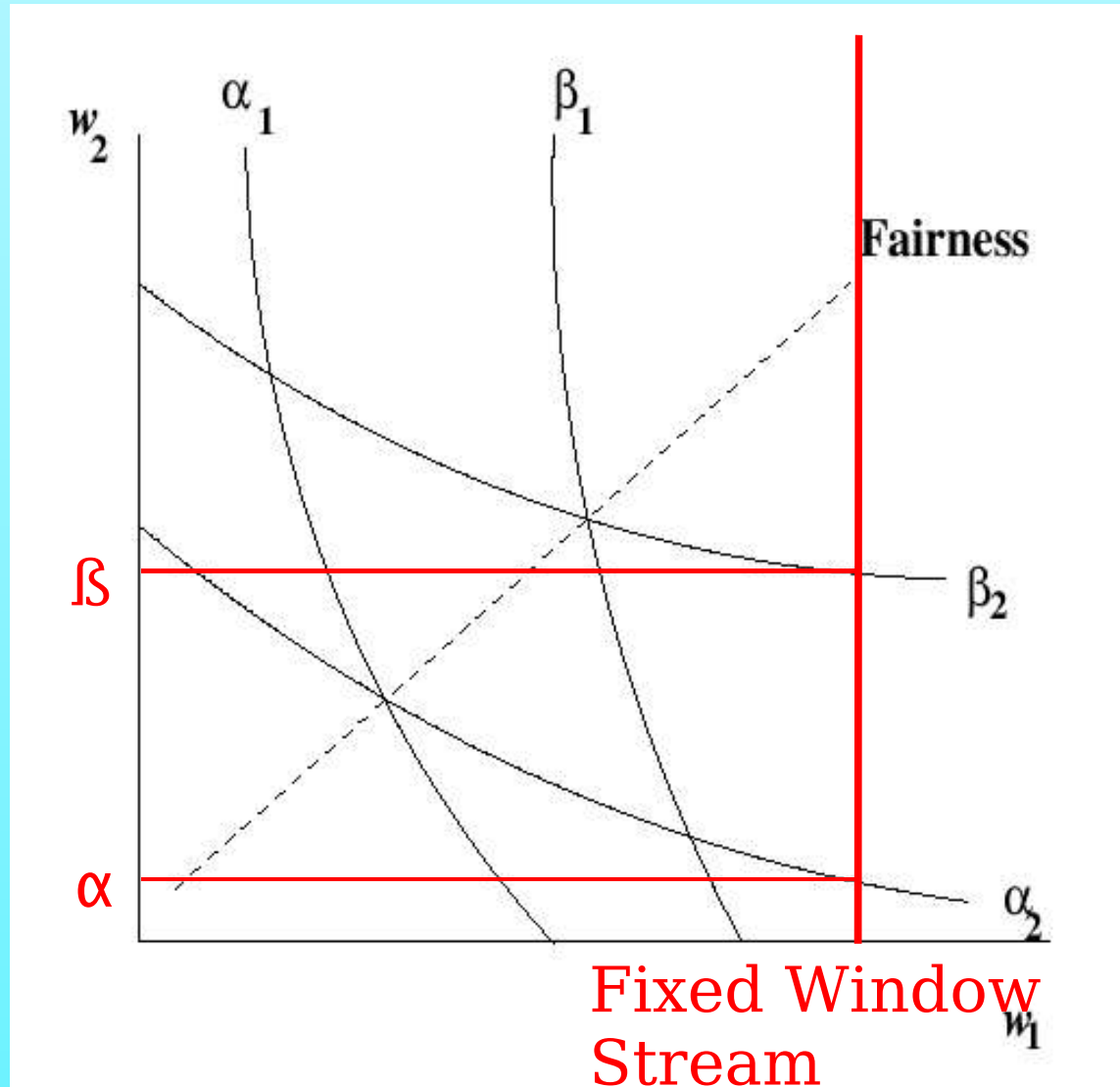


Vegas Fairness

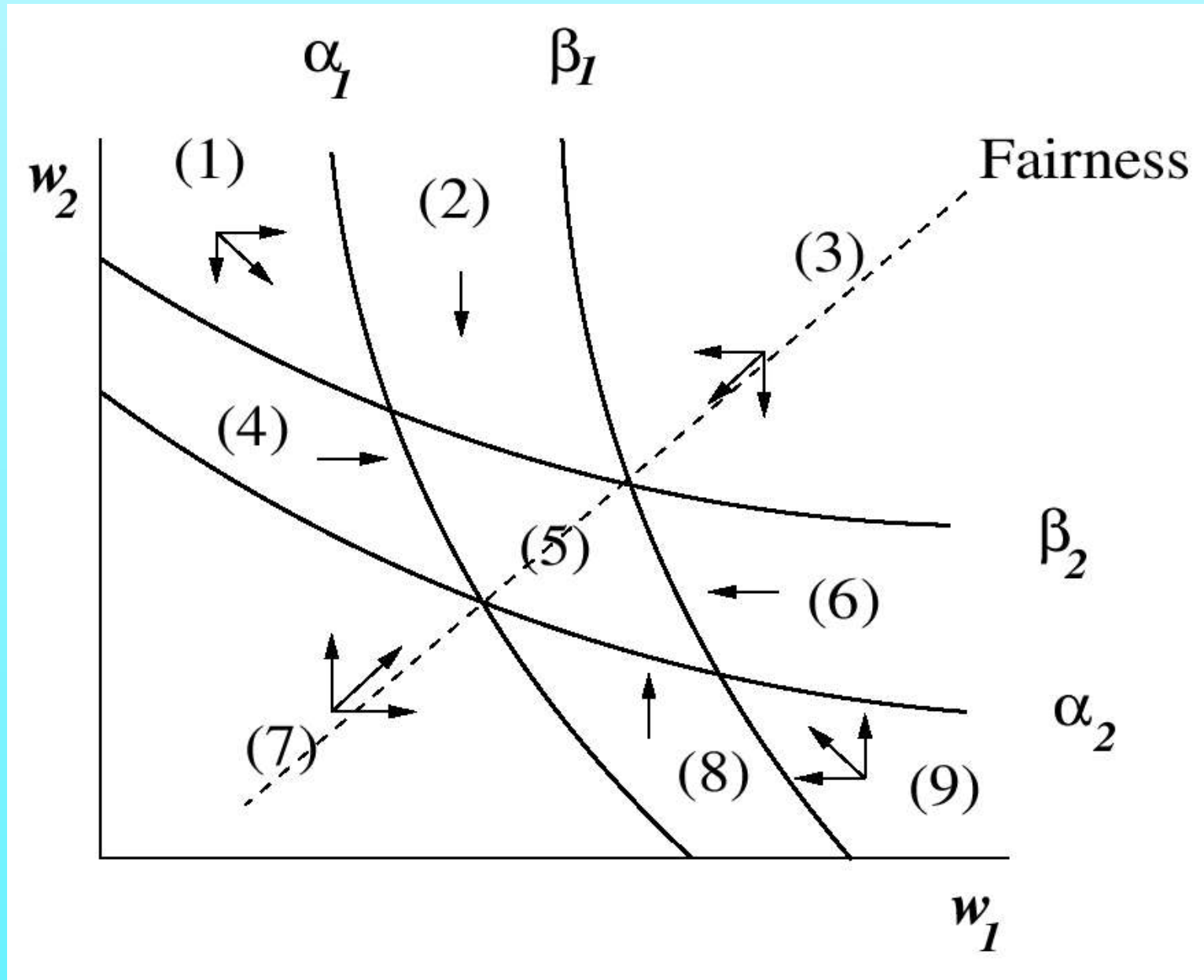
Fixed Window
Stream



Vegas Fairness

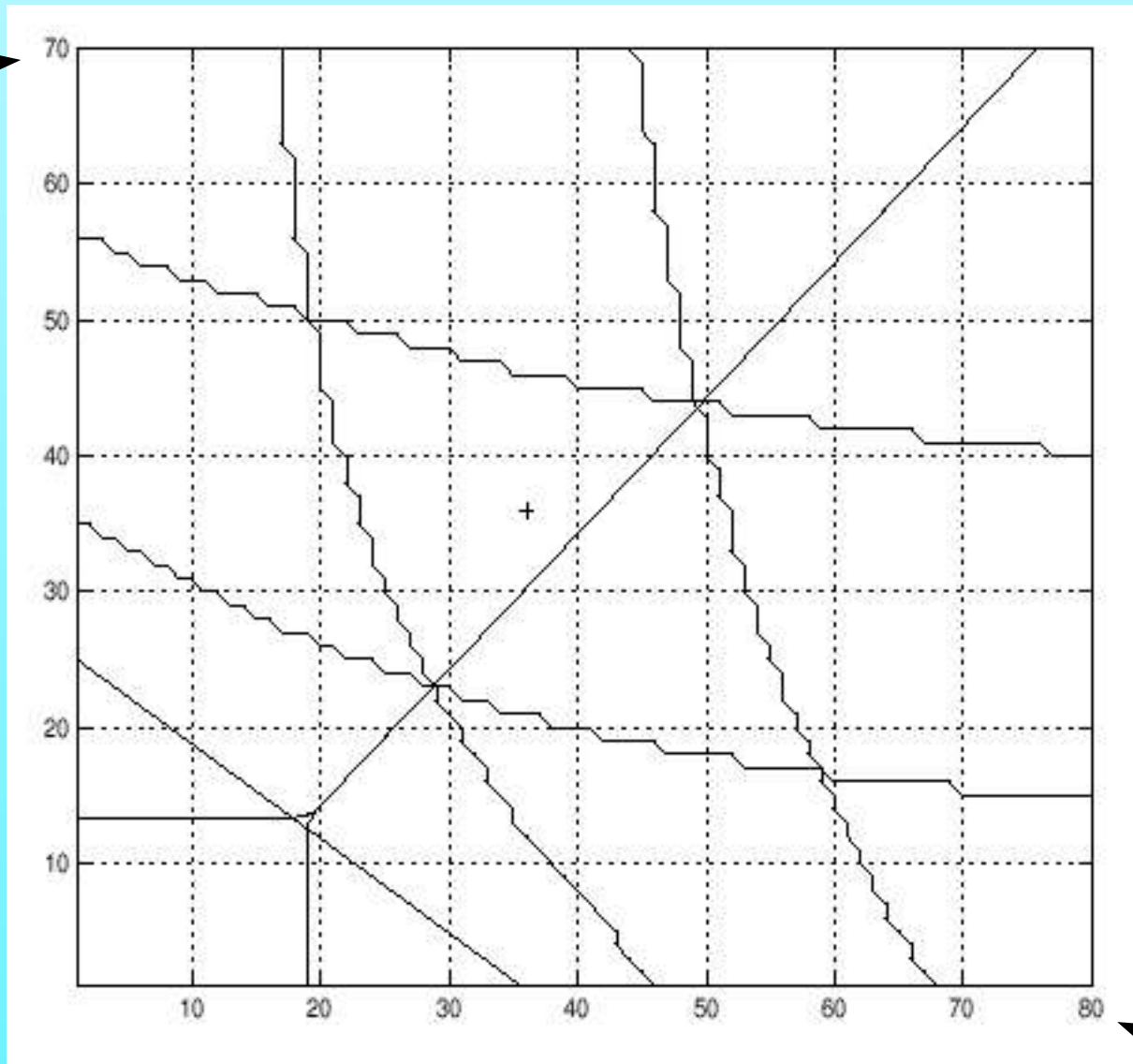


Vegas Fairness



Vegas Fairness

70 →
Delay
12.9ms

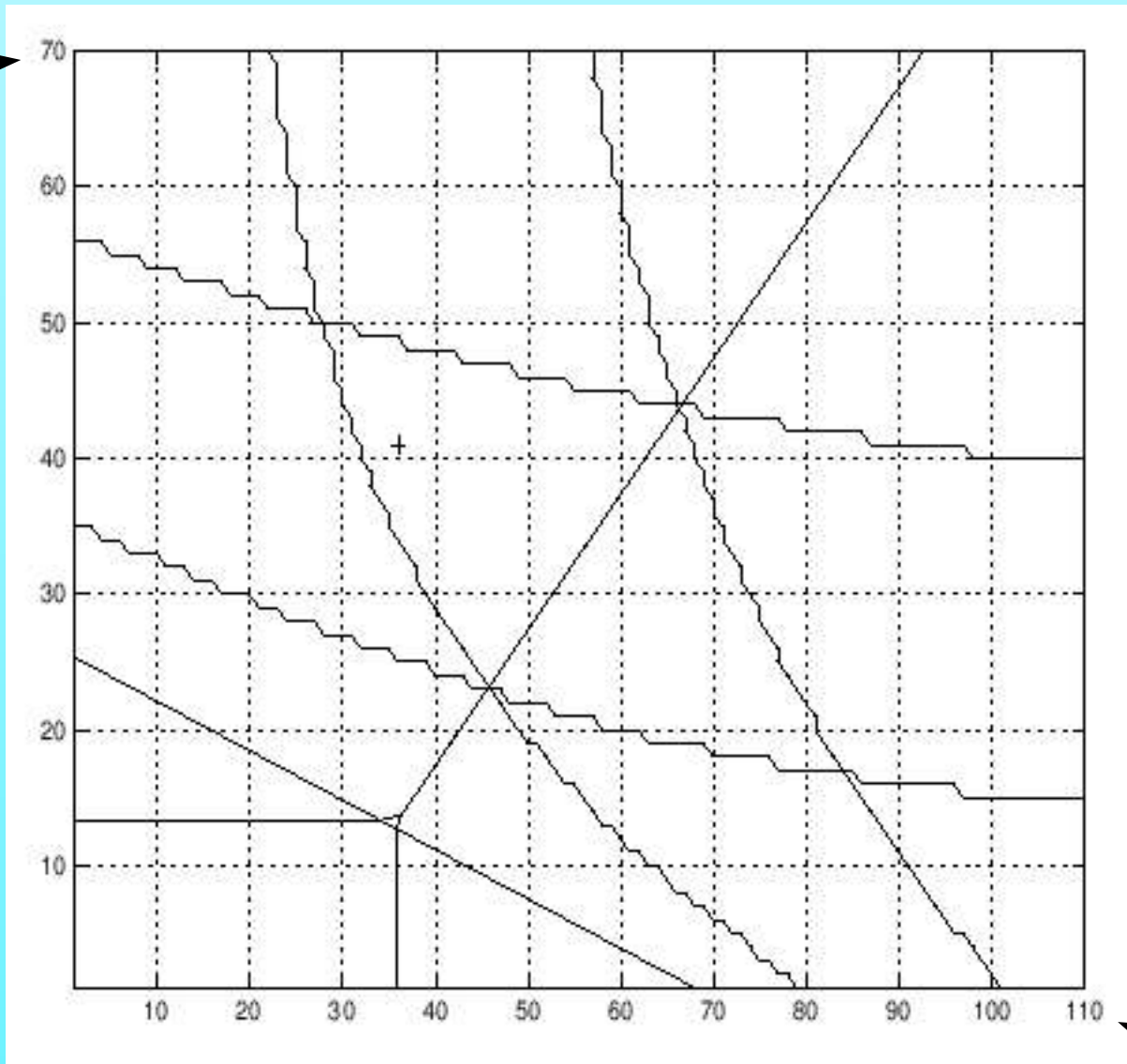


Delay
18.9ms

80

Vegas Fairness

70 →
Delay
12.9ms

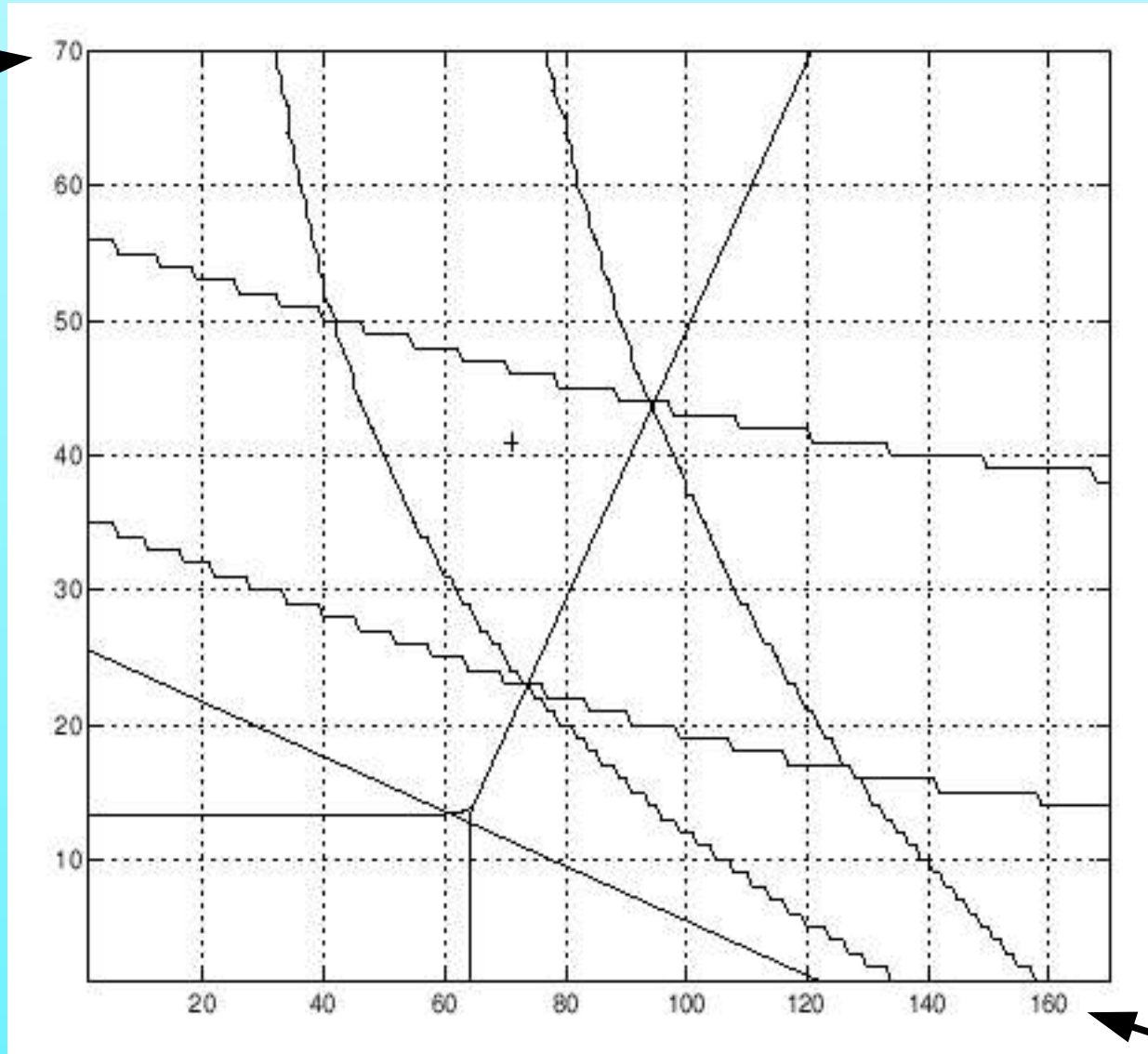


Delay
36.9ms

110

Vegas Fairness

70 →
Delay
12.9ms

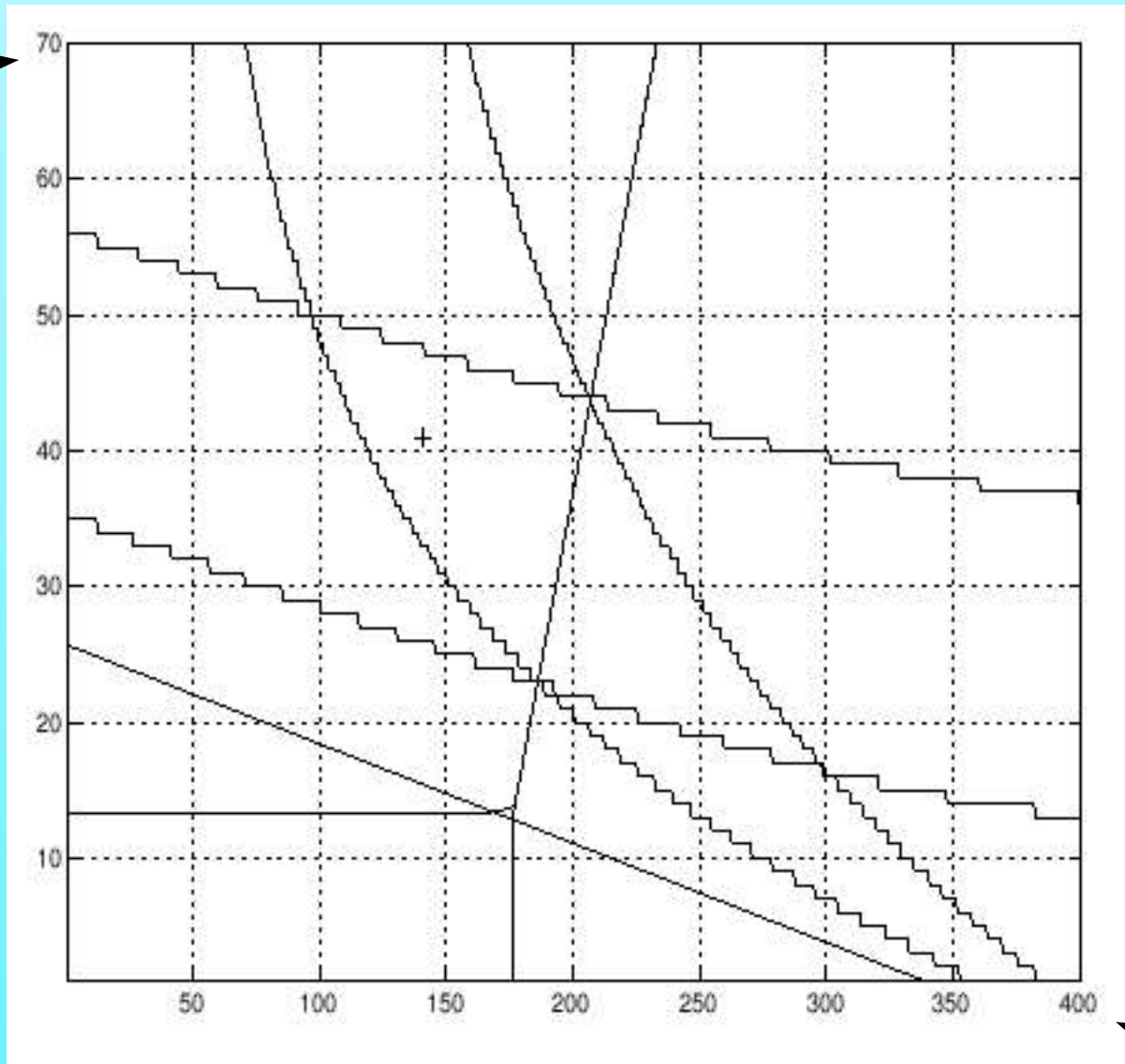


Delay
66.9ms

← 160

Vegas Fairness

70 →
Delay
12.9ms

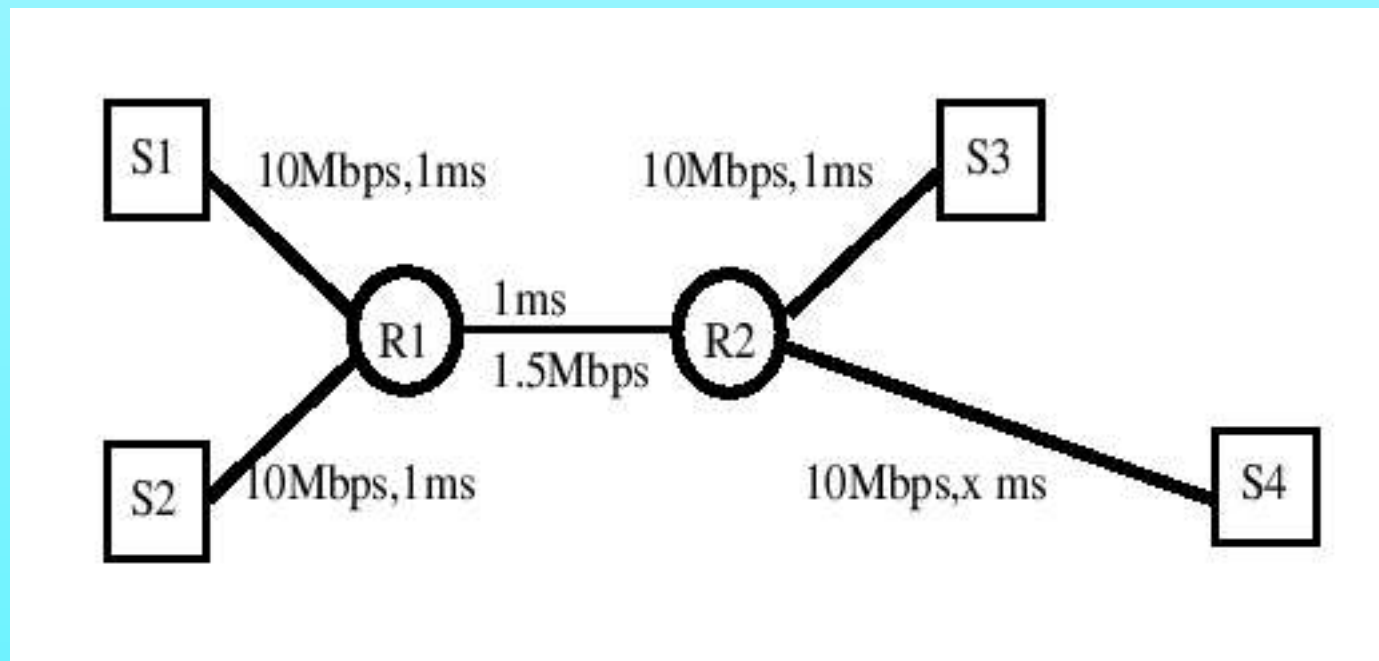


Delay
186.9ms

← 400

Simulation

- Vegas vs. Vegas
- Reno vs. Reno
- Reno vs. Vegas



Vegas vs. Vegas

x	w_1	w_2	ACK_1	ACK_2	$\frac{\max(ACK_1, ACK_2)}{\min(ACK_1, ACK_2)}$
4	3.5	3.5	21,425	16,068	1.33
13	4	7	17,522	19,965	1.14
22	4	7	20,061	17,427	1.15
58	4	13	19,507	17,973	1.09
148	4	30	21,068	16,398	1.29

Reno vs. Reno

x	ACK_1	ACK_2	ACK_1/ACK_2
4	21,100	15,637	1.35
13	25,460	11,785	2.16
22	25,684	11,672	2.20
58	34,429	2,627	13.11
148	35,598	959	37.12

Reno vs. Vegas

- Vegas less than β packets buffered
 - k
- Reno fills the rest (half the time)
 - $(B-k)/2$
- Vegas to Reno Throughput
 - $2k/(B-k)$

Reno vs. Vegas

Buffer Size	ACK of Reno	ACK of Vegas	Reno/Vegas	Interval
4	13,010	24,308	0.535	[0.167, 1.5]
7	16,434	20,903	0.786	[0.667, 3.0]
10	22,091	15,365	1.438	[1.167, 4.5]
15	25,397	12,051	2.107	[2.0, 7.0]
25	30,798	6,621	4.652	[3.667, 12.0]
50	34,443	2,936	11.73	[7.833, 24.5]

Reno vs. Vegas

Threshold	Maxthreshold	ACK of Vegas	ACK of Reno
3	6	61,069	57,774
4	8	54,259	64,305
5	10	50,823	70,020
7	14	44,469	77,202
10	20	34,081	87,379

Conclusion

- Rerouting
- Persistent Congestion
- Vegas vs. Vegas Fairness
- Reno vs. Reno Fairness
- Vegas and Reno do not cooperate

Thank You

'The time has come,' Jack Meier said,
'To speak of many thing...'