Efficient Switch Migration for Controller Load Balancing in Software Defined Networking

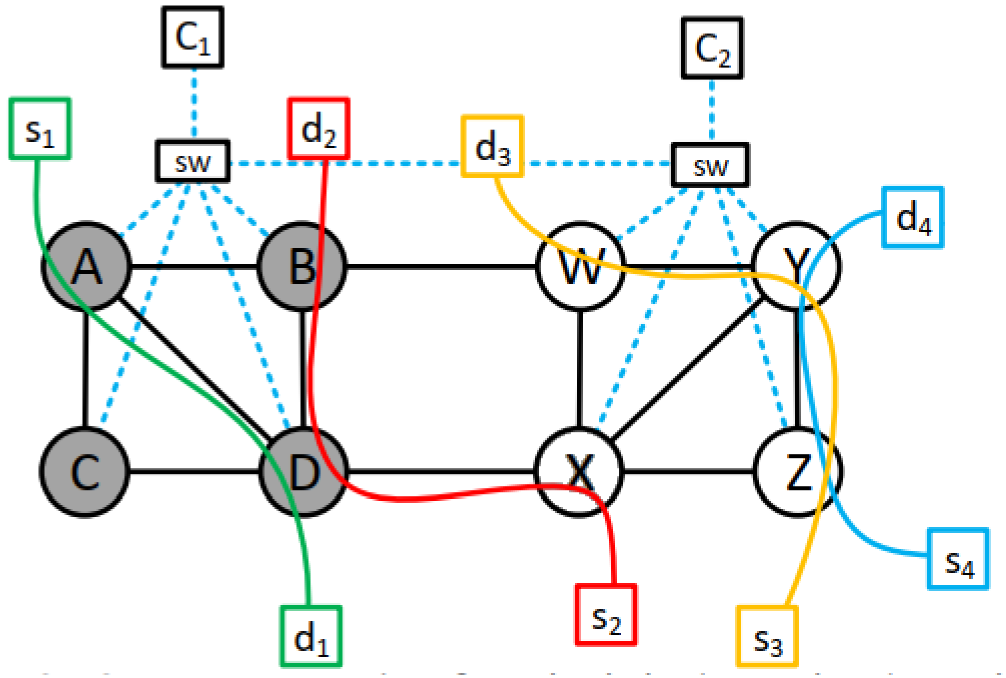
Rajorshi Biswas and Jie Wu
Dept. of Computer and Info. Sciences
Temple University

Outline

- Switch migration and System Model
- Some existing works
- Problem Definition: Minimizing Cost
- Different solutions
- Simulation and Experimental results
- Q&A

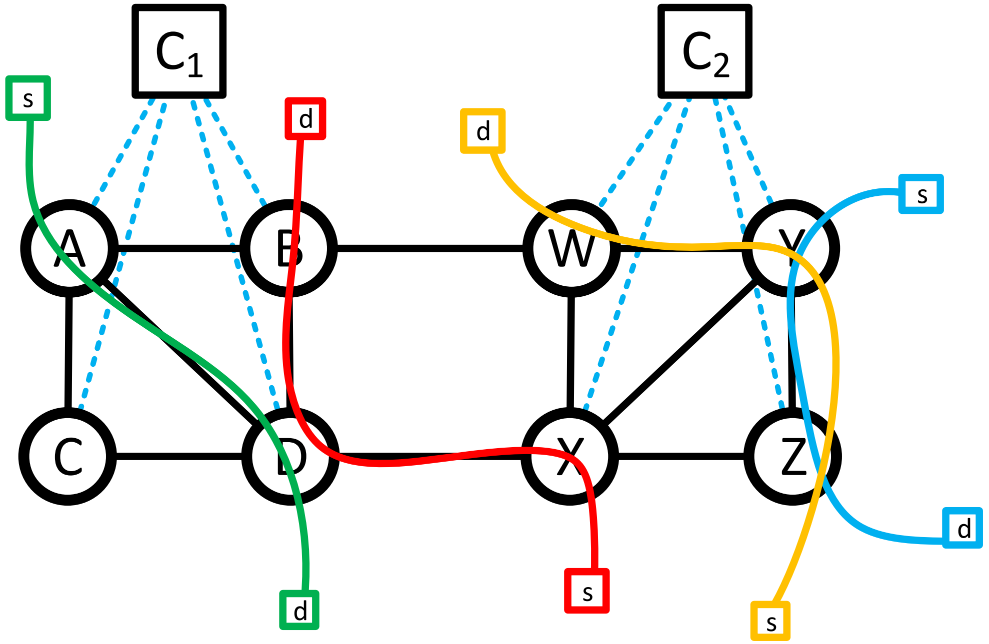
System Model

- Switch Migration
  - Changing the controller of an SDN switch
- Controller Load
  - Path finding requests
  - Intermediate node query requests
- Response Delay:
  - # of hops to controller
  - Controller load
- Green Flow
  - path construction (A) + intermediate query (D)
- Red Flow
  - path construction (X) + path construction (D) + intermediate query (B)



Switch Migration is Challenging

- Challenges
  - Sporadic assignment leads to higher number of path construction.
  - Flows change frequently.
  - Live migration is not possible.



Previous Works

Systems	**Limitations**
ILP based Systems: • X. Zhang, L. Li and C. -b. Yan, "Robust Controller Placement Based on Load Balancing in Software Defined Networks," ICNSC, 2020 • L. Li, N. Du, H. Liu, R. Zhang and C. Yan, "Towards robust controller placement in software-defined networks against links failure," 2019 IFIP/IEEE Symposium on Integrated Network and Service Management.	• ILP based solutions takes long time in large topologies. • Does not consider dynamic/incremental adjustment.
Heuristic/Greedy • F. He and E. Oki, "Load Balancing Model against Multiple Controller Failures in Software Defined Networks," ICC 2020.	• Does not consider the control network delay. • Dynamic/incremental adjustments is not considered.

Problem: Minimize Cost of Assignment

- Cost is a weighted sum of three metrics
  - $P(A,c)$ number of path construction request to c.
  - $Q(A,c)$ number of intermediate query requests to c.
  - $D(A,c)$ total number of hops from each switch to c.
  - $C(A,c) = \omega_1 P(A,c) + \omega_2 Q(A,c) + \omega_3 D(A,c)$
- $C(A) = \sum C(A,c)$

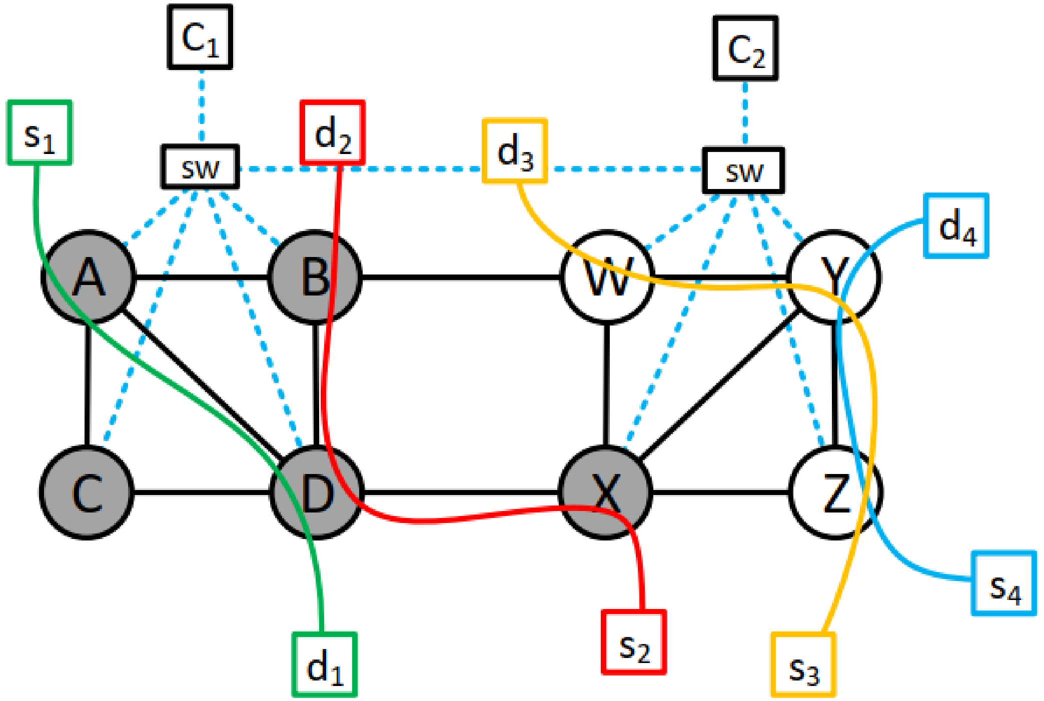
- Problem:
  - Find a Switch-Controller Assignment that minimizes cost.
- Constraints:
  - Controller capacity constraints
  - Switch migration can be only to neighbors
- Two Scenarios:
  - Initial deployment
    - Greedy
    - Clustering
  - Incremental deployment
    - Greedy

NP-Hard, Graph Partitioning Problem

Initial deployment: Minimize Cost

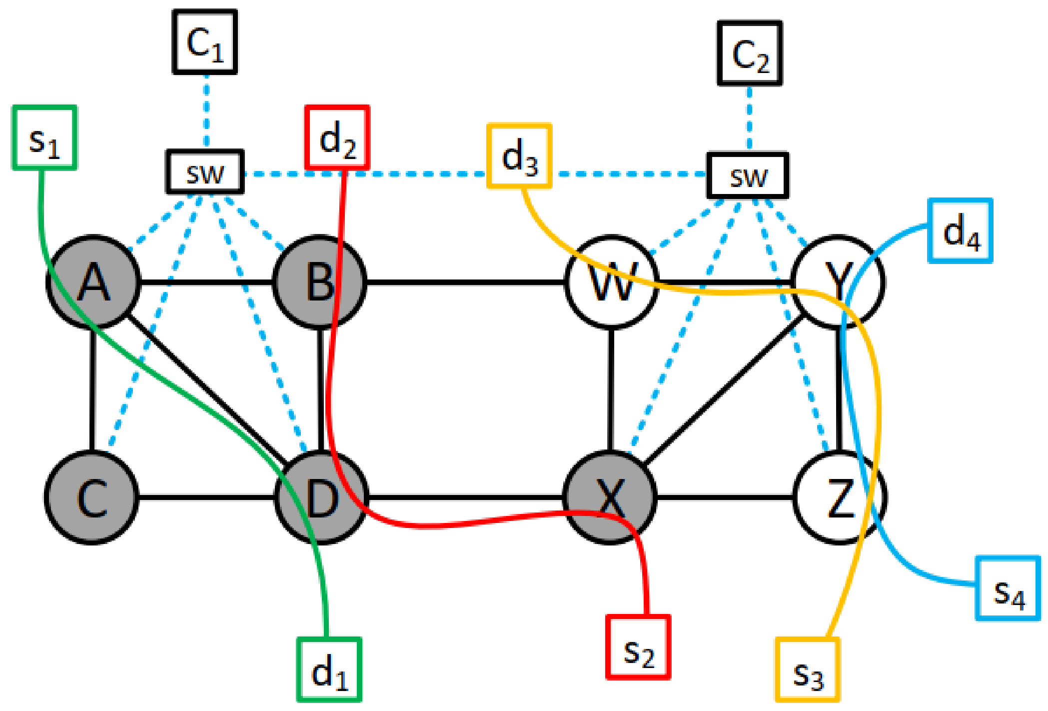
- Greedy Solution:
  - Consider a bucket for each controller.
  - Initially, add the switch to the bucket which produce minimum amount of cost.
  - Consider the neighbors for future extension.
  - Add a switch from the neighbors that produce minimum cost.
- Complexity:

  $O(|C|(|V|^2 + |V||F|))$



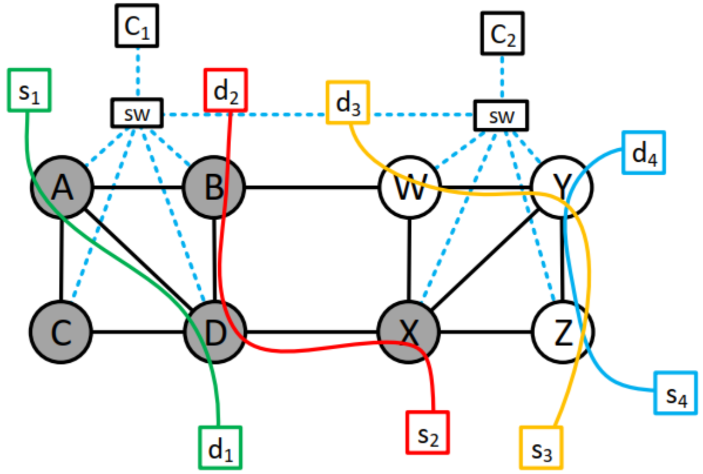
Initial deployment: Minimize Cost

- An Example:
- First round:
  - [A] [W]
  - Candidates [B, C, D] [B, X, Y]
  - C->C_1 is the minimum cost
- Second round:
  - [A, C] [W]
  - Candidates [B,D] [B, X, Y]
- Final Round:
  - [A,B,C,D,X] [W,Y,Z]



Initial deployment: Minimize Cost

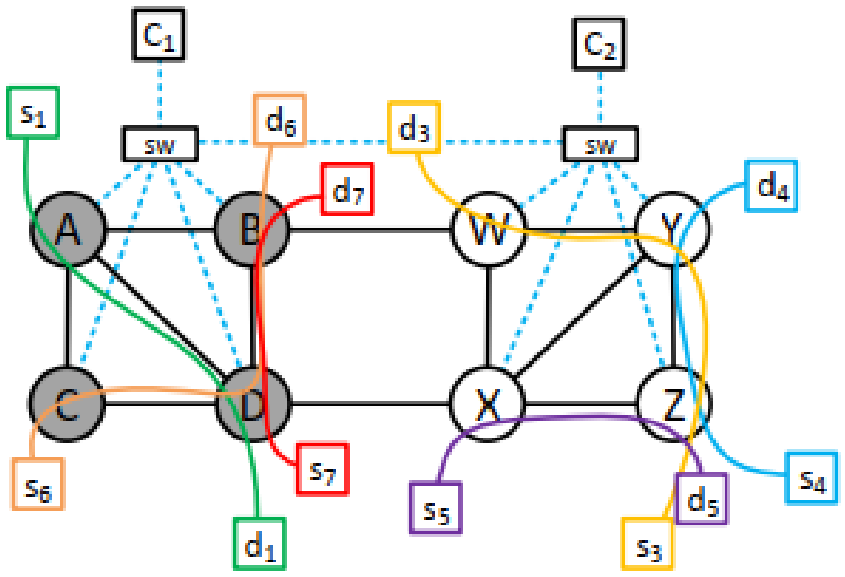
- Clustering Solution:
  - Create distance matric from the topology
  - This distance matrix is normalized and used for hierarchical clustering.
  - We set the number of clusters as the number of controller.
  - Each cluster is assigned to the controller that produces minimum cost.
- Complexity: $O(|V|^3)$
- Example:
  - [A,B,C,D,X] [W,Y,Z]



	A	B	C	D	W	X	Y	Z	C_1	C_2
A	0	1	1	1	2	2	3	3	2	3
B	1	0	2	1	1	2	2	3	2	3
C	1	2	0	1	3	2	3	3	2	3
D	1	1	1	0	2	1	2	2	2	3
W	2	1	3	2	0	1	1	2	3	2
X	2	2	2	1	1	0	1	1	3	2
Y	3	2	3	2	1	1	0	1	3	2
Z	3	3	3	2	2	1	1	0	3	2
C_1	2	2	2	2	3	3	3	3	0	3
C_2	3	3	3	3	2	2	2	2	3	0

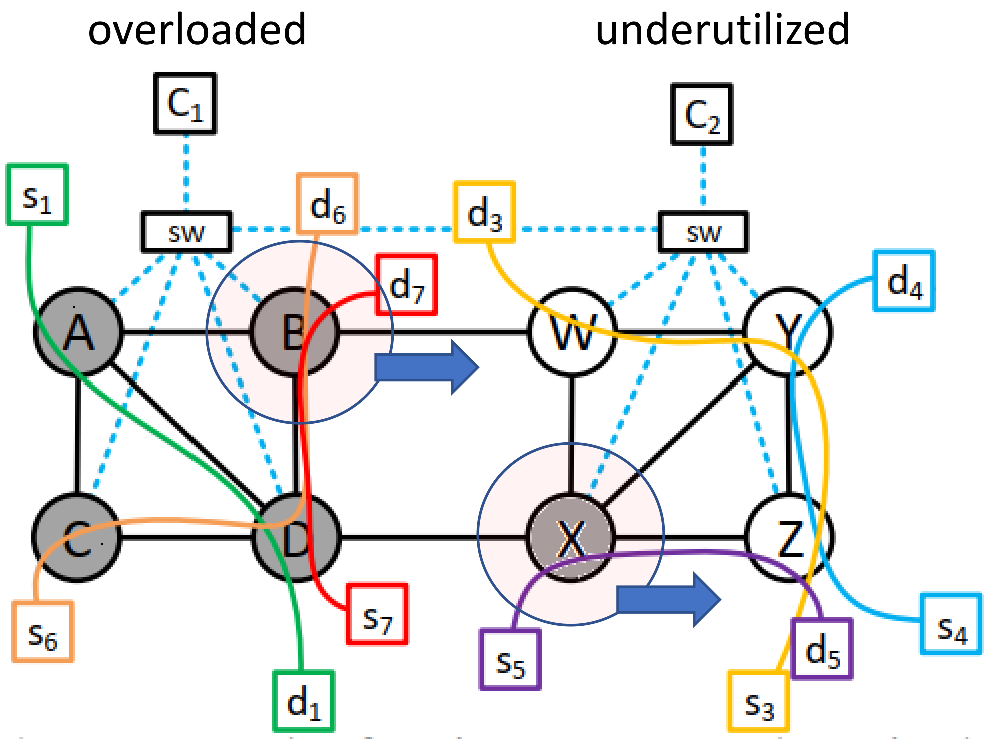
Incremental Deployment

- Problem:
  - Find a Switch-Controller Assignment that minimizes cost.
- Constraints
  - Controller capacity constraints
  - Old switch assignment-new switch assignment < K
  - Switch migration can be only to neighbors



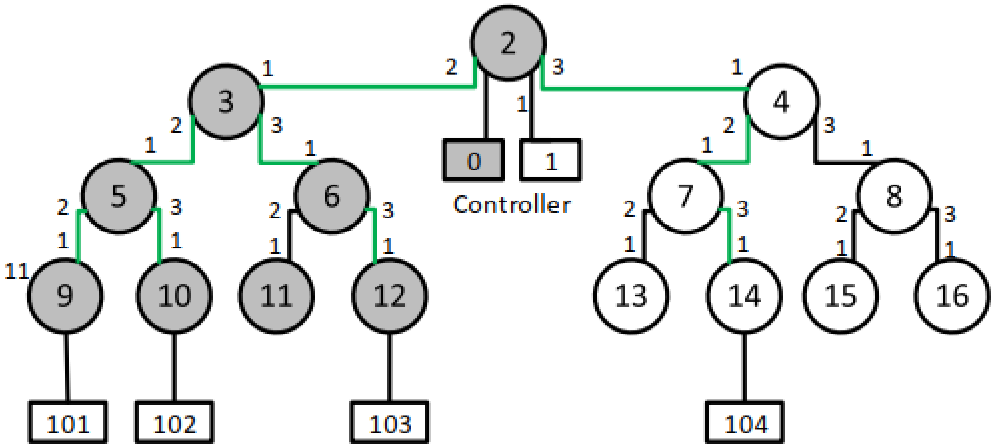
Incremental Deployment Solution

- Greedy:
  - Find overloaded and underutilized controllers. $C_u \cup C_o = C$
  - Find the neighbors of C_o that belongs to C_u
  - Calculate the benefit of migration for each neighbors.
  - $Benefit\ of\ migration = pre\ mig.cost - after\ mig.cost$
  - Choose the neighbors with max benefit.
  - Continue K times or until every is balanced.
- Complexity: $O(|F||V|K))$

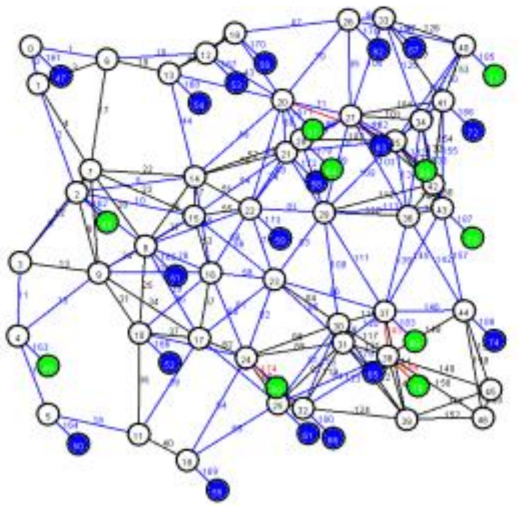


Migration of X is more beneficiary than migration of B

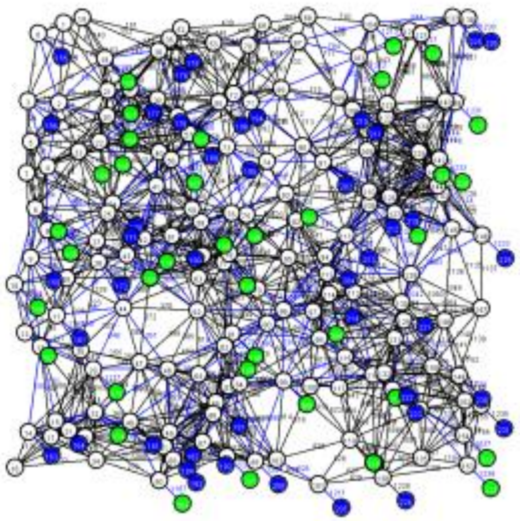
Experiments and Simulations



Migration delay: 5.2s

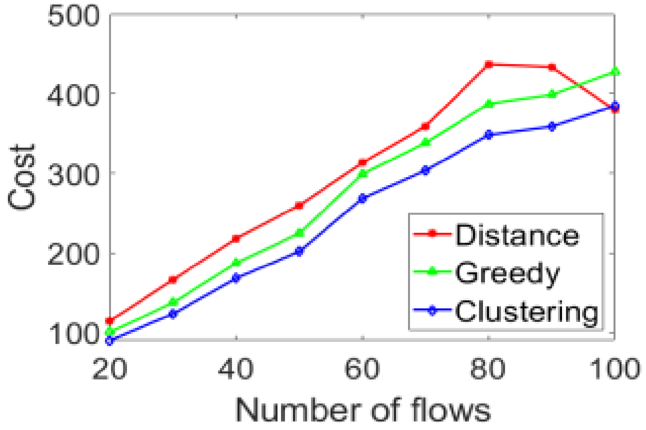


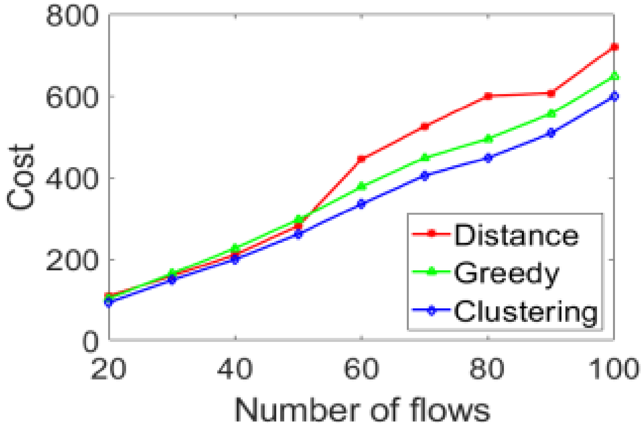
Sparse T1



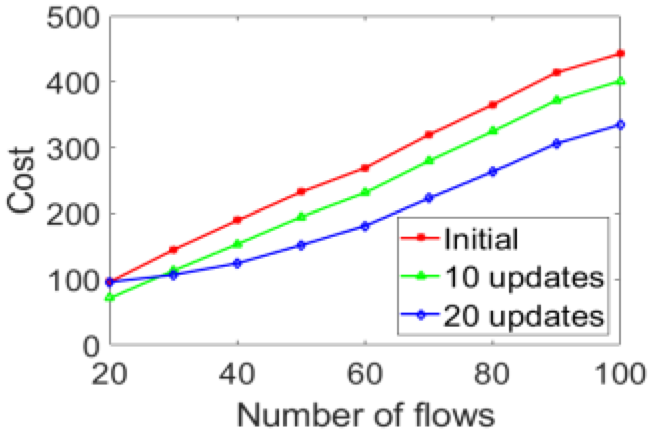
Dense T2

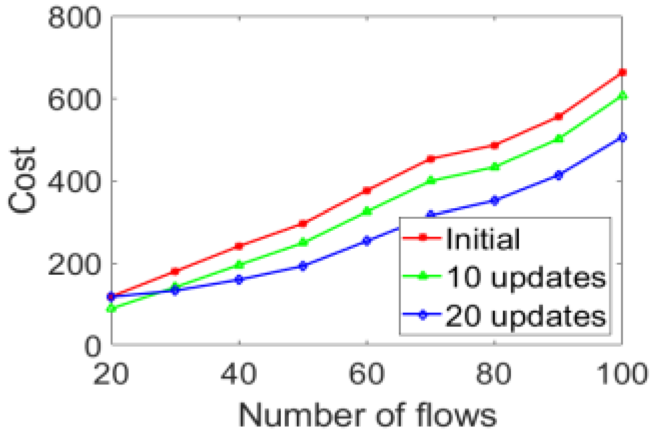
Simulation Results





Initial Deployment:
Distance based has the highest cost
Greedy is 10% lower and
Clustering is 20% lower than distance based





Incremental Deployment:
Distance based has the highest cost
10 updates is 11% lower and
20 updates is 24% lower than distance based

Sparse T1

Dense T2

Thank You

Q&A