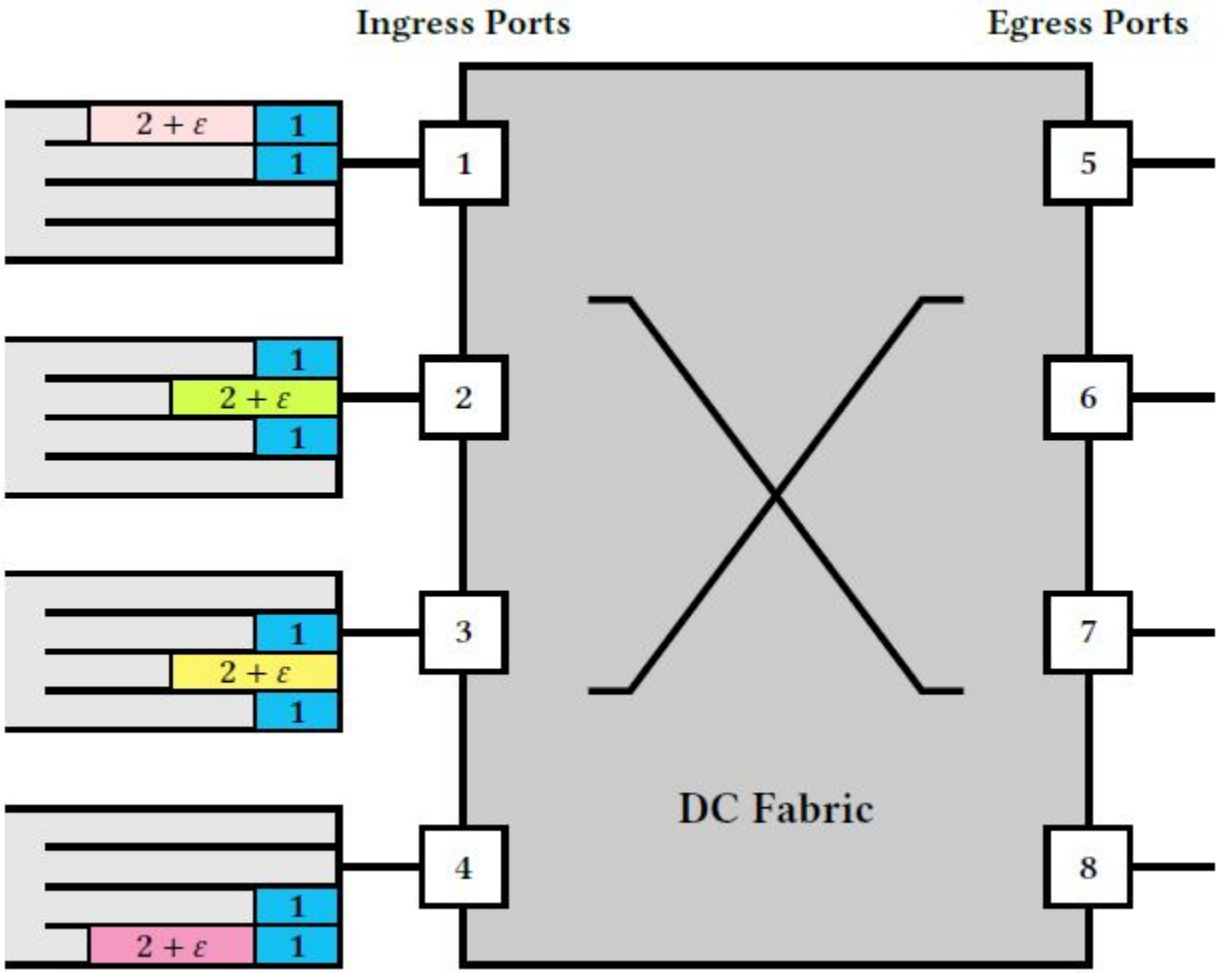
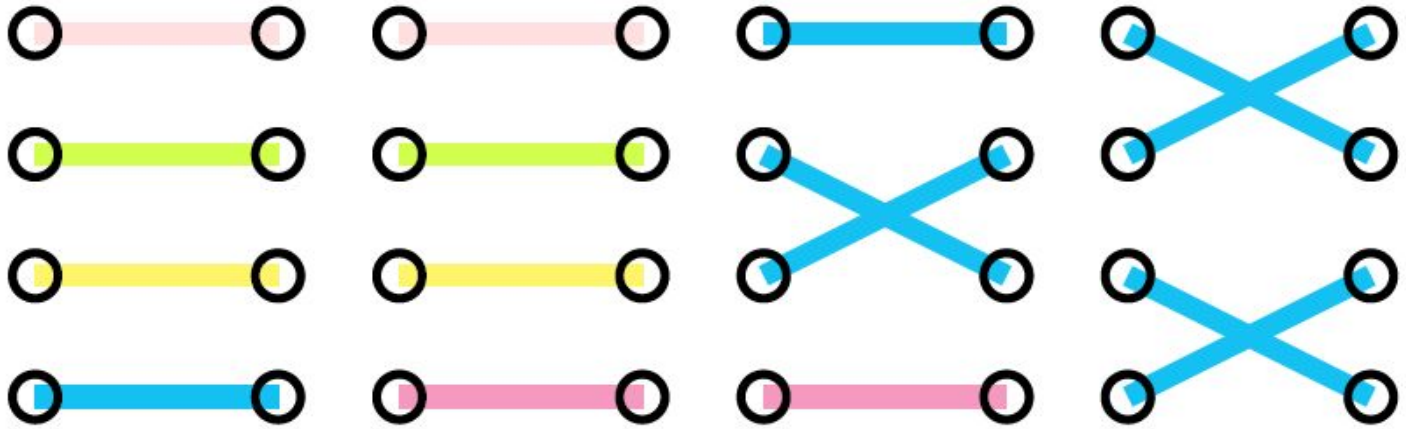
Visit here for Final Report

https://www.overleaf.com/project/5f241f9f97ab75000196e5d8

Problem Explanation

- A coflow consists of several individual jobs (flows)
- A job goes from an ingress (input) to an egress (output)
- Ingresses/egresses connected to master switch
  - Can only connect pairs of ingresses/egresses
  - (e.g. a telephone operator’s switch board)


Ingress Ports
Egress Ports
2 + ε
1
1
1
2 + ε
1
1
2 + ε
1
1
2 + ε
1
1
1
2
3
4
5
6
7
8
DC Fabric



$t = 1$ $t = 2$ $t = 3$ $t = 4$

$$\text{average CCT} = (3 \times 2 + 3 + 4)/5 = 2.6$$

(b) Sincronia

Problem Explanation (cont.)

- A coflow consists of several individual jobs (flows)
- A job goes from an ingress (input) to an egress (output)
- Ingresses/egresses connected to master switch
  - Can only connect pairs of ingresses/egresses
  - (e.g. a telephone operator's switch board)
- A coflow completes when all jobs complete
- Goal is to minimize the average coflow completion time
- Closest comparison is open job shop with recirculation

Coflow Ordering

- Bottleneck-Select-Scale-Iterate (BSSI) Algorithm
- Picks most bottlenecked port of remaining coflows
- Picks flow set with maximum weighted duration
- Corresponding coflow becomes next last ordering
- Weights are updated accordingly
- Terminates when last coflow is scheduled as first order

Algorithm 1 Bottleneck-Select-*Scale*-Iterate Algorithm

$\mathbb{C} = [n]$ ▷ Initial set of unscheduled coflows

procedure ORDER-COFLOWS(J)

for $k = n$ to 1 **do** ▷ Note ordering is from last to first

▷ **Find the most bottlenecked port**

$b \leftarrow \arg\max_p \sum_{c \in \mathbb{C}} d_c^p$

▷ **Select weighted largest job to schedule last**

$\sigma(k) \leftarrow \arg\min_{c \in \mathbb{C}} (w_c / d_c^b)$

▷ **Scale the weights**

$$w_c \leftarrow w_c - w_{\sigma(k)} \times \frac{d_c^b}{d_{\sigma(k)}^b} \quad \forall c \in \mathbb{C} \setminus \{\sigma(k)\}$$

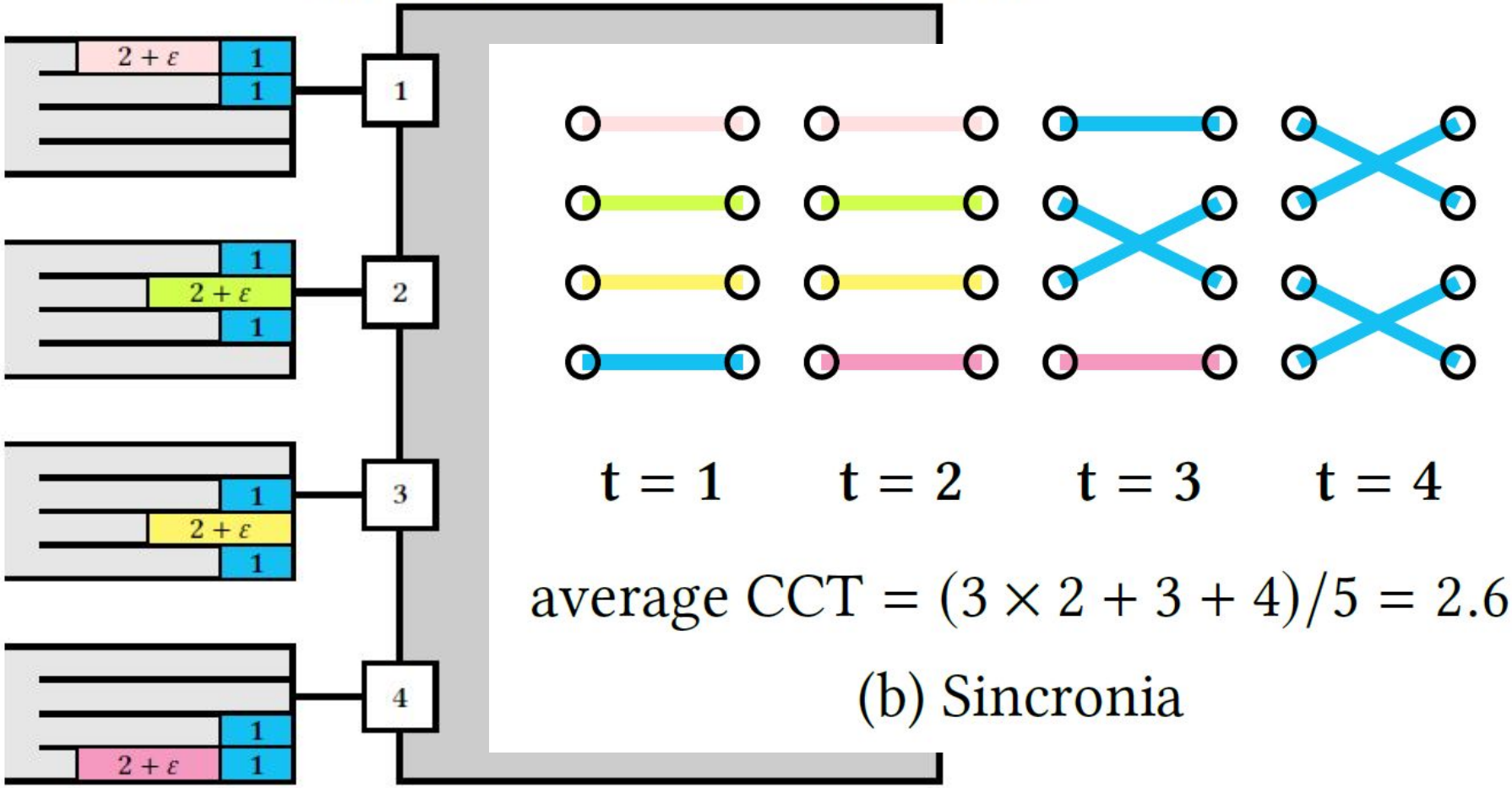
▷ **Iterate on updated set of unscheduled jobs**

$\mathbb{C} \leftarrow \mathbb{C} \setminus \{\sigma(k)\}$

return σ ▷ **Output the coflow ordering**

Job Scheduling

- Each ingress can only connect with one egress at once
- Next available ingress is iterated upon
- The job with the highest ordering priority is scheduled
  - Egress must be available



average CCT = $(3 \times 2 + 3 + 4)/5 = 2.6$

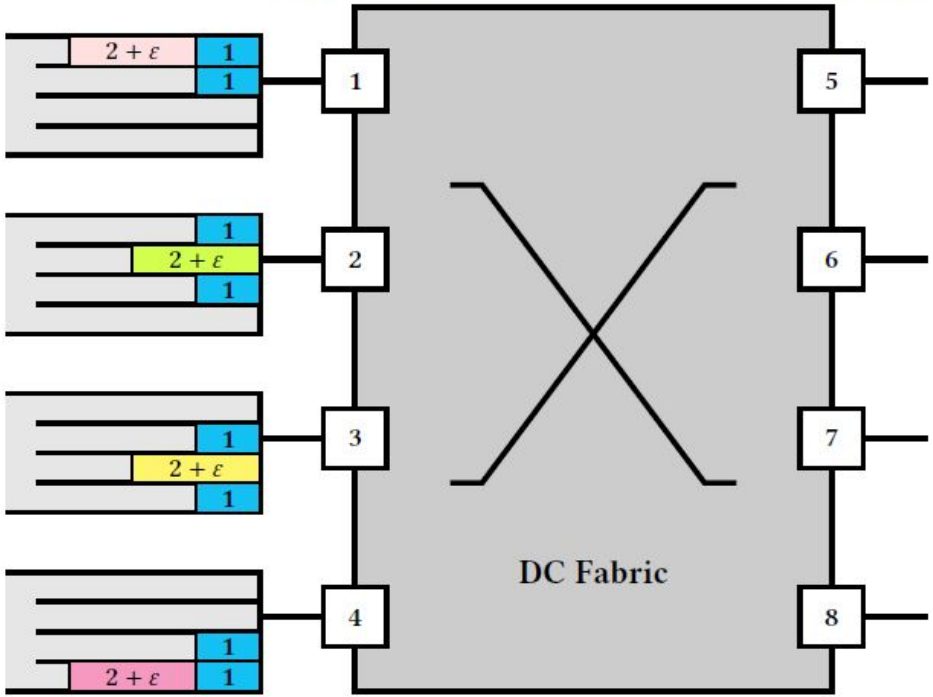
(b) Sincronia

Job Scheduling (cont.)

- Each ingress can only connect with one egress at once
- Next available ingress is iterated upon
- The job with the highest ordering priority is scheduled
  - Egress must be available
- Ingress availability time is updated to completion of job
  - Also if no egress is available for jobs available
- Iterates until all jobs are scheduled
- Average weighted coflow completion time is calculated

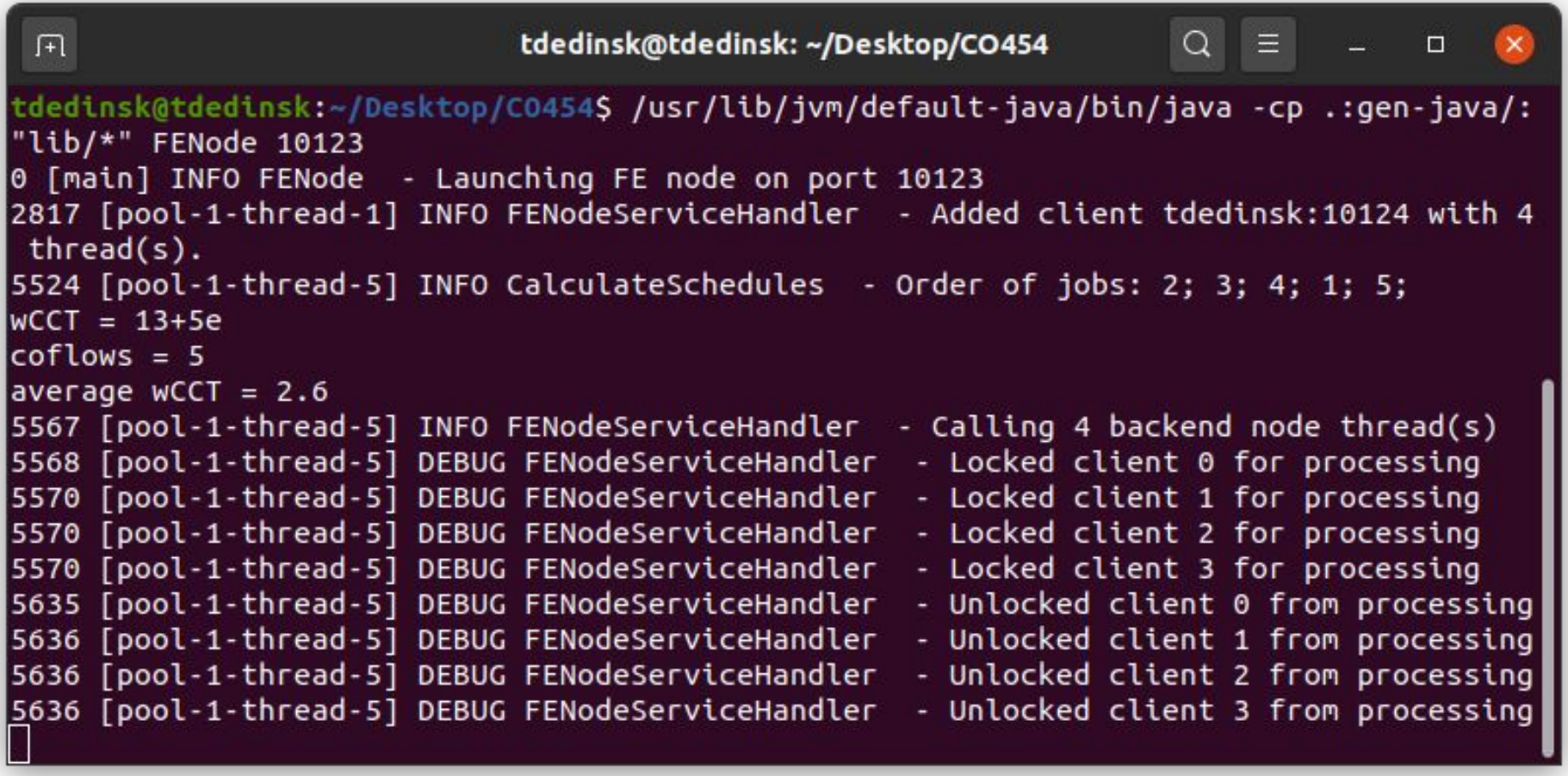
Implementation

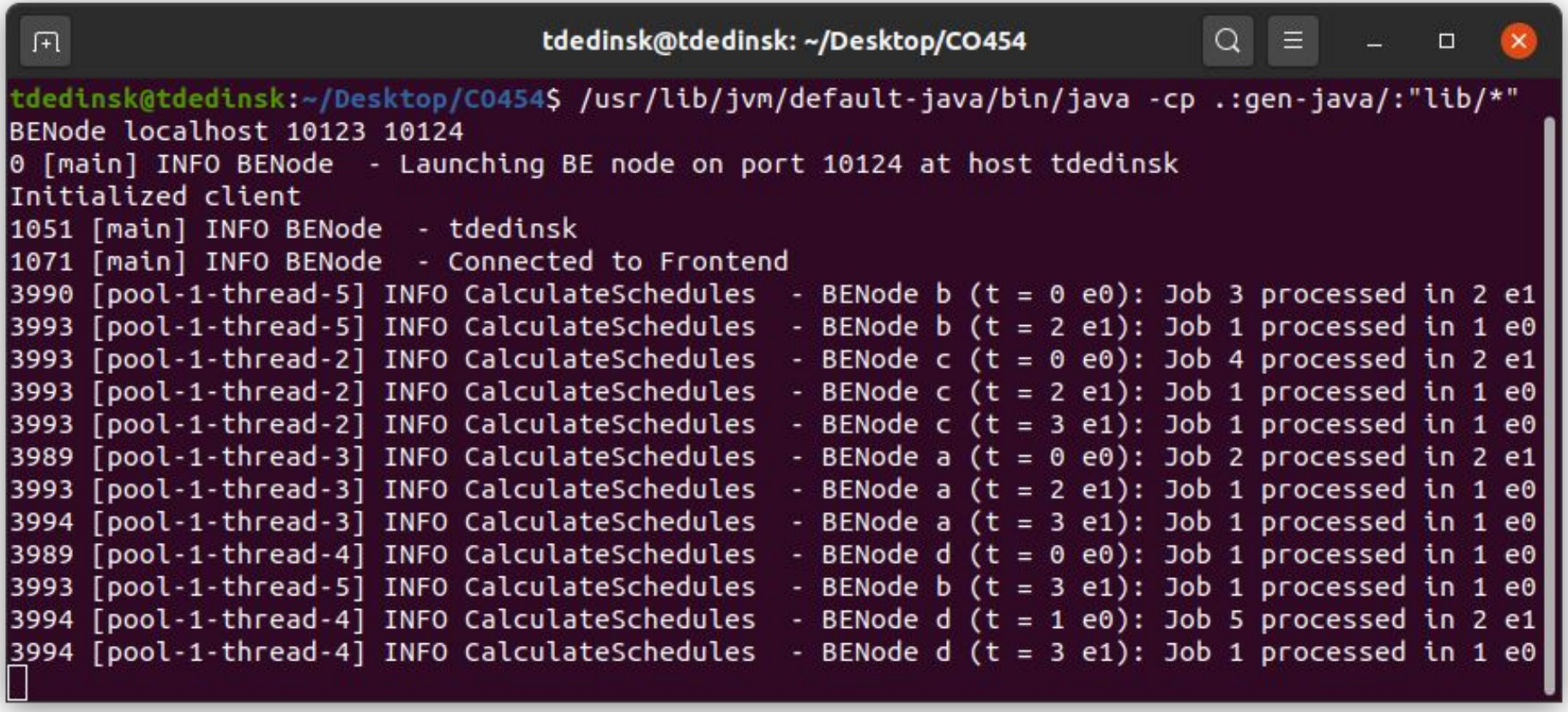
- Java 1.11 with Apache Thrift 0.13.0
- Three main processes emulating a network
  - Client sends sets of jobs for each ingress
  - BE Node receives sets of jobs for each egress
  - FE Node manages flow interchange



tdedinsk@tdedinsk: ~/Desktop/CO454

```
tdedinsk@tdedinsk:~/Desktop/CO454$ /usr/lib/jvm/default-java/bin/java -cp .:gen-
java/:"lib/*" Client localhost 10123 schedule.txt
tdedinsk@tdedinsk:~/Desktop/CO454$
```


tdedinsk@tdedinsk: ~/Desktop/CO454
tdedinsk@tdedinsk:~/Desktop/CO454$ /usr/lib/jvm/default-java/bin/java -cp .:gen-java/:"lib/*" FENode 10123
0 [main] INFO FENode - Launching FE node on port 10123
2817 [pool-1-thread-1] INFO FENodeServiceHandler - Added client tdedinsk:10124 with 4 thread(s).
5524 [pool-1-thread-5] INFO CalculateSchedules - Order of jobs: 2; 3; 4; 1; 5;
wCCT = 13+5e
coflows = 5
average wCCT = 2.6
5567 [pool-1-thread-5] INFO FENodeServiceHandler - Calling 4 backend node thread(s)
5568 [pool-1-thread-5] DEBUG FENodeServiceHandler - Locked client 0 for processing
5570 [pool-1-thread-5] DEBUG FENodeServiceHandler - Locked client 1 for processing
5570 [pool-1-thread-5] DEBUG FENodeServiceHandler - Locked client 2 for processing
5570 [pool-1-thread-5] DEBUG FENodeServiceHandler - Locked client 3 for processing
5635 [pool-1-thread-5] DEBUG FENodeServiceHandler - Unlocked client 0 from processing
5636 [pool-1-thread-5] DEBUG FENodeServiceHandler - Unlocked client 1 from processing
5636 [pool-1-thread-5] DEBUG FENodeServiceHandler - Unlocked client 2 from processing
5636 [pool-1-thread-5] DEBUG FENodeServiceHandler - Unlocked client 3 from processing

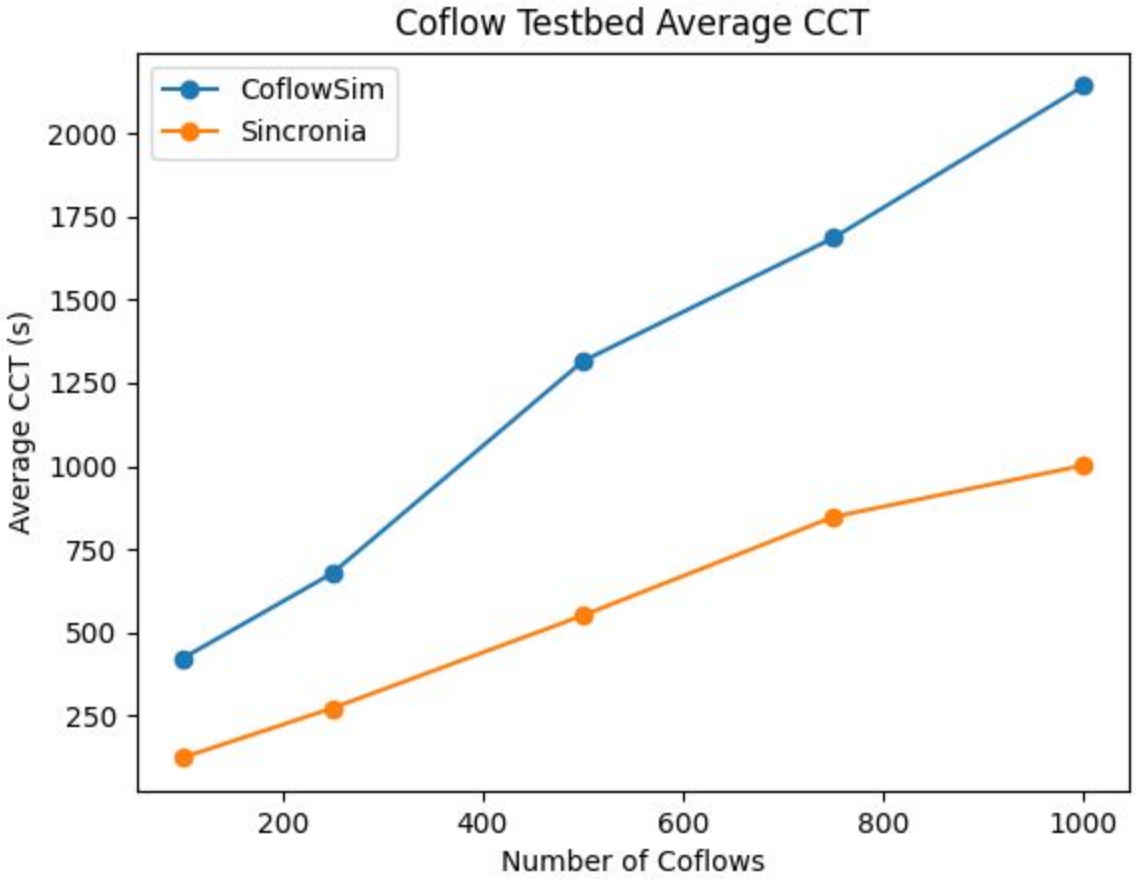

tdedinsk@tdedinsk: ~/Desktop/CO454
tdedinsk@tdedinsk:~/Desktop/CO454$ /usr/lib/jvm/default-java/bin/java -cp .:gen-java/:"lib/*"
BENode localhost 10123 10124
0 [main] INFO BENode - Launching BE node on port 10124 at host tdedinsk
Initialized client
1051 [main] INFO BENode - tdedinsk
1071 [main] INFO BENode - Connected to Frontend
3990 [pool-1-thread-5] INFO CalculateSchedules - BENode b (t = 0 e0): Job 3 processed in 2 e1
3993 [pool-1-thread-5] INFO CalculateSchedules - BENode b (t = 2 e1): Job 1 processed in 1 e0
3993 [pool-1-thread-2] INFO CalculateSchedules - BENode c (t = 0 e0): Job 4 processed in 2 e1
3993 [pool-1-thread-2] INFO CalculateSchedules - BENode c (t = 2 e1): Job 1 processed in 1 e0
3993 [pool-1-thread-2] INFO CalculateSchedules - BENode c (t = 3 e1): Job 1 processed in 1 e0
3989 [pool-1-thread-3] INFO CalculateSchedules - BENode a (t = 0 e0): Job 2 processed in 2 e1
3993 [pool-1-thread-3] INFO CalculateSchedules - BENode a (t = 2 e1): Job 1 processed in 1 e0
3994 [pool-1-thread-3] INFO CalculateSchedules - BENode a (t = 3 e1): Job 1 processed in 1 e0
3989 [pool-1-thread-4] INFO CalculateSchedules - BENode d (t = 0 e0): Job 1 processed in 1 e0
3993 [pool-1-thread-5] INFO CalculateSchedules - BENode b (t = 3 e1): Job 1 processed in 1 e0
3994 [pool-1-thread-4] INFO CalculateSchedules - BENode d (t = 1 e0): Job 5 processed in 2 e1
3994 [pool-1-thread-4] INFO CalculateSchedules - BENode d (t = 3 e1): Job 1 processed in 1 e0

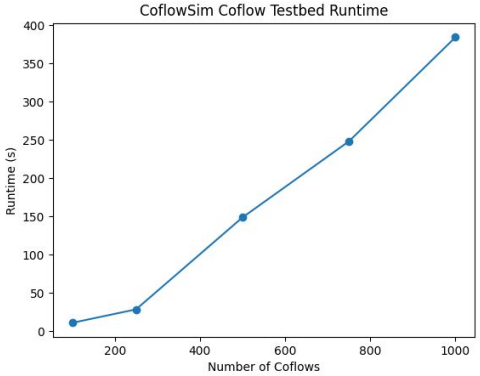
Brute Force Solver

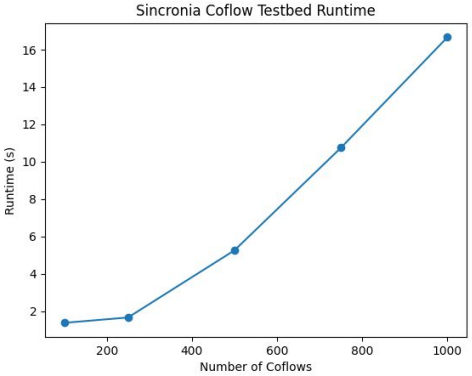
- Made to verify how optimal approximation was
- Each egress schedule has permutations of job order
- Combine all permutations to explore all possibilities
- Diverge at each job selection based on egress priority
- Leads to way too many possible schedules
- Recursive and iterative versions have memory issues

Examples

- Tested and verified on toy example in paper
  - Sincronia gets awCCT of 2.6, optimal gets 2.4
- Other edge cases were tested
  - Missing ingresses, egresses, etc.
- Four main batches of tests were analyzed
  - Comparison between Sincronia and Varys

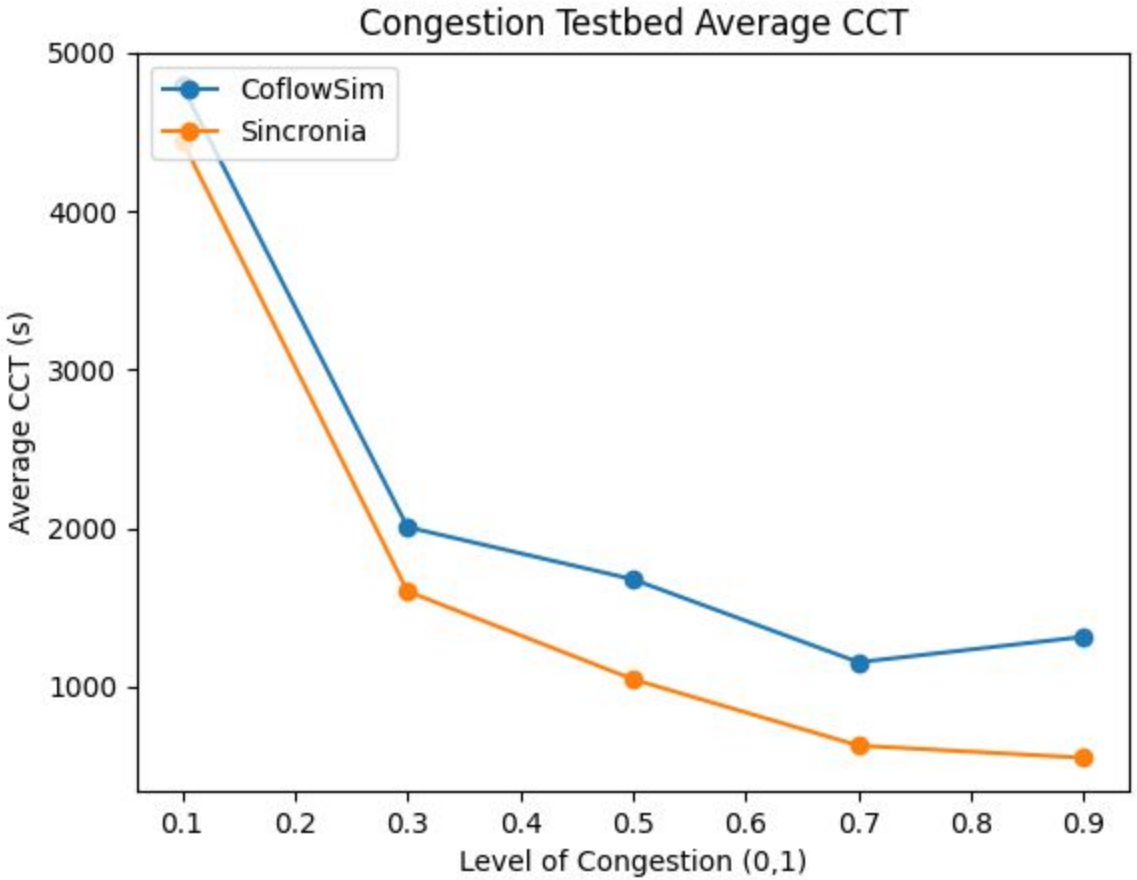


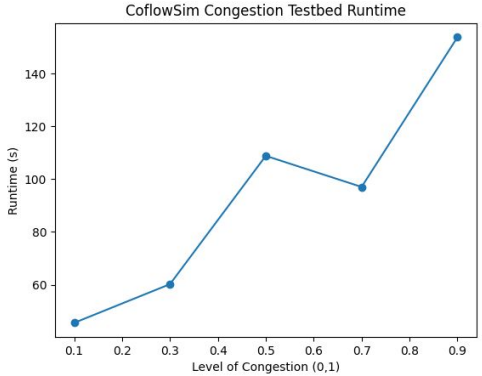


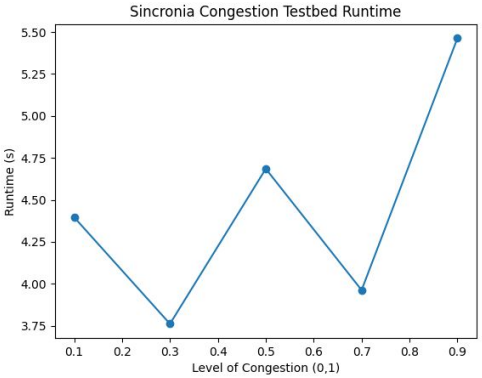


Left: Comparison; Right: Duration

Testbed for variable number of coflows (0.9 load, 0.5 contention, 20 max coflow width, 150 ports)

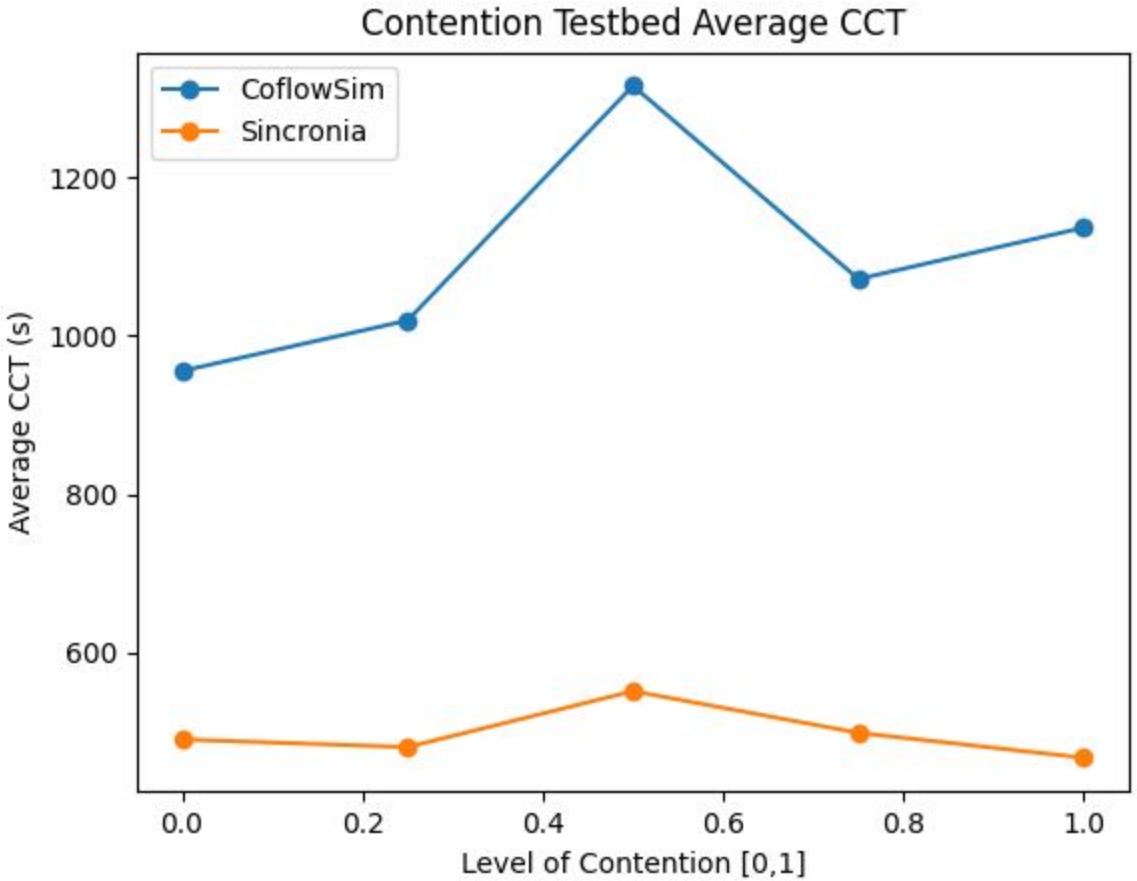


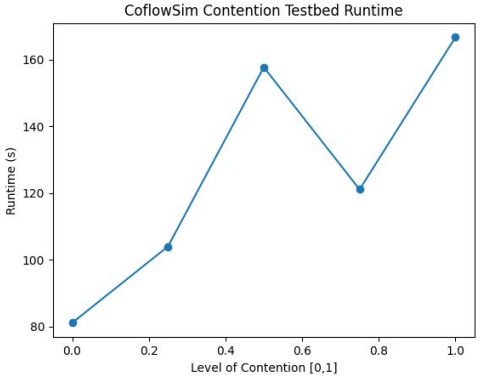


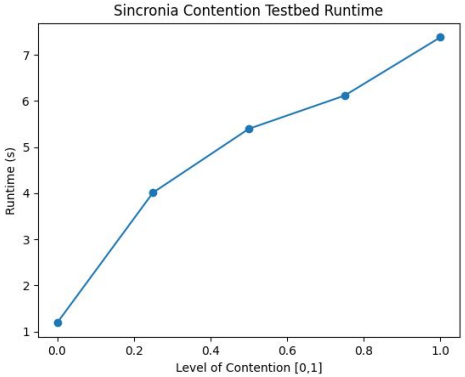


Left: Comparison; Right: Duration

Testbed for variable level of network congestion (500 coflows, 0.5 contention, 20 max coflow width, 150 ports)

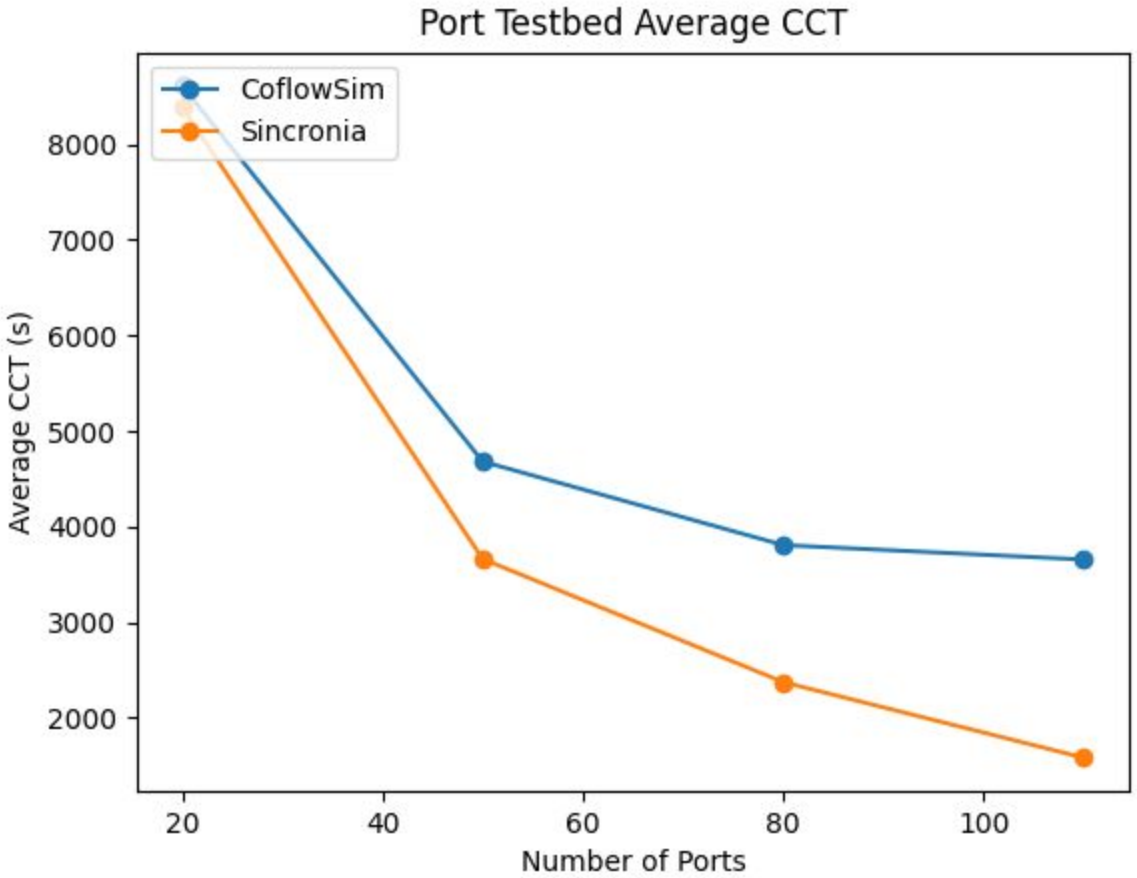


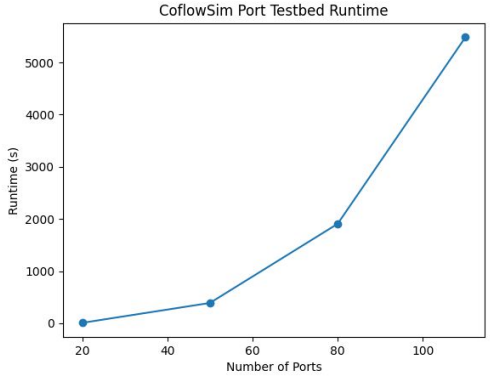


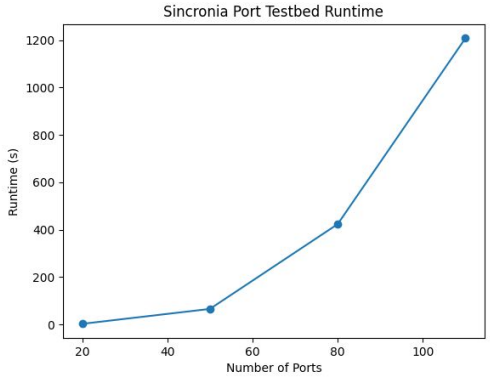


Left: Comparison; Right: Duration

Testbed for variable level of port contention (500 coflows, 0.9 load, 20 max coflow width, 150 ports)







Left: Comparison; Right: Duration

Testbed for variable number of ports (500 coflows, 0.9 load, 0.5 contention, max coflow width = # of ports)