draft-welzl-rmcat-coupled-cc-01

Coupled Congestion Control for RTP Media

Michael Welzl, Safiqul Islam, Stein Gjessing

Networks and Distributed Systems Group

Department of Informatics

University of Oslo

A note about evaluations

As a starting point, considering

- greedy and non-greedy flows
  - Evaluation with realistic RMCAT traffic planned as next step

Why do we need coupled cc?

- Each individual data stream (flow) has its own congestion control mechanism
  - Hence, M flows, with their own congestion control modules, trying to reach a certain fairness lead to:
    - More queue growth
    - More delay
    - More packet drops
    - Fairness problems in case of heterogeneous RTTs

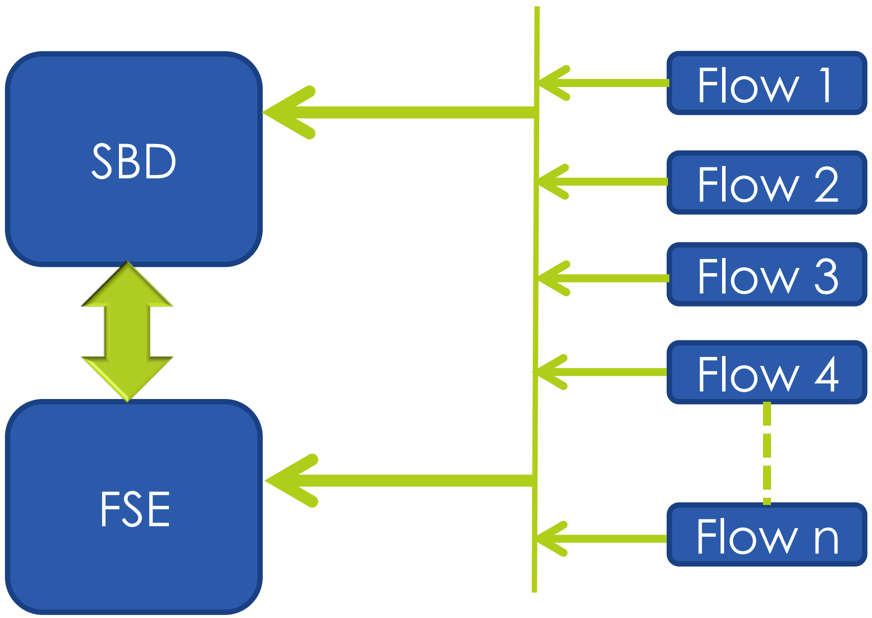
How to solve this

- This can be solved by using a single Congestion Control(CC) instance for the flows
  - To begin with, only for flows initiated from the same sender sharing the same bottleneck
- Congestion Manager, RFC 3124, had some unresolved issues, and was complicated to implement
  - We suggest something more in the style of RFC 2140 (but rate based, and with more features)

Flow State Exchange (FSE)

- A passive entity which stores information from the flows, calculates rate and provides this calculated rate back

- Minimal change to existing CC: each time it updates its sending rate (New_CR), the flow calls update (New_CR, New_DR), and gets the new rate



FSE

- Flow Numbers, #
- Flow Group Identifier, FGI
- Priority P
- Calculated Rate, CR
- Desired Rate, DR

#	FGI	P	CR	DR	Rate
1	1	1	6	8	6
2	1	0.5	1	1	1

FSE maintains S_CR (which is meant to be the sum of the calculated rates) and TLO (Total Leftover Rate) per FG.

FSE – how it works

- Flow 1 experienced congestion, causing S_CR to drop from 11 to 9.
- Let assume that flow 2 has sent an update to the FSE.
- For all the flows in its FG (including itself), it calculates the sum of all the calculated rates, new_S_CR. Then it calculates the difference between CR(f) and new_CR, DELTA.

#	FGI	P	CR	DR	Rate
1	1	1	6	8	6
2	1	0.5	1	1	1

S_CR = 9, and TLO = 0

New_CR = 2, new_DR = inf

```
for all flows i in FG do
    new_S_CR = new_S_CR + CR(i)
end for
DELTA =  new_CR - CR(f)
```

New_S_CR = 7
DELTA = 2 - 1 = 1

FSE – how it works

- It updates S_CR, CR(f) and DR(f).

```
CR(f) = new_CR
if DELTA > 0 then
        S_CR = S_CR + DELTA
else if DELTA < 0 then
        S_CR = new_S_CR + DELTA
end if
DR(f) = min(new_DR,CR(f))
```

#	FGI	P	CR	DR	Rate
1	1	1	6	8	6
2	1	0.5	2	2	1

S_CR = 9, and TLO = 0

S_CR = 10, and TLO = 0

CR(f) = 2

Delta positive, S_CR = 9 + 1 = 10

DR(f) = 2

FSE – how it works

- It calculates the leftover rate TLO, removes the terminated flows from the FSE and calculates the sum of all the priorities, S_P.

```
for all flows i in FG do
    if P(i)<0 then
        delete flow
    else
        S_P = S_P + P(i)
    end if
end for

if DR(f) < CR(f) then
    TLO = TLO + (P(f)/S_P) * S_CR - DR(f))
end if
```

#	FGI	P	CR	DR	Rate
1	1	1	6	8	6
2	1	0.5	2	2	1

S_CR = 10, and TLO = 0

S_P = 1.5

FSE – how it works

- It calculates the sending rate.

```
Rate = min(new_DR, (P(f)*S_CR)/S_P + TLO)
if Rate != new_DR and TLO > 0 then
      TLO = 0  // f has 'taken' TLO
end if
```

#	FGI	P	CR	DR	Rate
1	1	1	6	8	6
2	1	0.5	3.33	3.33	3.33

S_CR = 10, and TLO = 0

Rate = min (inf, 0.5/1.5 * 10 + 0) = 3.33

- It updates DR(f) and CR(f) with Rate.

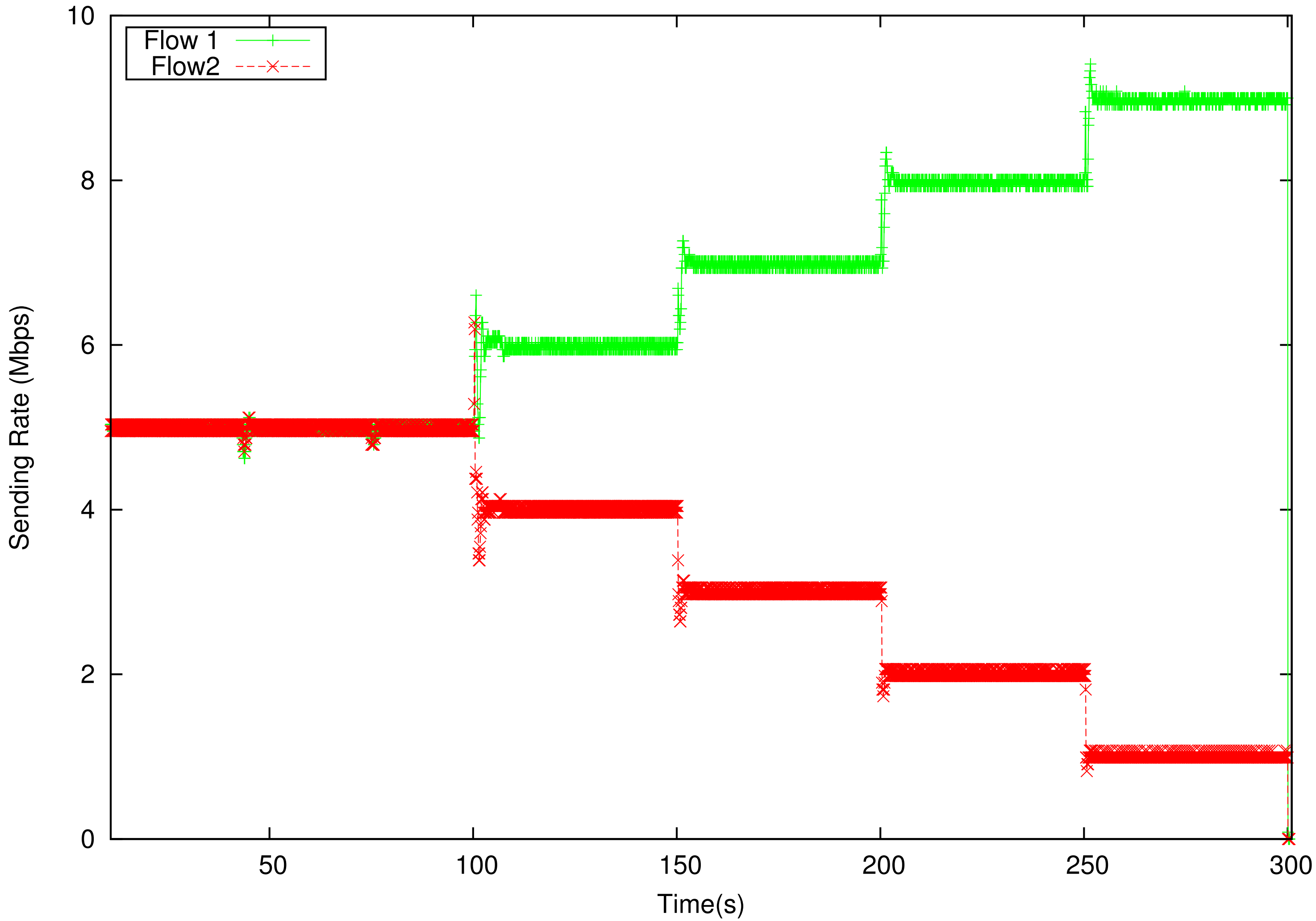
```
if Rate > DR(f) then
      DR(f) = Rate
end if
 CR(f) = Rate
```

DR(f) = 3.33, CR(f) = 3.33

Simulation Results

- Good News !!

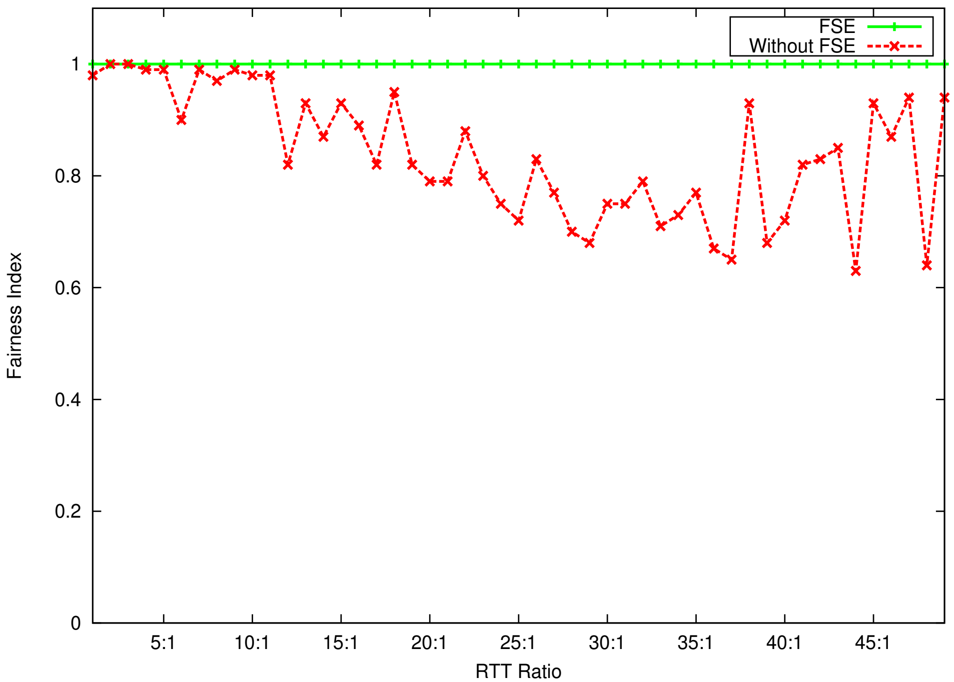
Priority



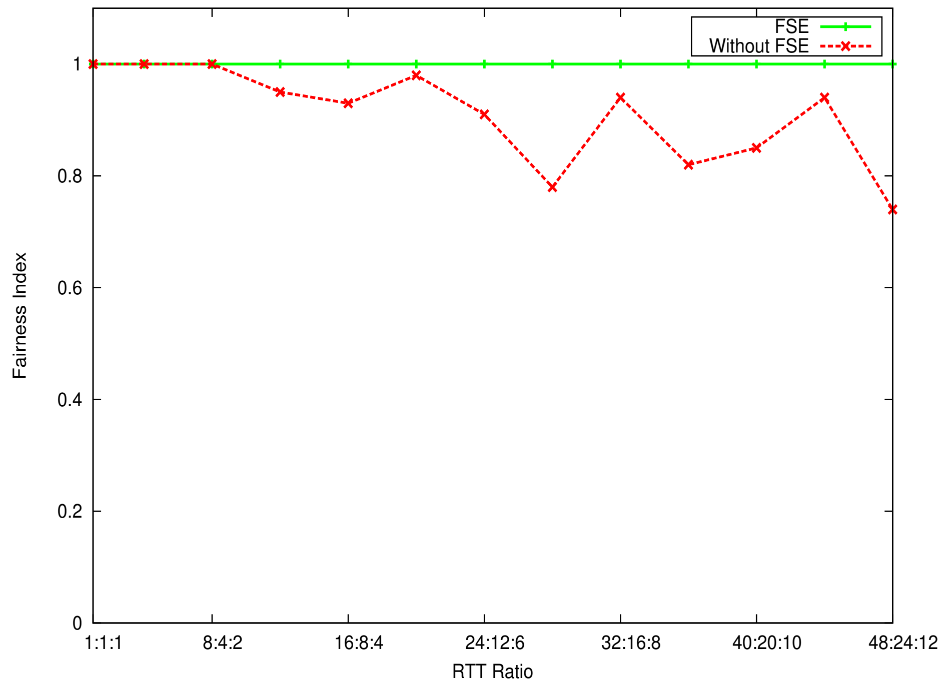
Flow 1
Flow2
Sending Rate (Mbps)
Time(s)
0
2
4
6
8
10
50
100
150
200
250
300

Fairness

Fairness Index- for 2 flows

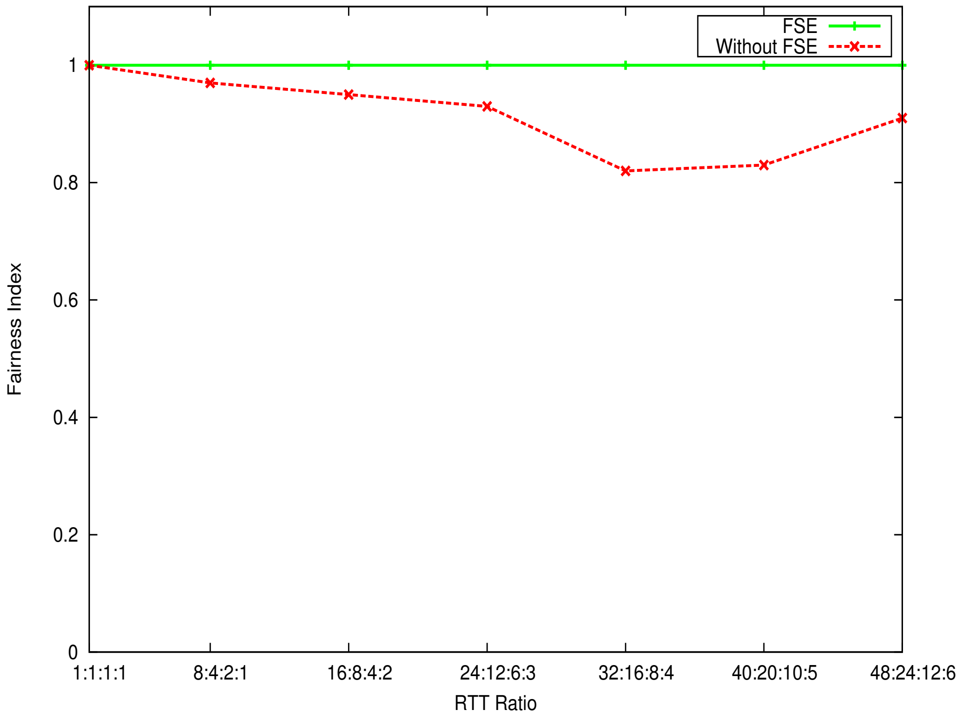


Fairness Index- for 3 flows

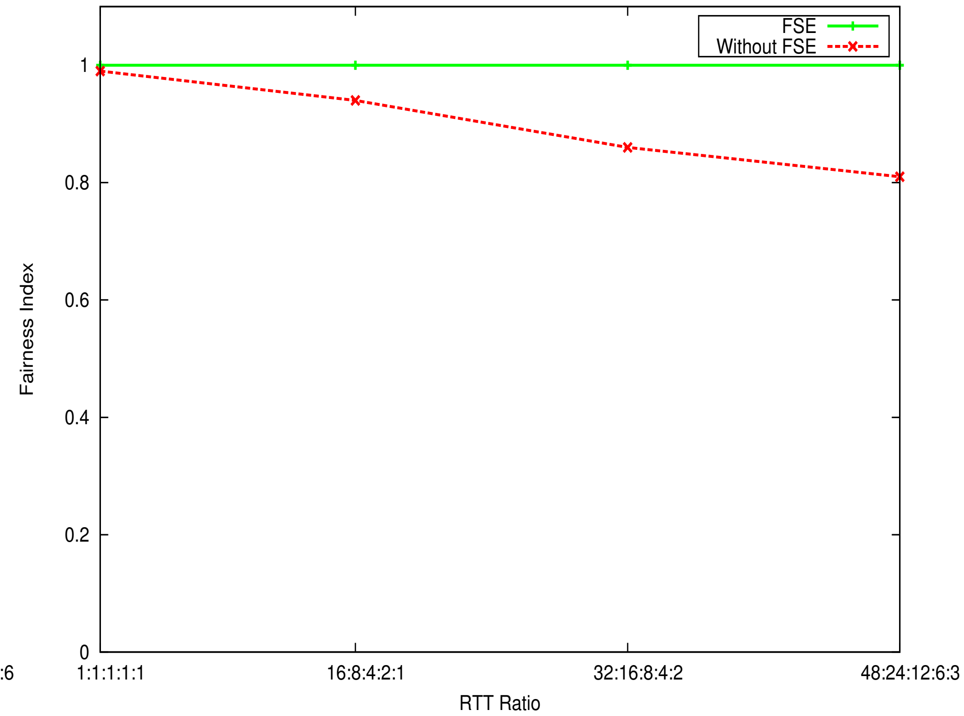


Fairness

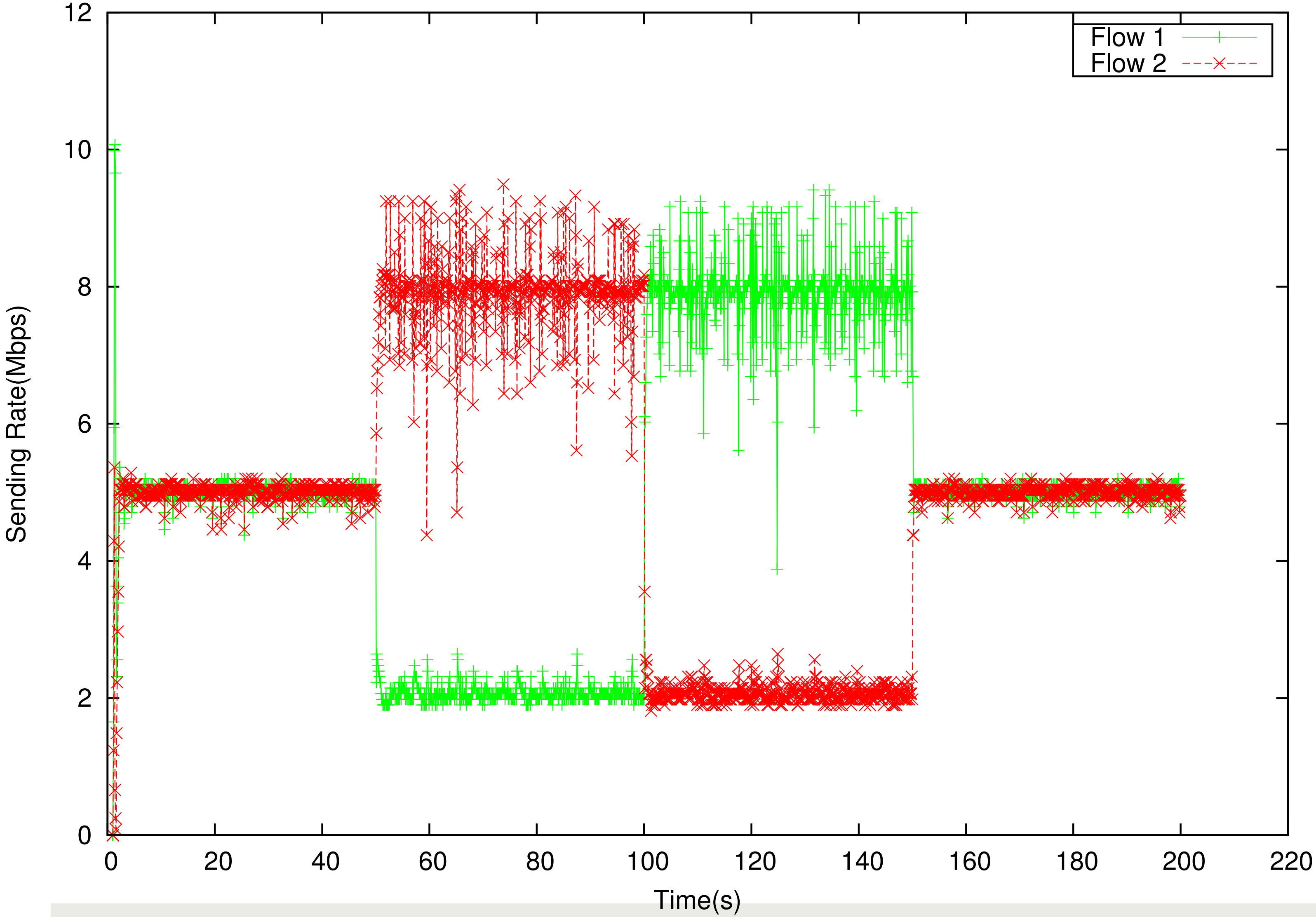
Fairness Index- for 4 flows



Fairness Index- for 5 flows



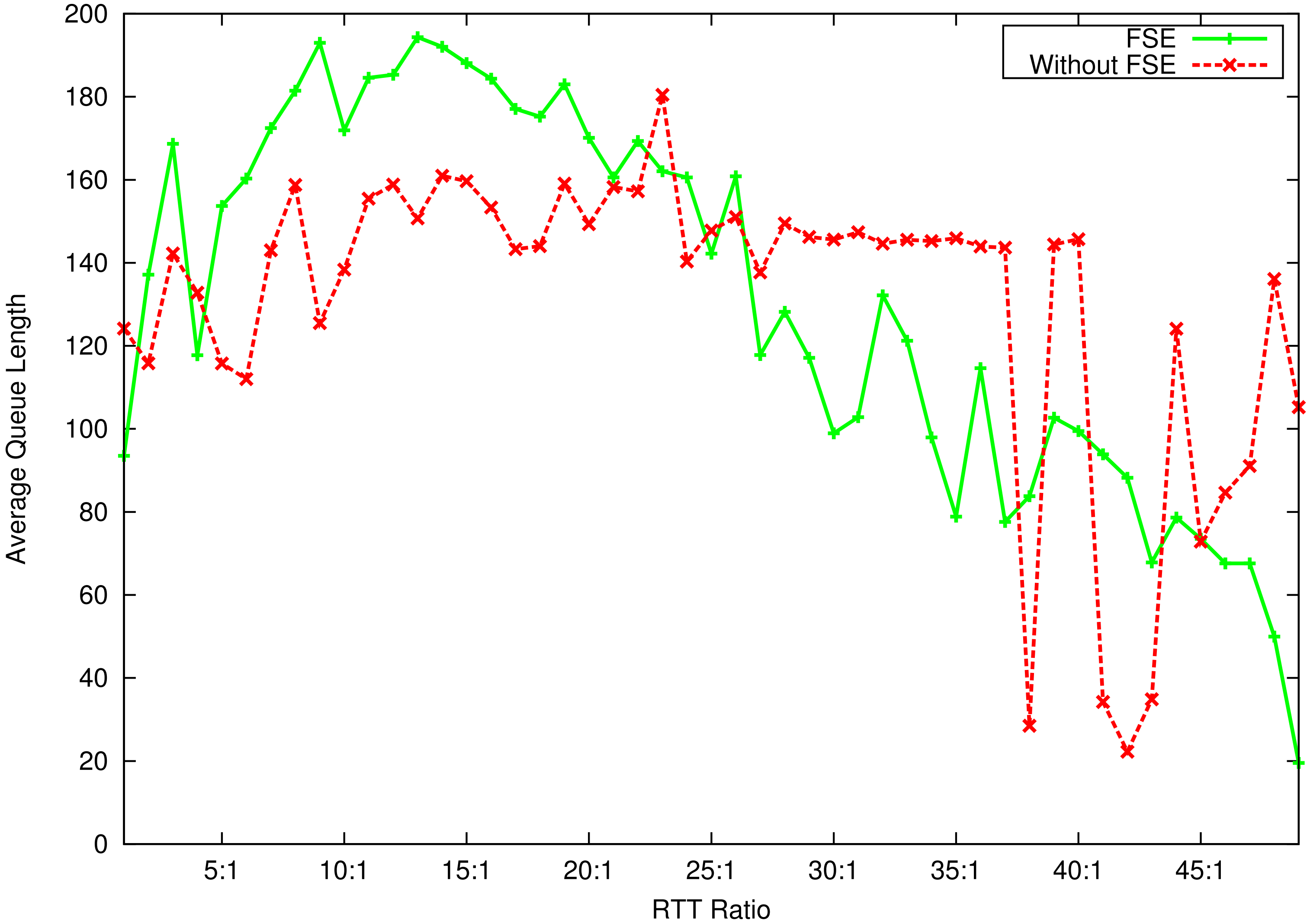
Benefits From The Non-Greedy Flows



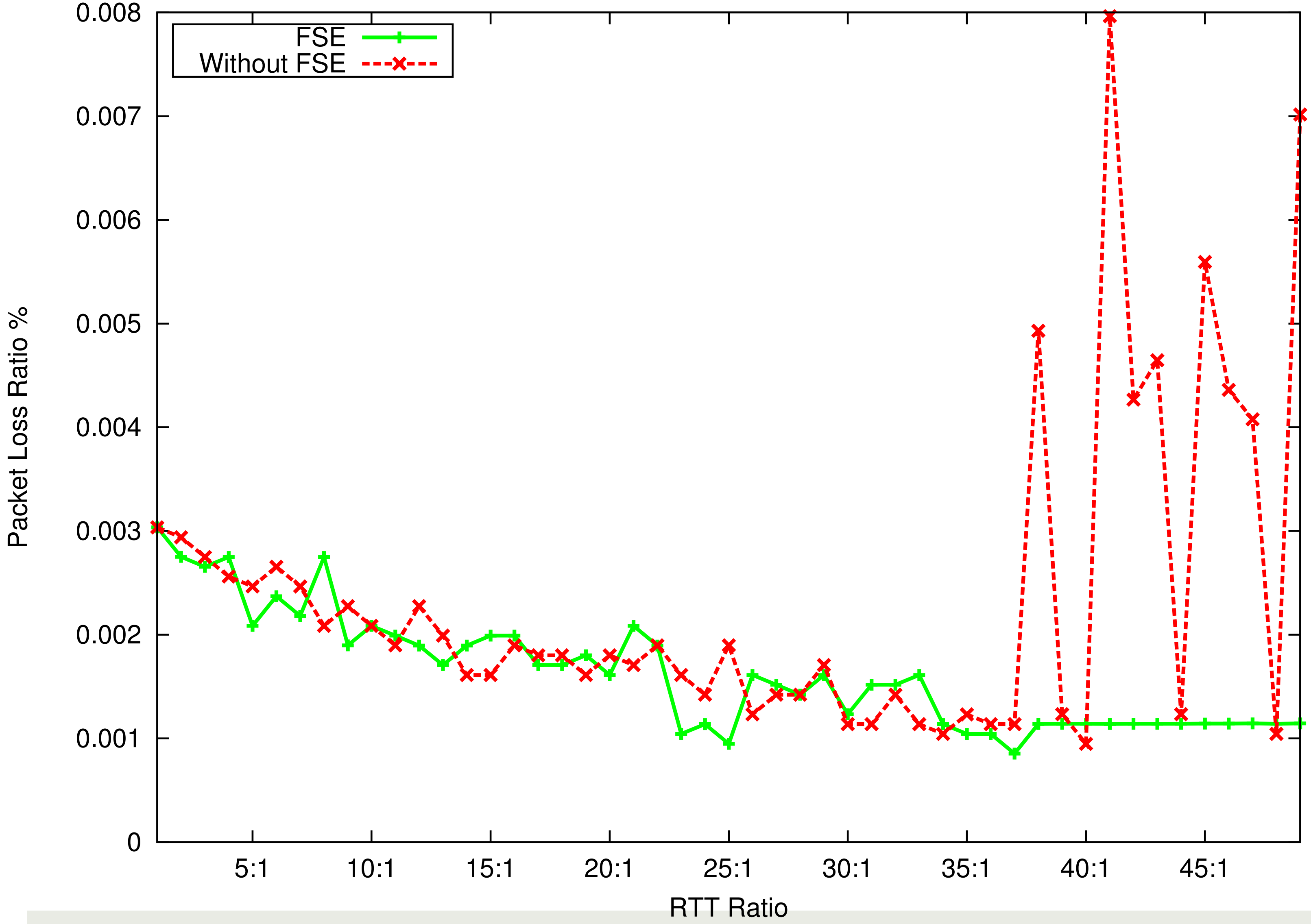
Simulation Results

- Sad Part !!

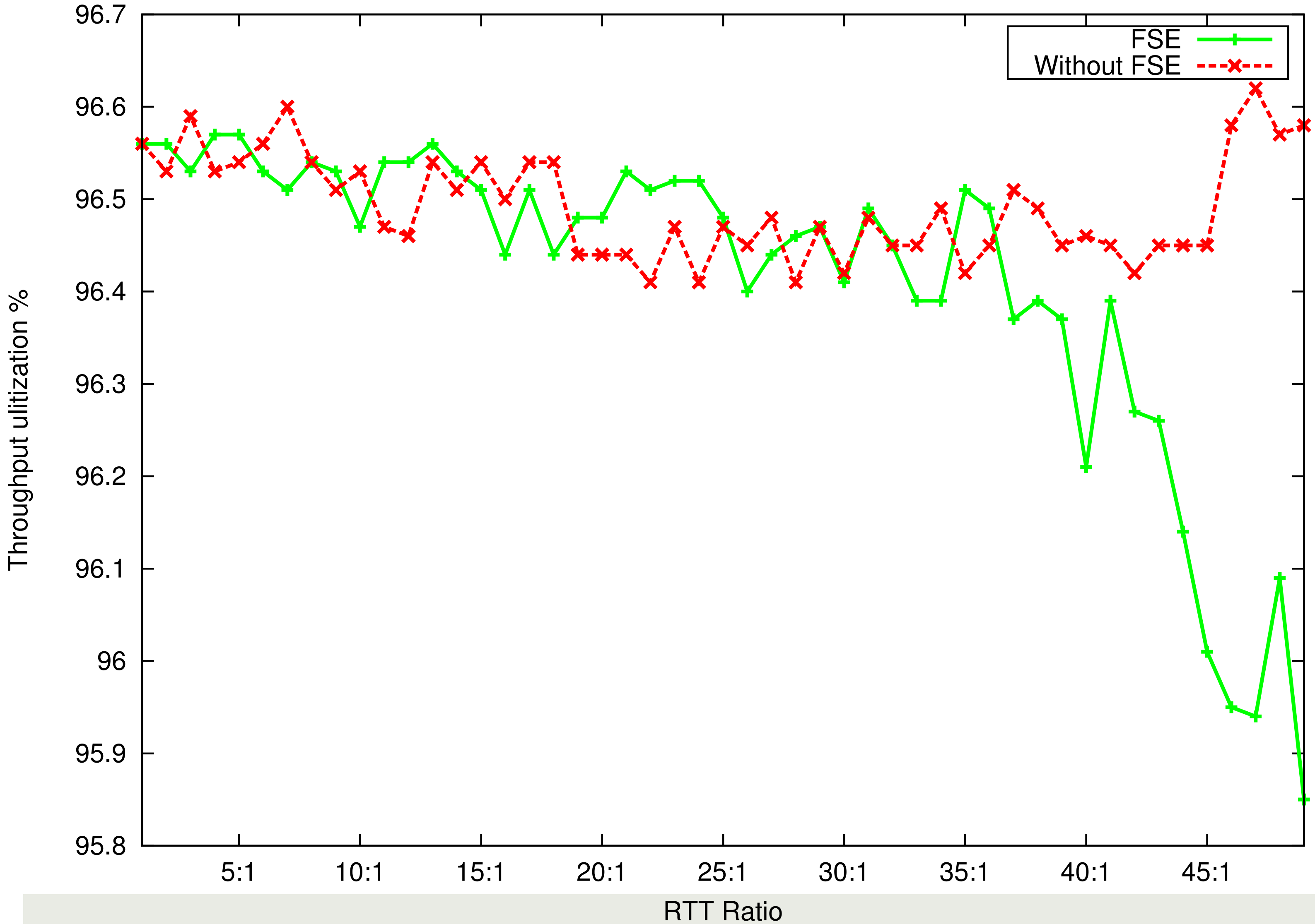
Average Queue Length – 2 Flows



Packet Loss Ratio – 2 Flows



Throughput for 2 flows

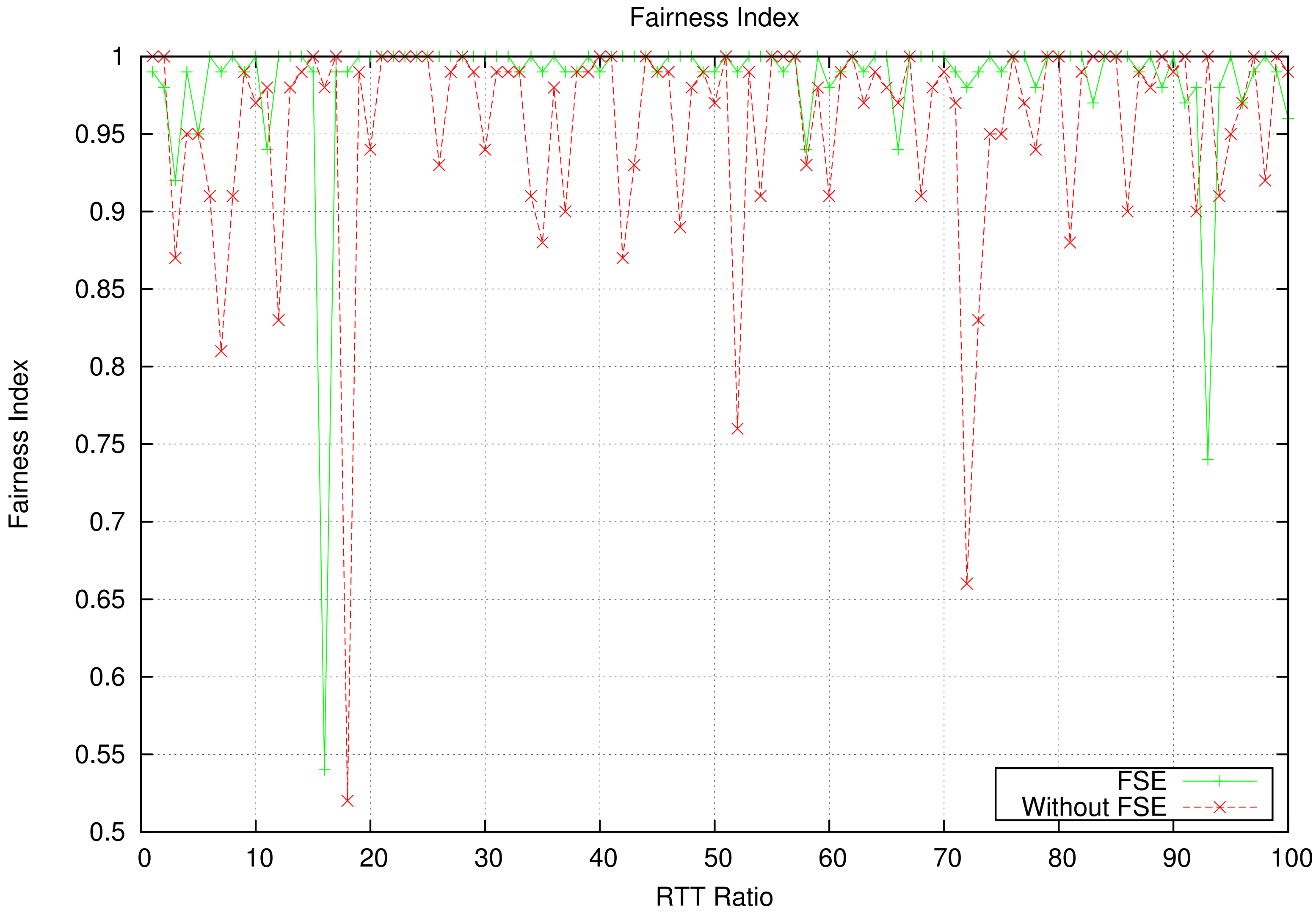


Future plans

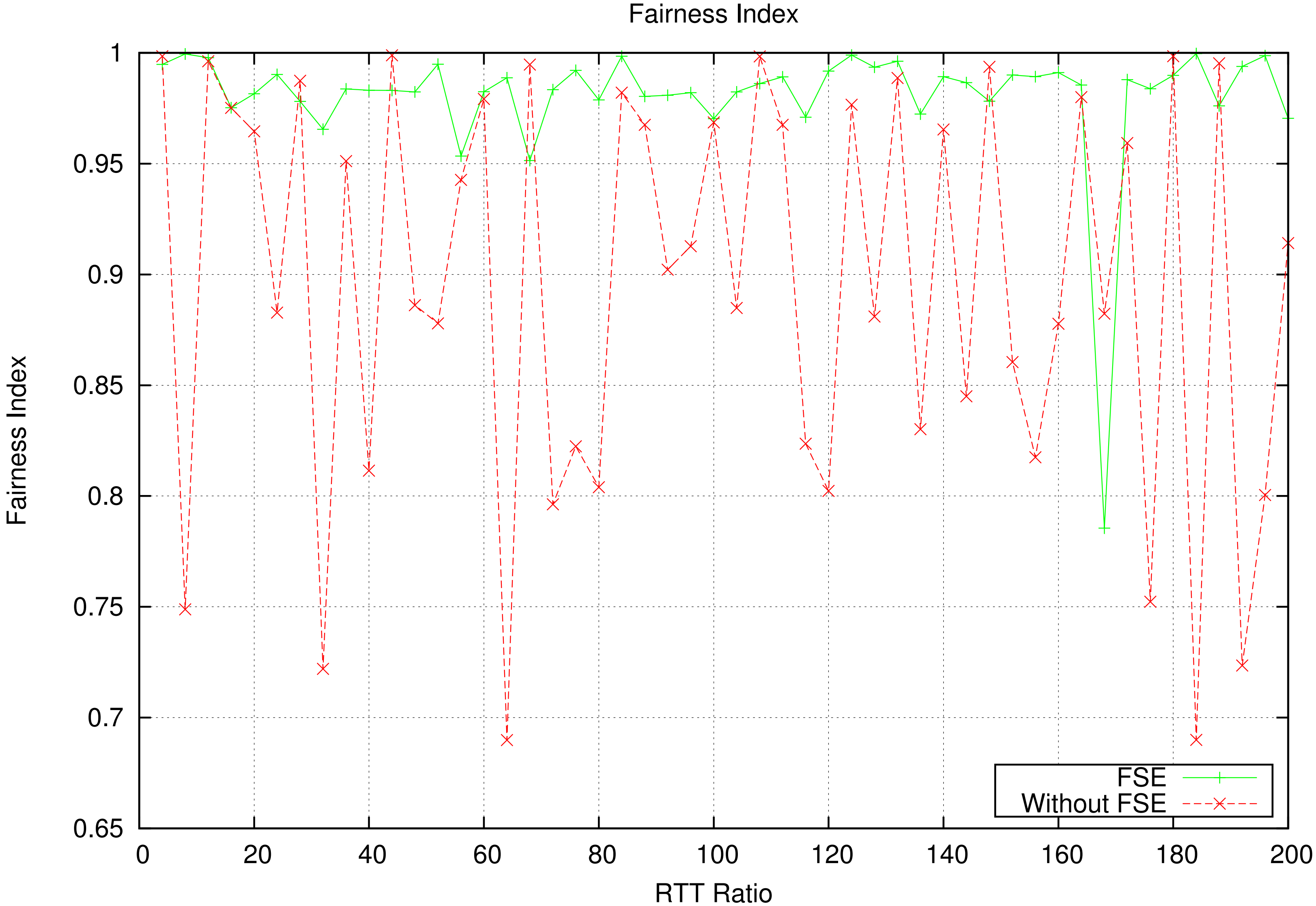
- We want to keep the FSE as simple as possible
  - Trying passive for now – see if the problems are due to TFRC, or require other changes to the algorithm
- Else, we go for (slightly) active
  - When congestion is noticed by a flow, FSE immediately informs all other flows in the same FSE

Backup Slides

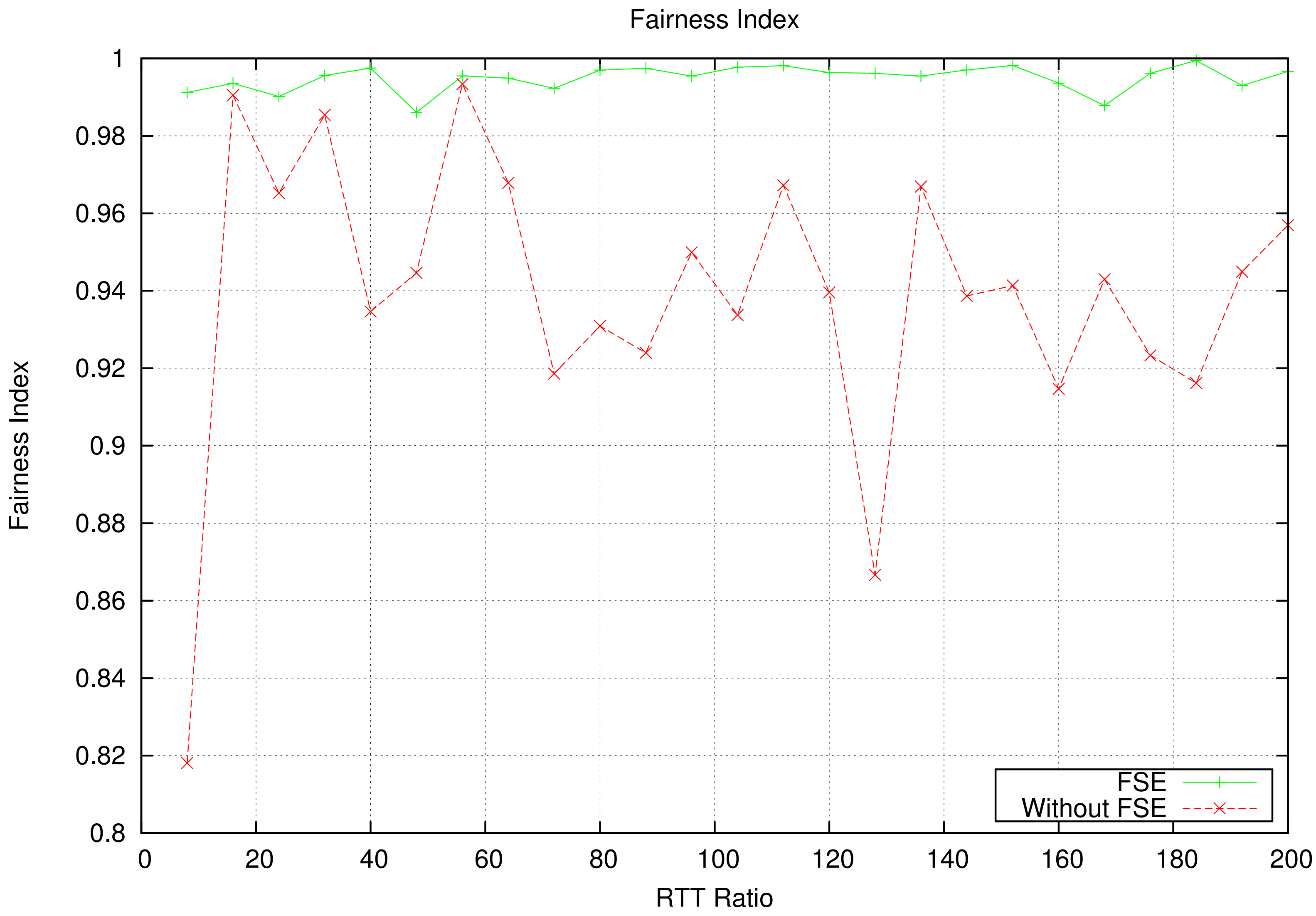
Fairness Index – 2 flows



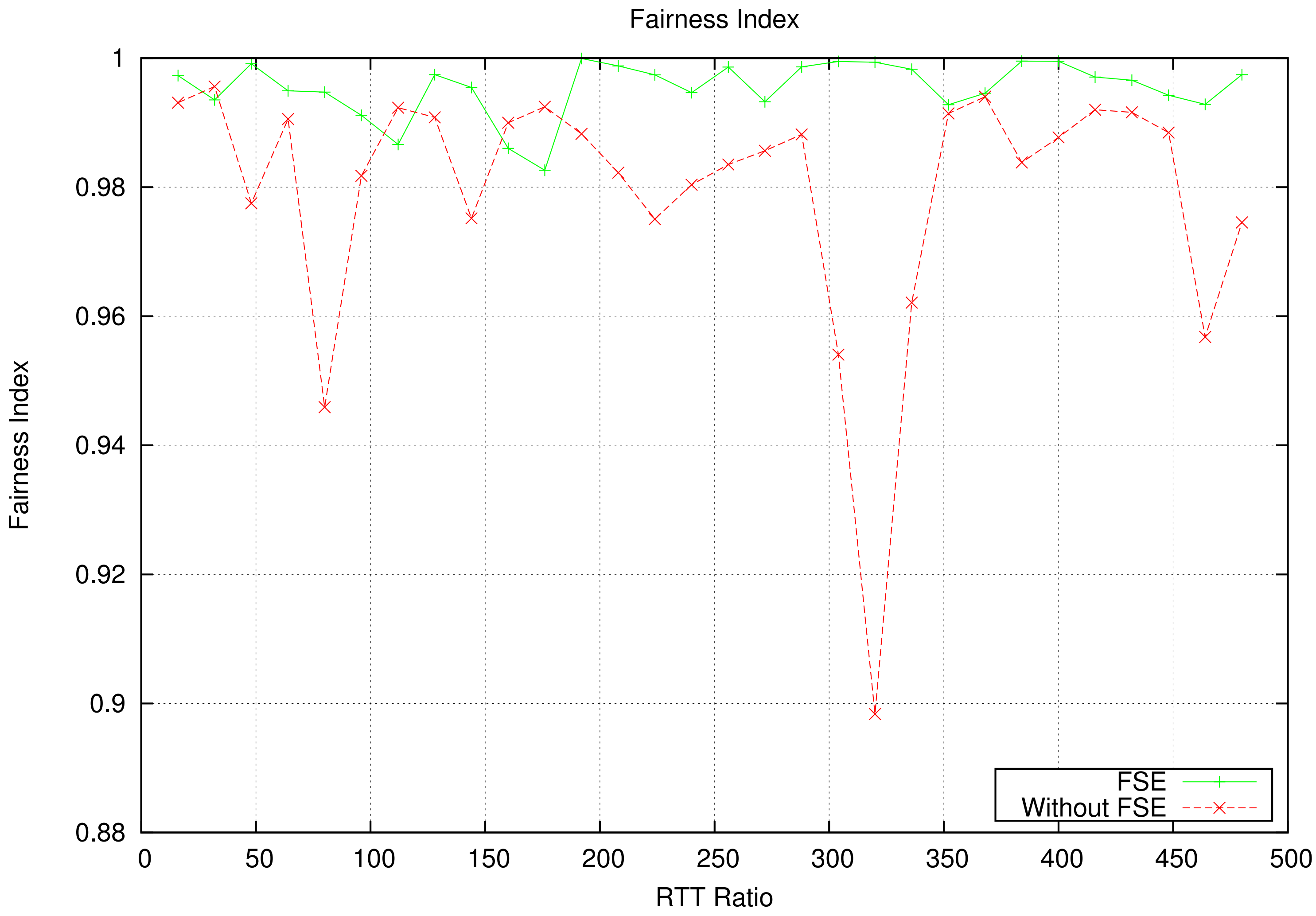
Fairness Index – 3 Flows



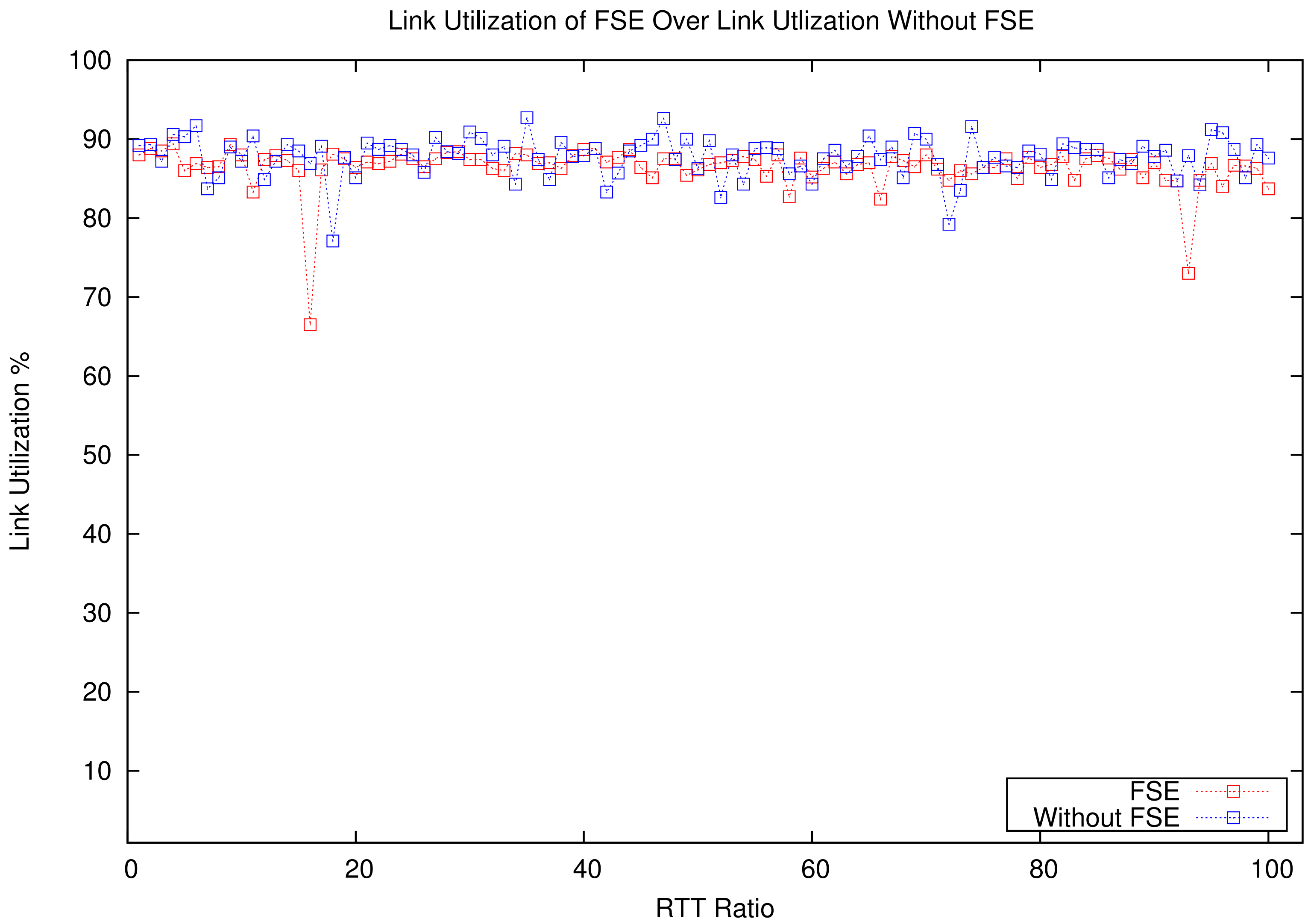
Fairness Index – 4 flows



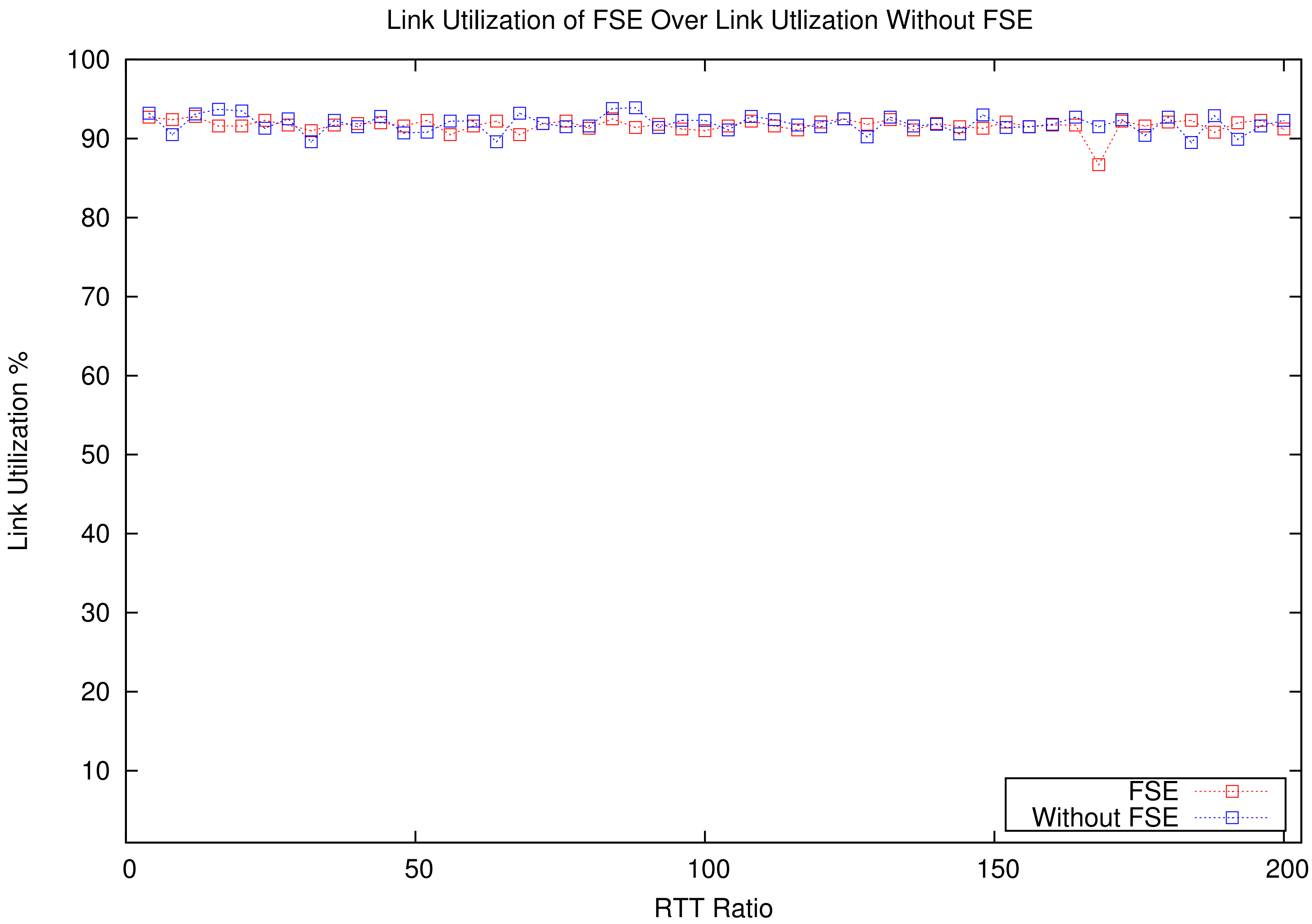
Fairness Index – 5 Flows



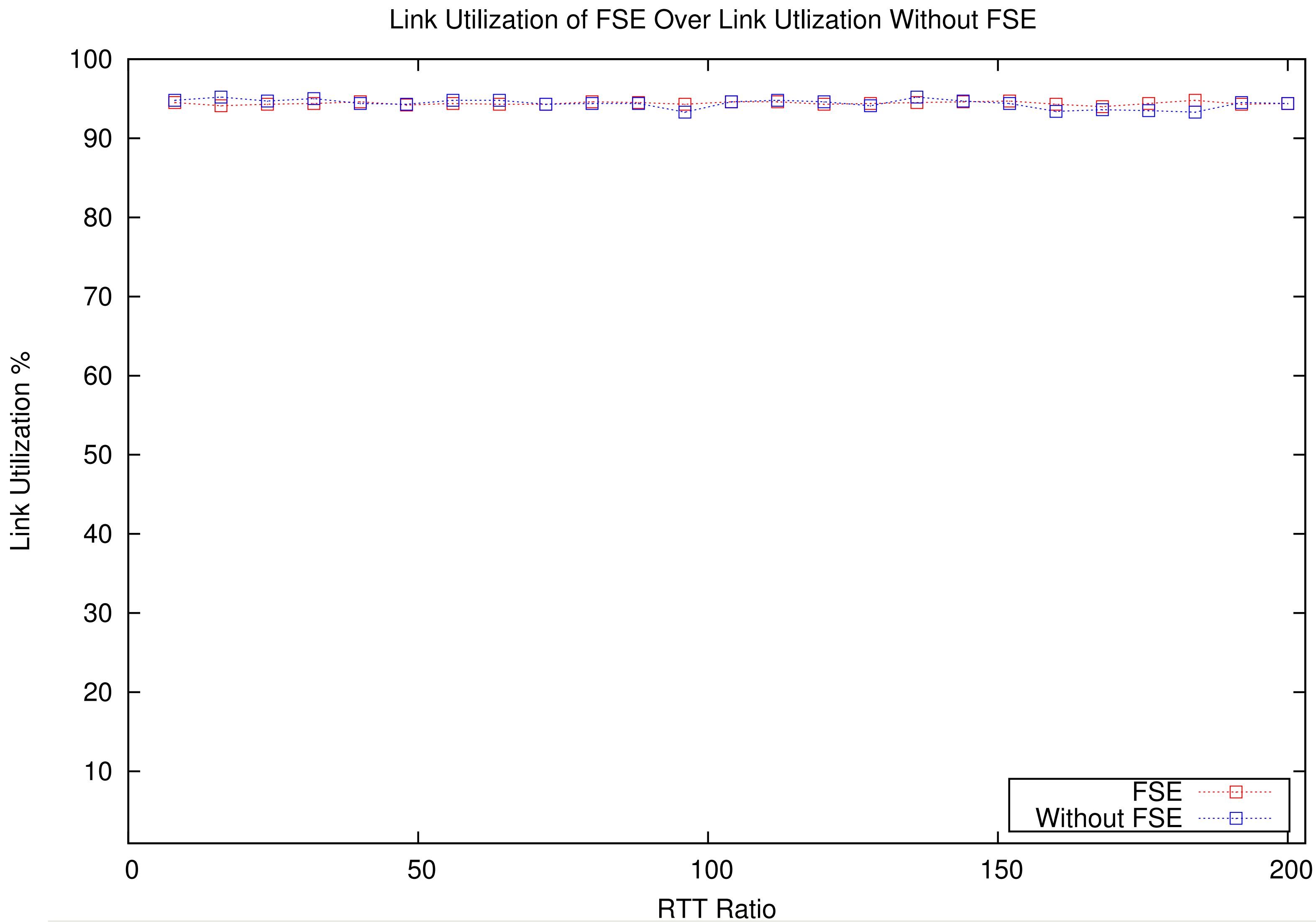
Throughput – 2 Flows



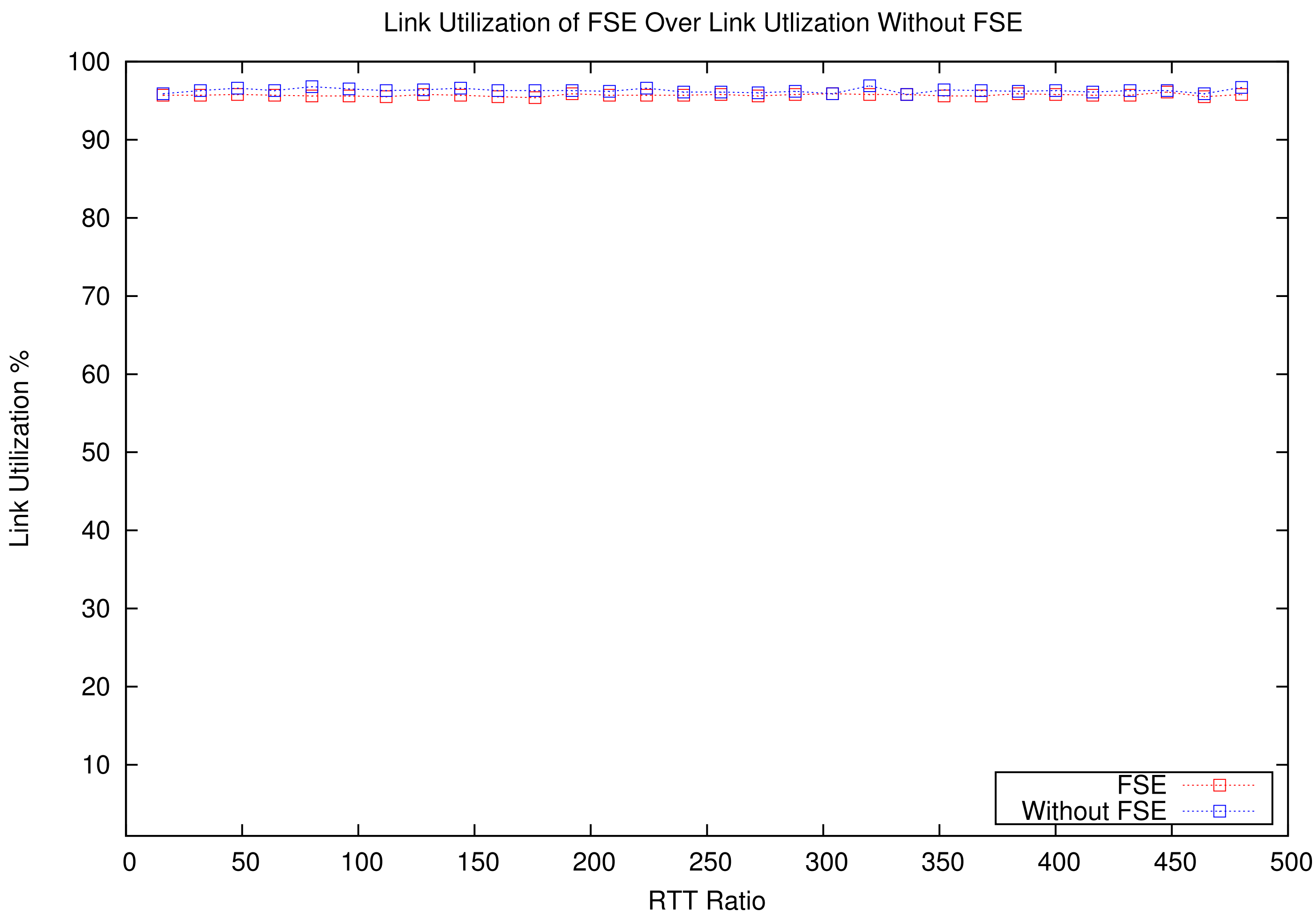
Throughput – 3 Flows



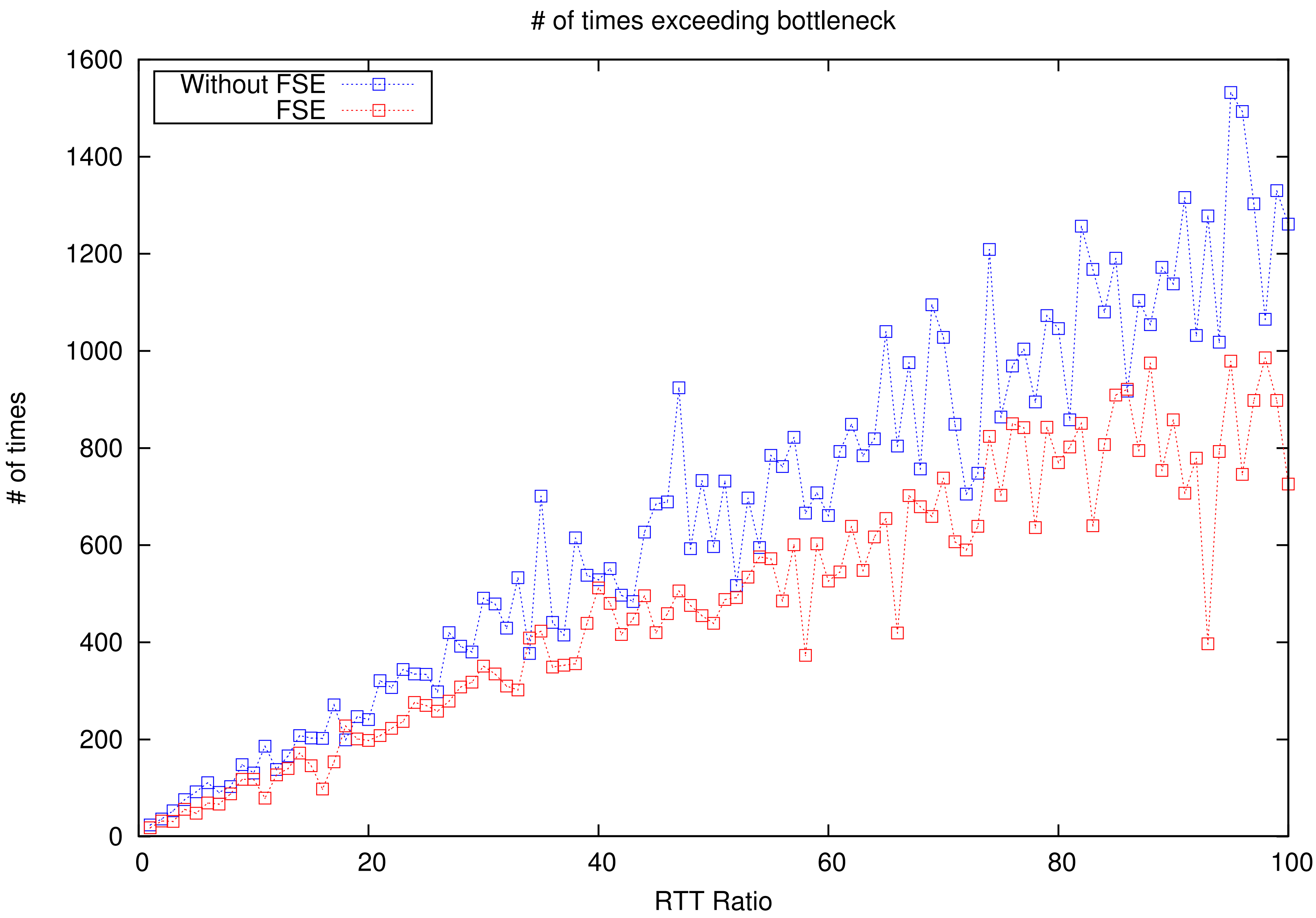
Throughput – 4 Flows



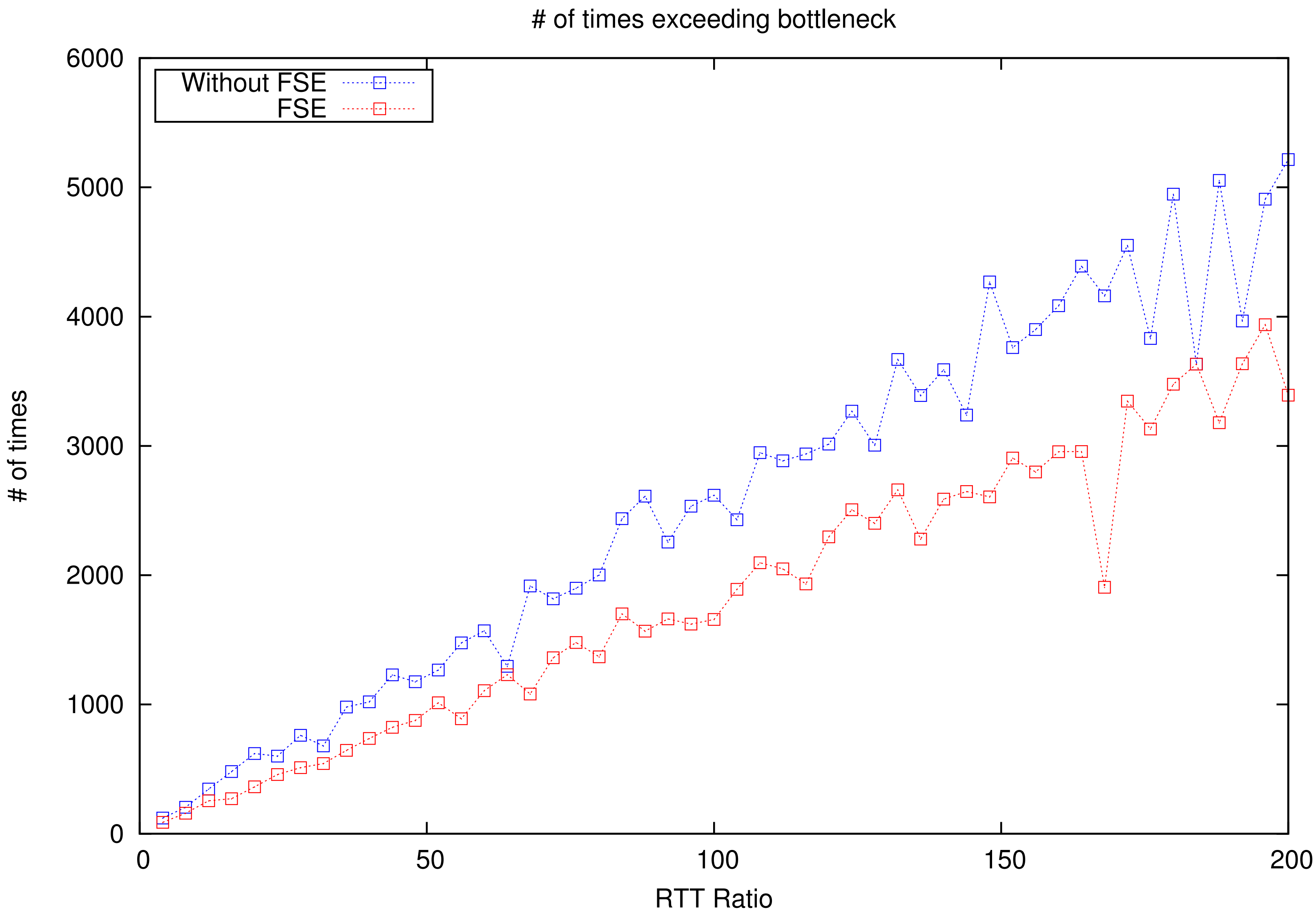
Throughput – 5 Flows



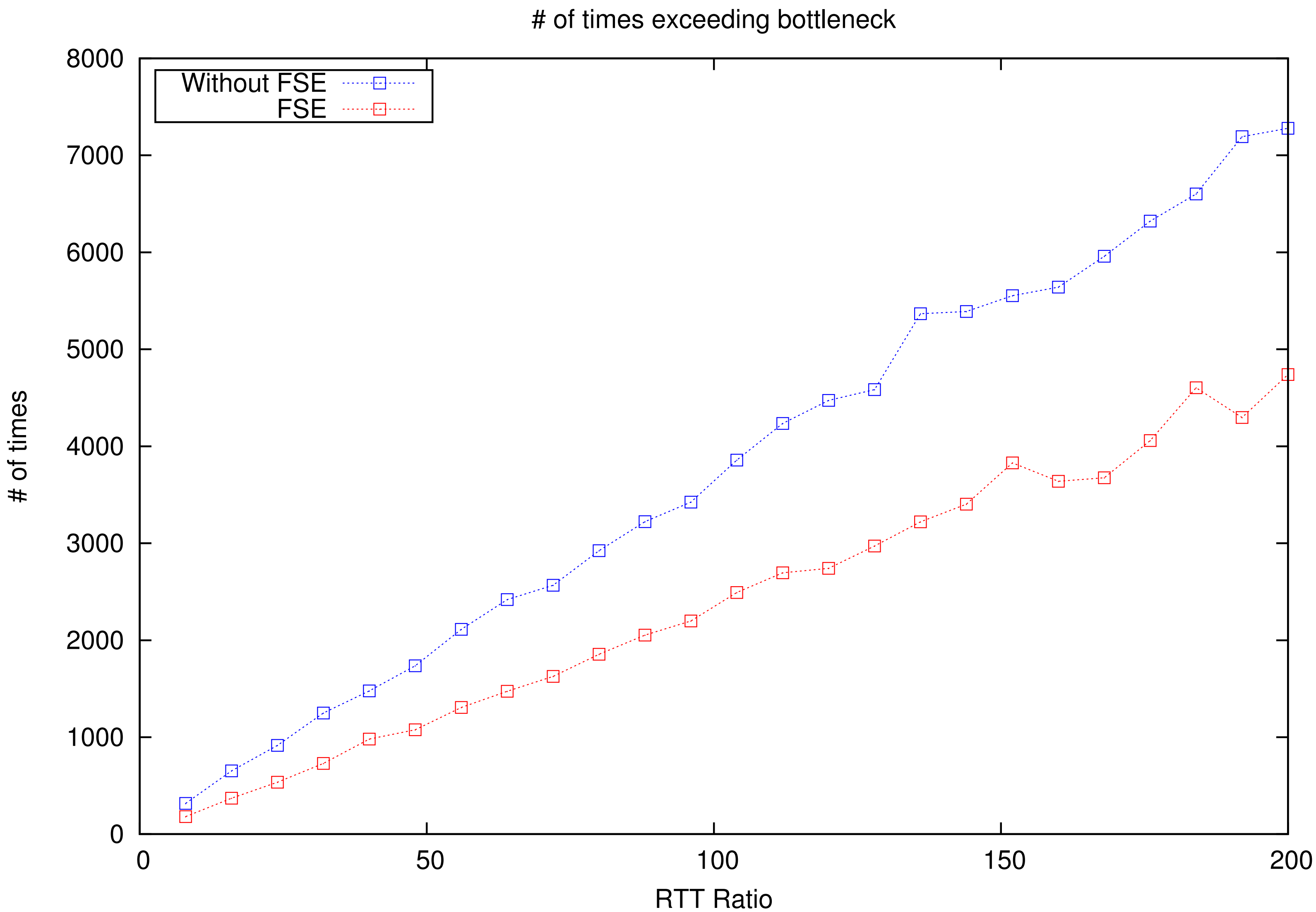
Exceeding Bottleneck – 2 Flows



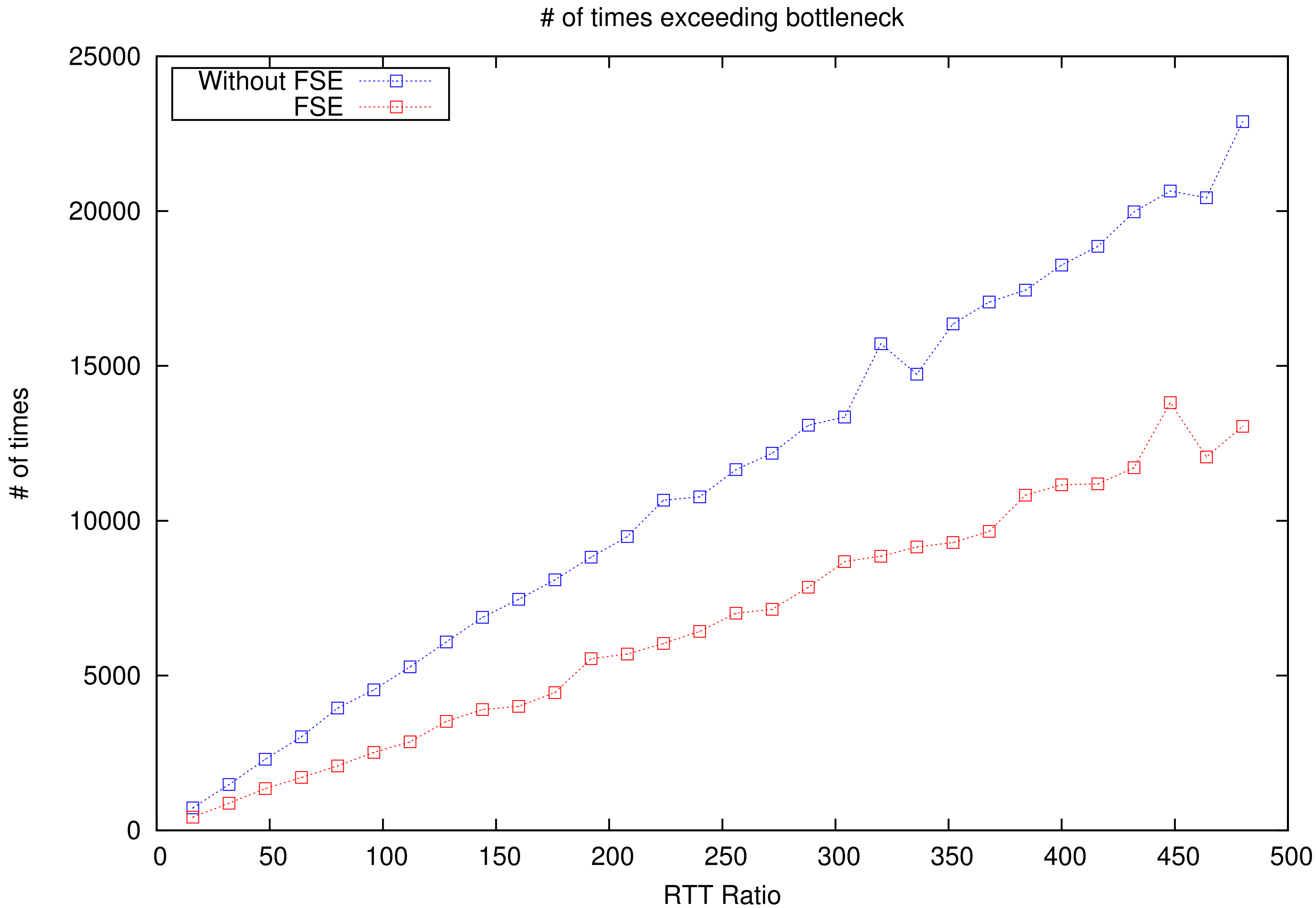
Exceeding Bottleneck – 3 Flows



Exceeding Bottleneck – 4 Flows



Exceeding Bottleneck – 5 Flows



Benefits from the non-greedy flows