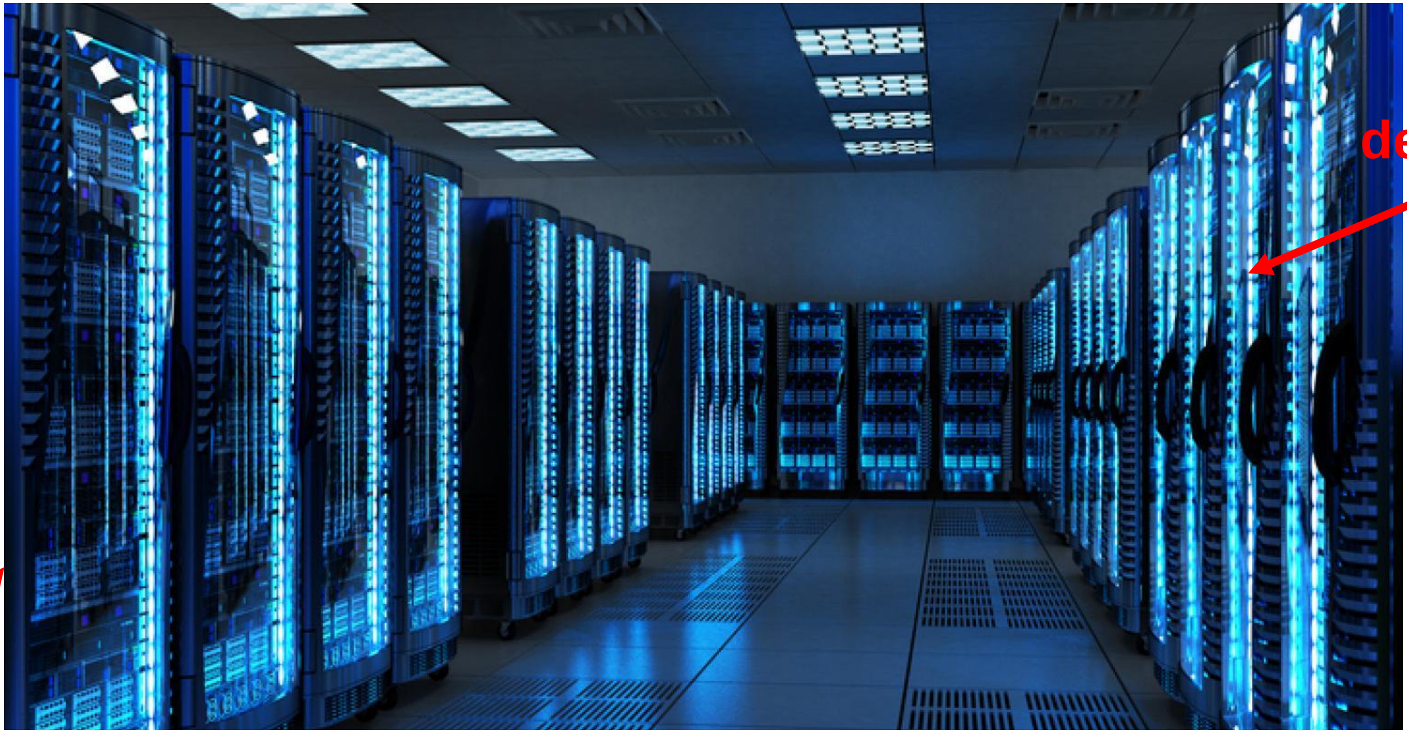
Data Structures
Programming Project #2

Data Center

- A data center consists of multiple severs
- The servers are connected by switches in a local area network



destination

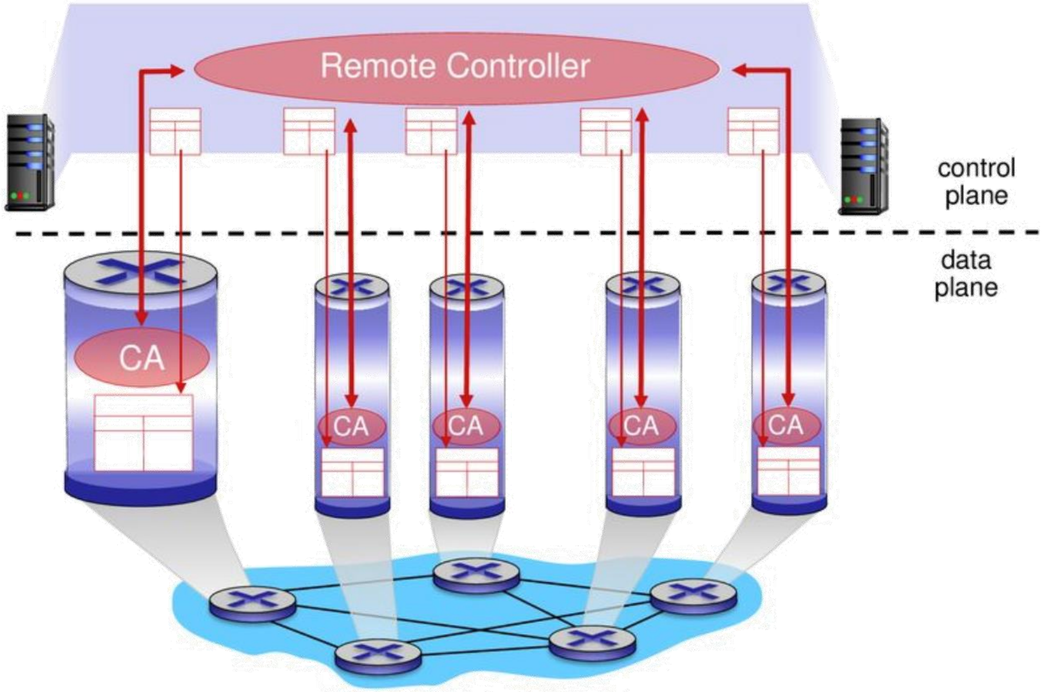
source

Switches

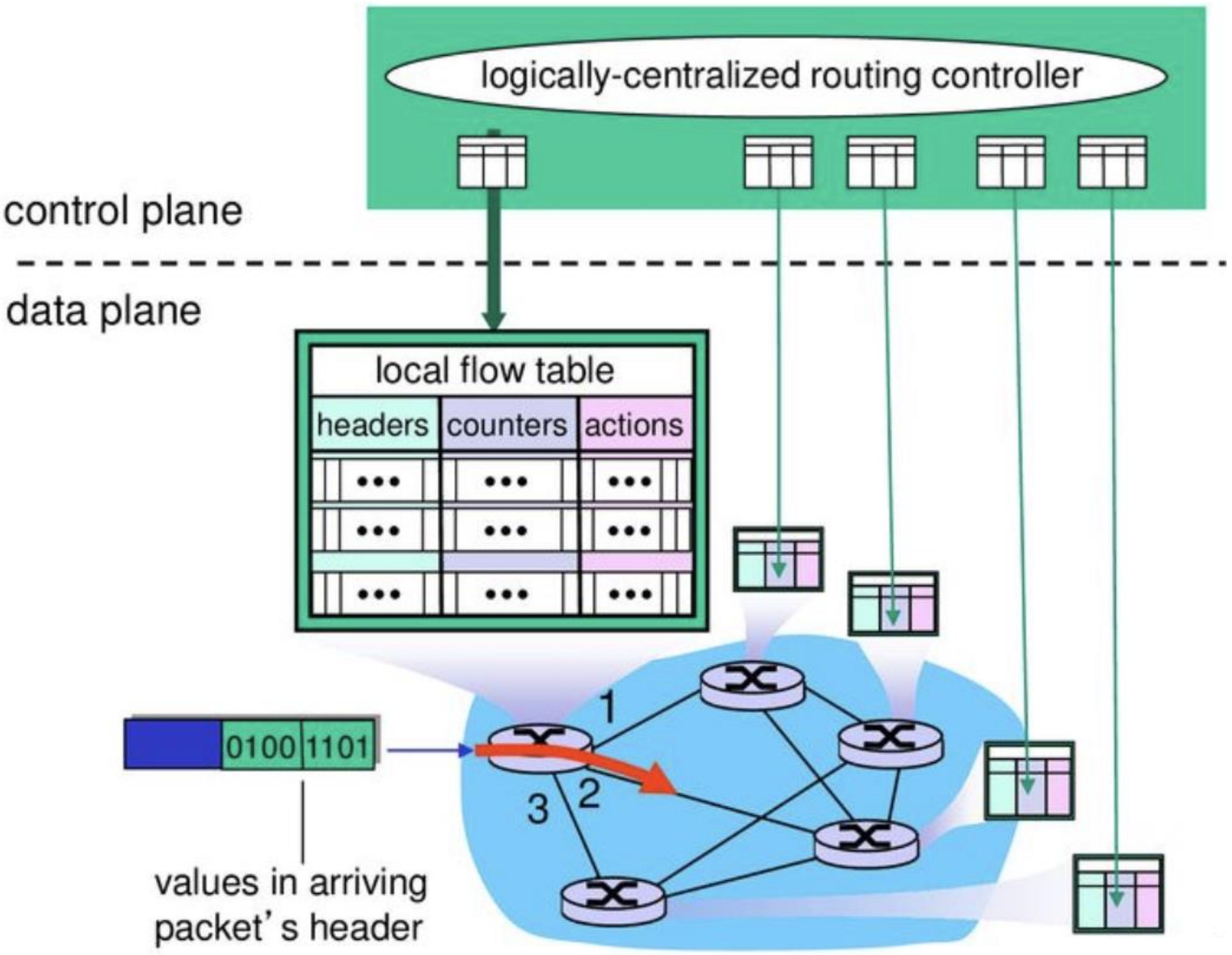
- Each switch has multiple ports
- Receive and forward the packets from a port to another port

SDN-enabled Switches

- A centralized controller is introduced – software-defined networking (SDN)

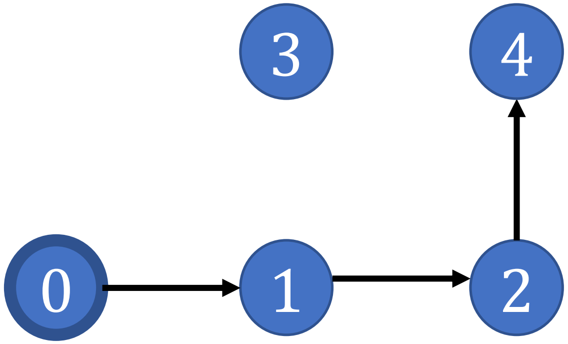


Installing Rules in the SDN-enabled Switches

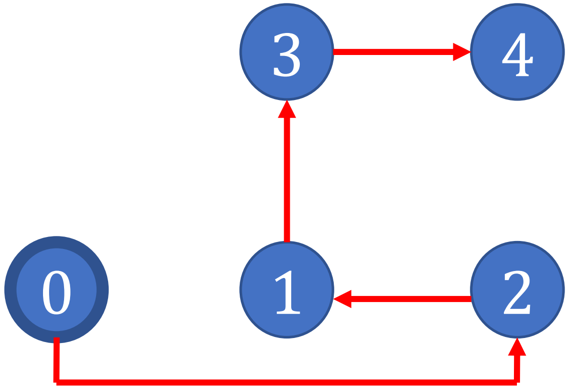


Routing Path Update (aka Network Update)

- Given the **old** and **new** routing paths
- Update the routing paths



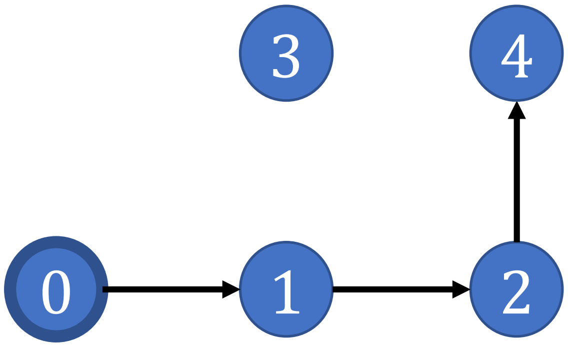
0	1	2	3	4
1	2	4	-1	-1



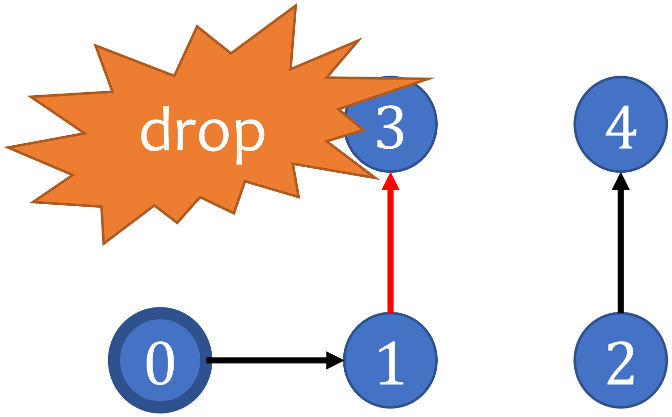
0	1	2	3	4
2	3	1	4	-1

Difficulty of Network Update in SDN

- The controller is logically-centralized
- However, the underlying mechanism is distributed
- Each switch receives the update message and updates its rule independently and asynchronously



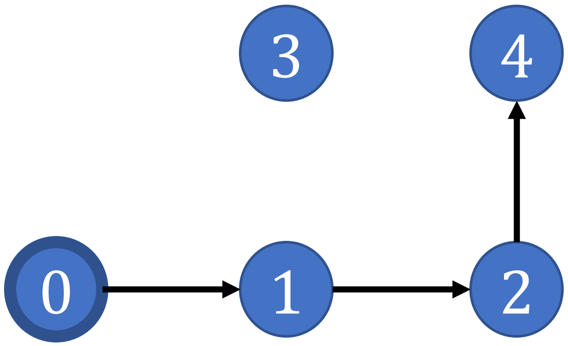
0	1	2	3	4
1	2	4	-1	-1



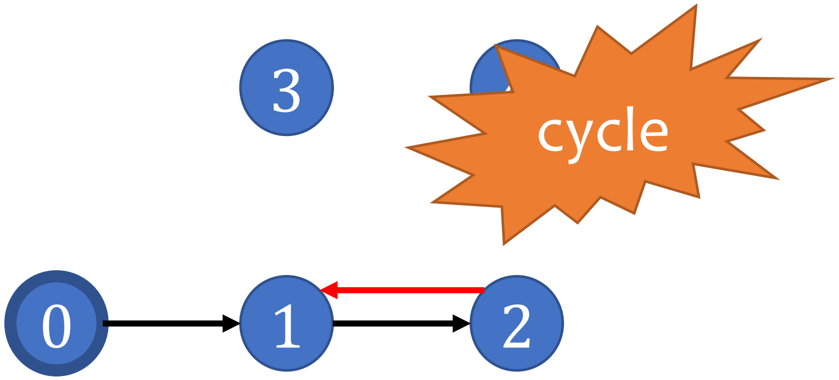
0	1	2	3	4
1	3	4	-1	-1

Difficulty of Network Update in SDN

- The controller is logically-centralized
- However, the underlying mechanism is distributed
- Each switch receives the update message and updates its rule independently and asynchronously



0	1	2	3	4
1	2	4	-1	-1



0	1	2	3	4
1	3	4	-1	-1

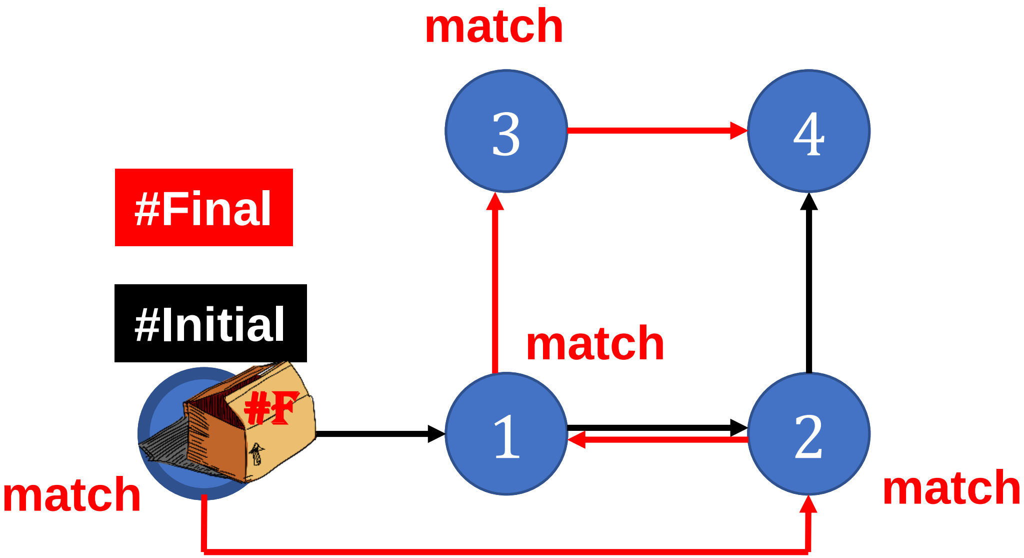
Difficulty of Network Update in SDN

- The controller is logically-centralized
- However, the underlying mechanism is distributed
- Each switch receives the update message and updates its rule independently and asynchronously

- How to solve the issue?
- Two-phase commit
- Round-based update

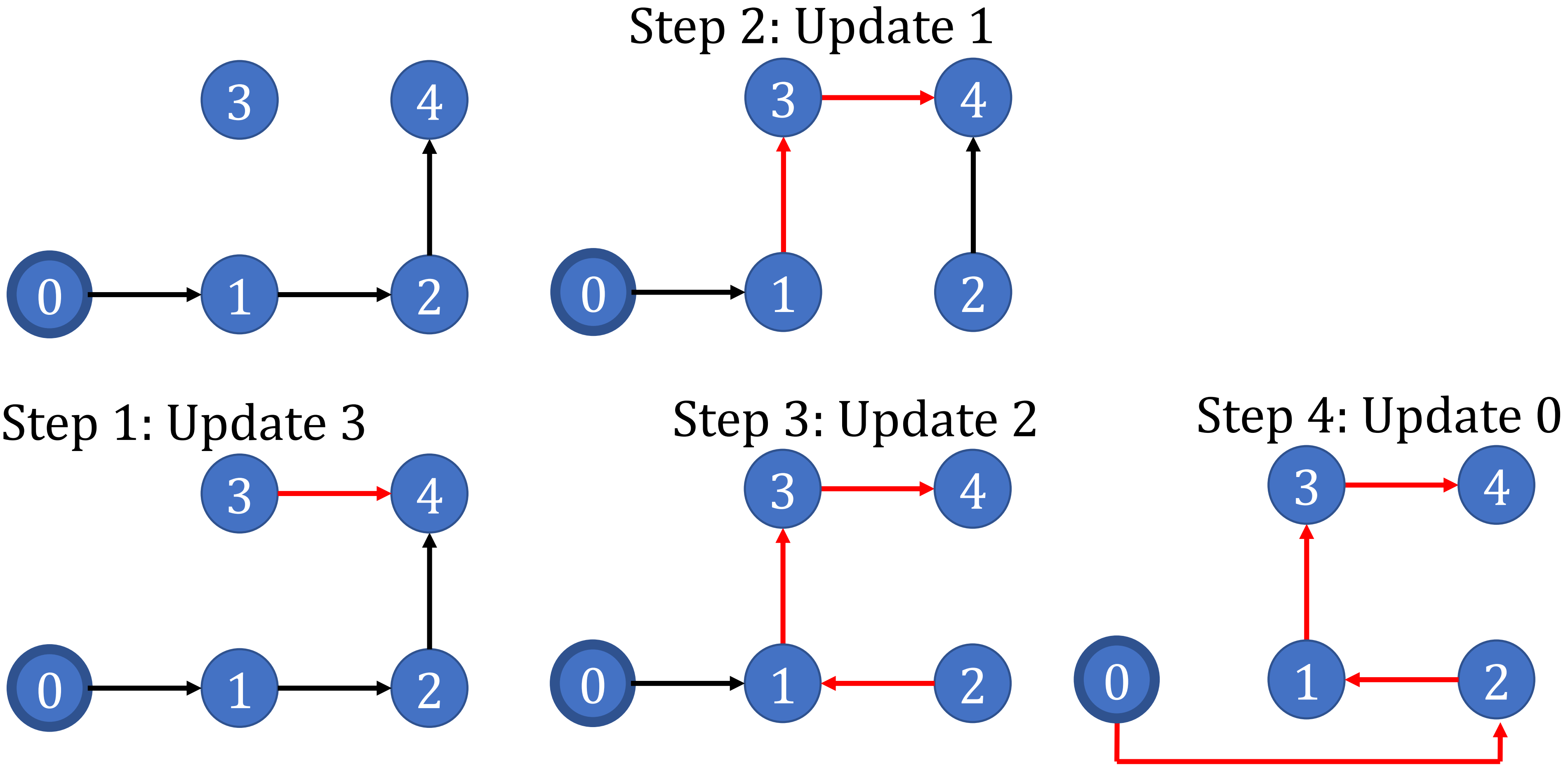
Solution of Network Update in SDN

- Two-phase commit
- Drawback: waste the TCAM size during the update



Solution of Network Update in SDN

- Round-based update (1st attempt)



Step 1: Update 3

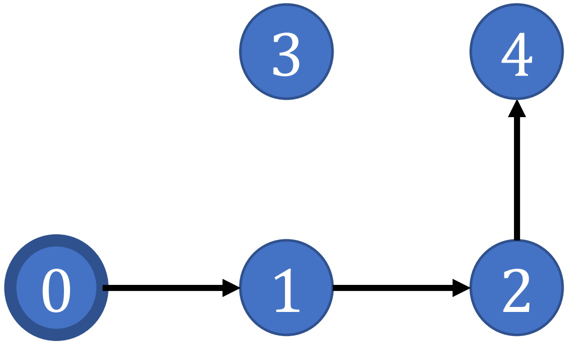
Step 2: Update 1

Step 3: Update 2

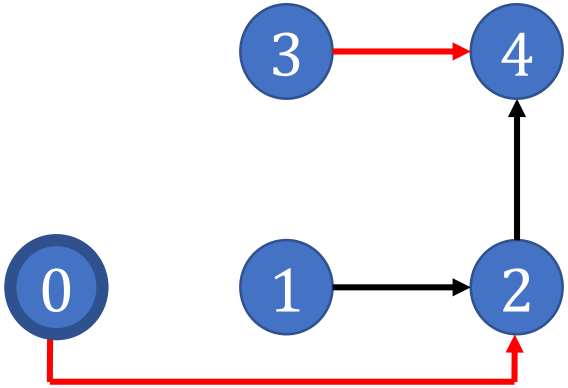
Step 4: Update 0

Solution of Network Update in SDN

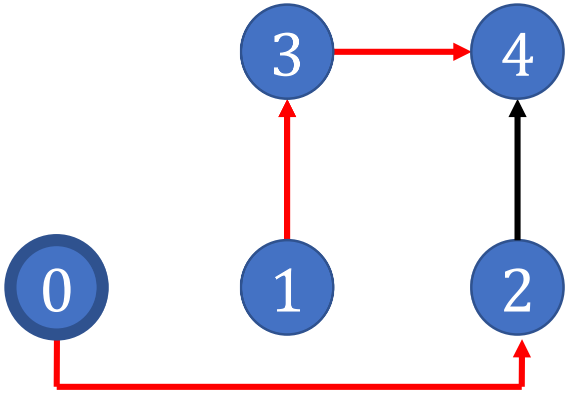
- Round-based update (2nd attempt)



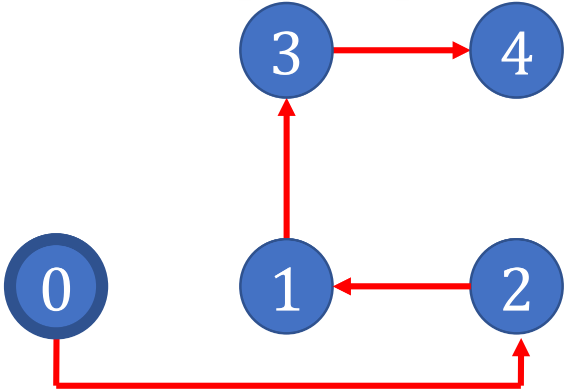
Step 1: Update 0 and 3



Step 2: Update 1

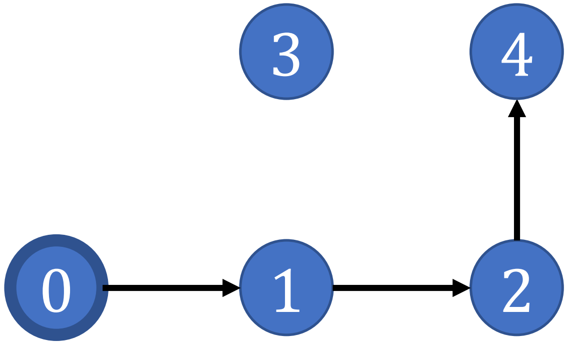


Step 3: Update 2

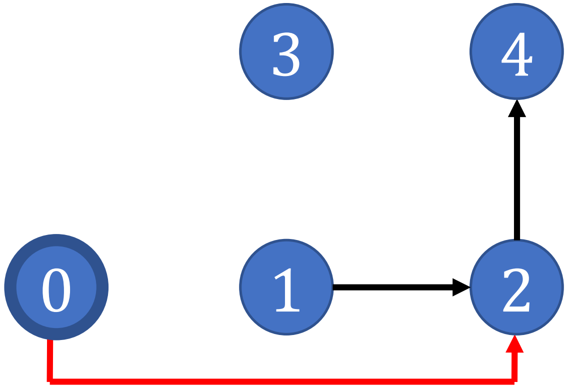


Solution of Network Update in SDN

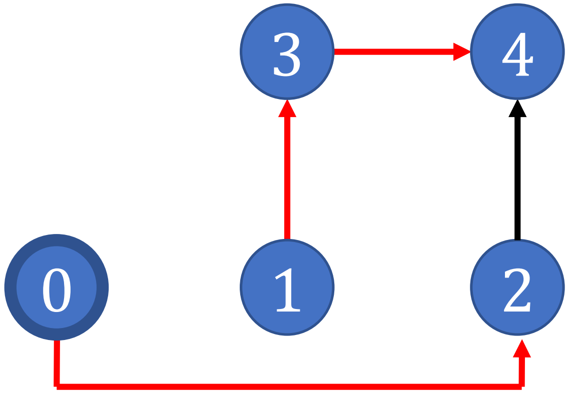
- Round-based update (3rd attempt)



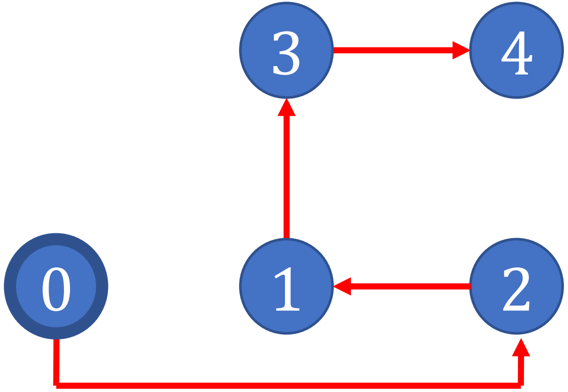
Step 1: Update 0



Step 2: Update 1 and 3

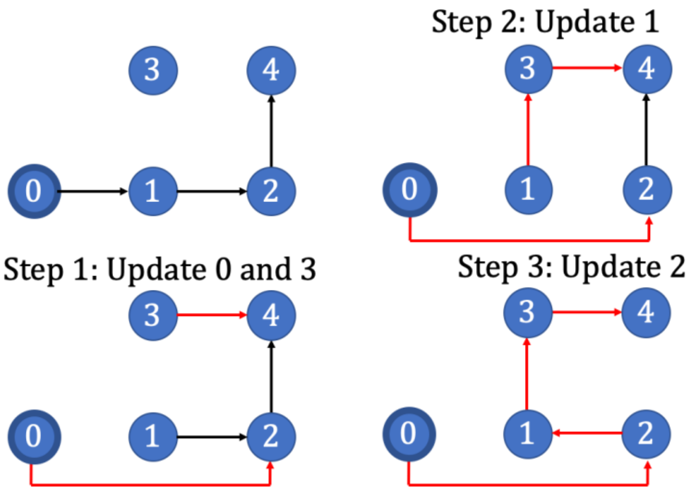


Step 3: Update 2



Programming Project #2: Minimize the number of update rounds

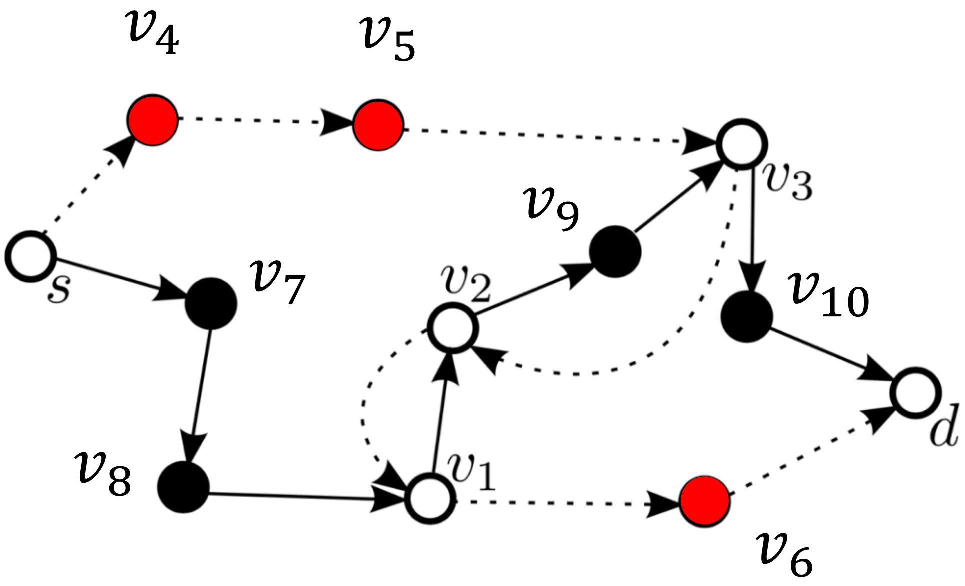
- Input:
  - Numbers of nodes in old and new paths
  - Nodes in old and new paths
- Procedure:
  - Minimize the rounds of update
- Output:
  - Rules of each switch in each round
- The grade is inversely proportional to the number of rounds



Round-Based Update Algorithm

Solid line: old path Dashed line: new path

s	1	2	3	4	5	6	7	8	9	10	d
7	2	9	10	-1	-1	-1	8	1	3	d	-1



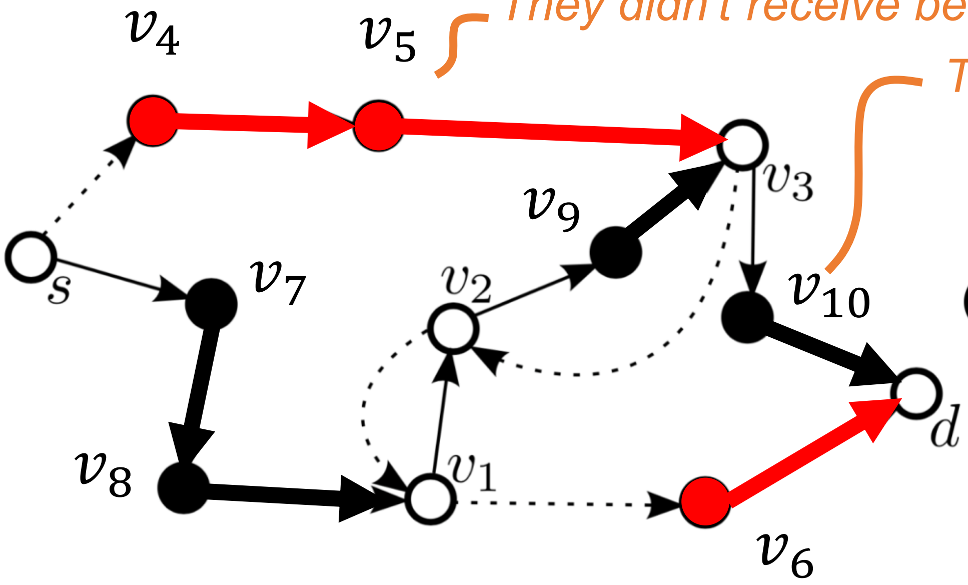
s	1	2	3	4	5	6	7	8	9	10	d
4	6	1	2	5	3	d	-1	-1	-1	-1	-1

Round-Based Update Algorithm

Solid line: old path   Dashed line: new path

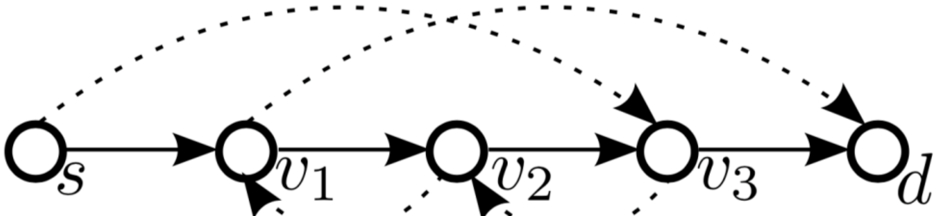
- Add the rules in red nodes in the first round
- Remove the rules in **black nodes** in the **last** round

➔ Reduce the the network to the line representation



They didn't receive before update

They will not receive after update



Round-Based Update Algorithm

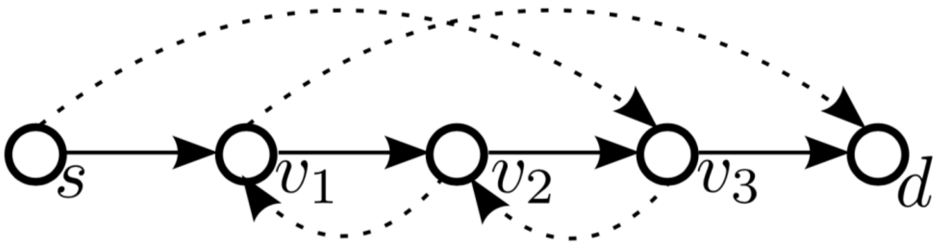
Solid line: old path   Dashed line: new path

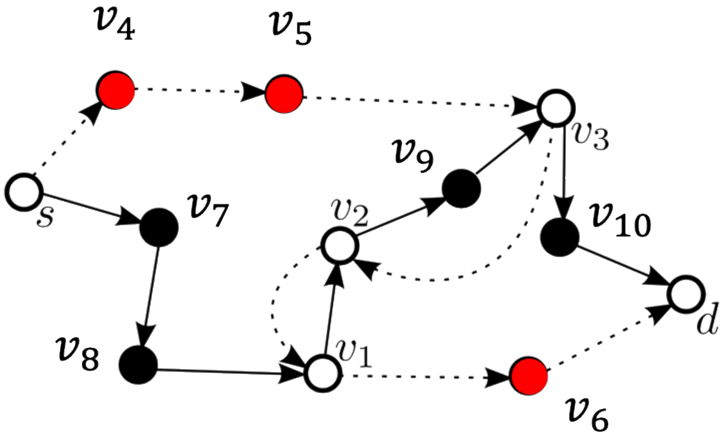
- Add the rules in red nodes in the first round
- Remove the rules in **black nodes** in the **last** round

➔ Reduce the the network to the line representation

s	1	2	3	4	5	6	7	8	9	10	d
7	2	9	10	**5**	**3**	**d**	8	1	3	d	-1

0th round





Round-Based Update Algorithm

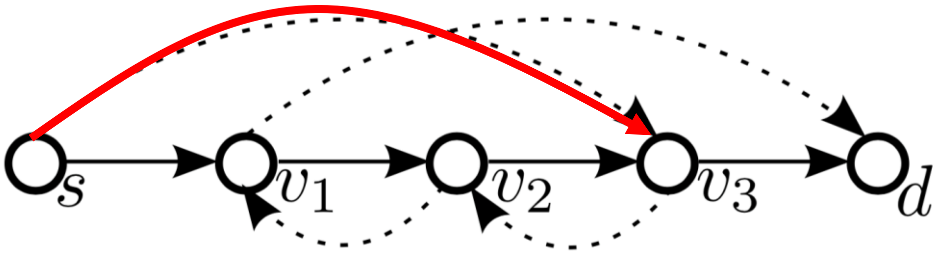
Solid line: old path   Dashed line: new path

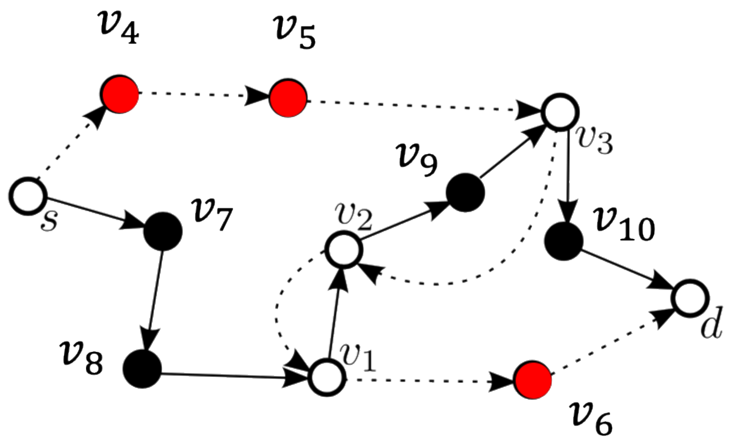
- Add the rules in red nodes in the first round
- Remove the rules in **black nodes** in the **last** round

➔ Reduce the the network to the line representation

s	1	2	3	4	5	6	7	8	9	10	d
4	2	9	10	5	3	d	8	1	3	d	-1

1nd round





Round-Based Update Algorithm

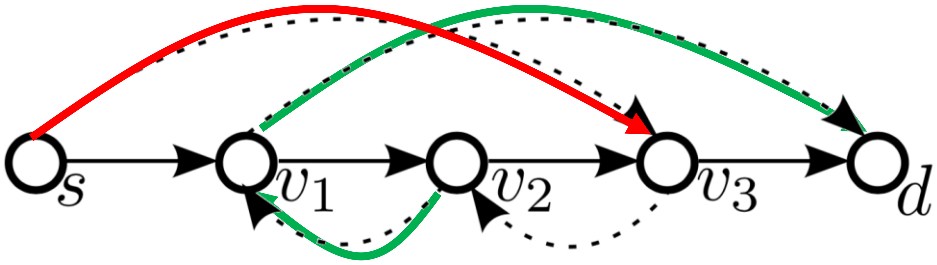
Solid line: old path   Dashed line: new path

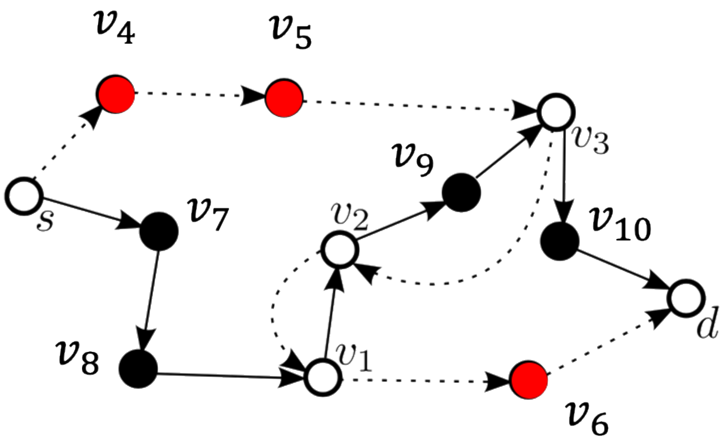
- Add the rules in red nodes in the first round
- Remove the rules in **black nodes** in the **last** round

➔ Reduce the the network to the line representation

s	1	2	3	4	5	6	7	8	9	10	d
4	6	1	10	5	3	d	8	1	3	d	-1

2nd round





Round-Based Update Algorithm

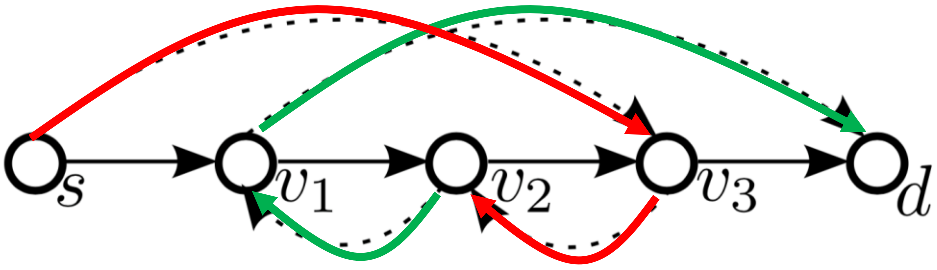
Solid line: old path   Dashed line: new path

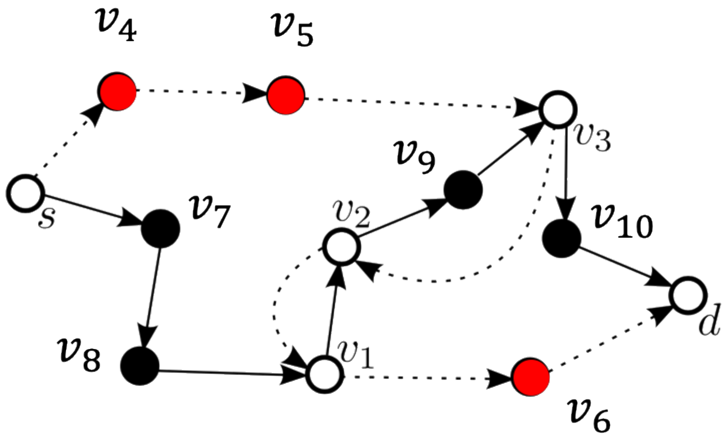
- Add the rules in red nodes in the first round
- Remove the rules in **black nodes** in the **last** round

➔ Reduce the the network to the line representation

s	1	2	3	4	5	6	7	8	9	10	d
4	6	1	2	5	3	d	8	1	3	d	-1

3rd round





Round-Based Update Algorithm

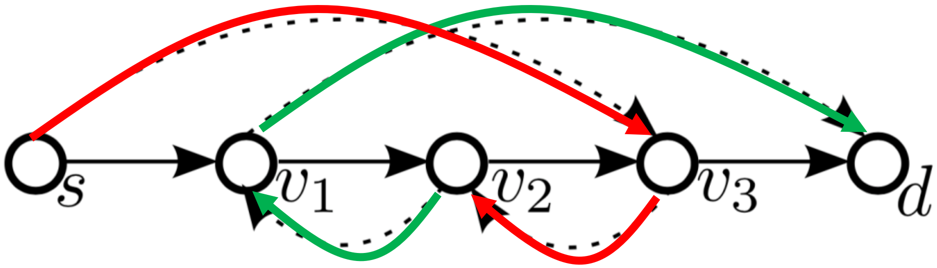
Solid line: old path   Dashed line: new path

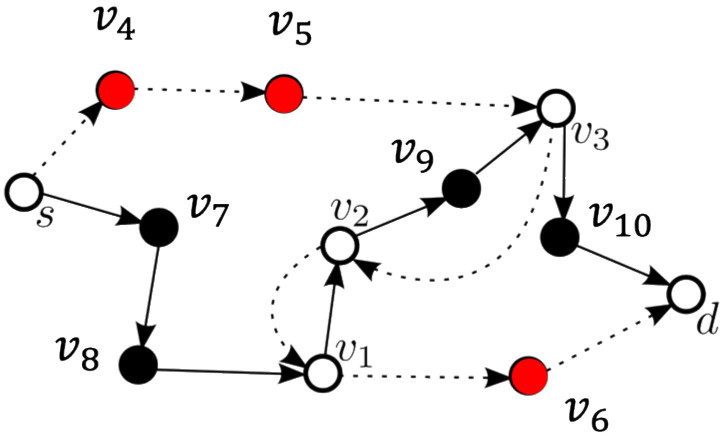
- Add the rules in red nodes in the first round
- Remove the rules in **black nodes** in the **last** round

➔ Reduce the the network to the line representation

s	1	2	3	4	5	6	7	8	9	10	d
4	6	1	2	5	3	d	-1	-1	-1	-1	-1

4th round





Implementation Rules

- You have to use th structure “li ed list” to implement the routing path ext-ho)

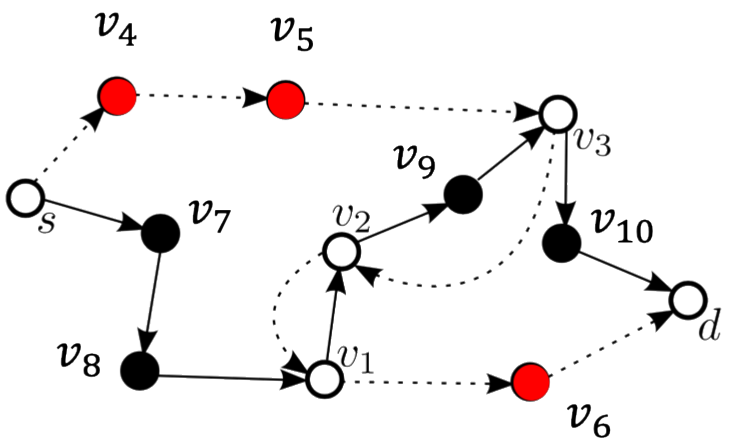
- Next-hop table
- A positive integer i resents node ID
- -1 in the examp ll (0)

```
struct node {
        int id;
        struct node *link;
}
struct node table[nodeNum];
```

Discussion

- Minimizing the number of update rounds is NP-hard
- You may not find the minimum number of update rounds for this problem unless $\mathrm{NP} = \mathrm{P}$

Input Sample: input.txt



Format:

#Nodes

Path1

Path2

s	1	2	3	4	5	6	7	8	9	10	d
7	2	9	10	-1	-1	-1	8	1	3	d	-1

s	1	2	3	4	5	6	7	8	9	10	d
4	6	1	2	5	3	d	-1	-1	-1	-1	-1

e.g.,

```
12
7    2    9    10   -1   -1   -1   8    1    3    11   -1
4    6    1    2    5    3    11   -1   -1   -1   -1   -1
```

Output Sample:

use printf

Format:
#Rounds
Path1
Path2
…

s	1	2	3	4	5	6	7	8	9	10	d
7	2	9	10	-1	-1	-1	8	1	3	d	-1

s	1	2	3	4	5	6	7	8	9	10	d
7	2	9	10	5	3	d	8	1	3	d	-1

s	1	2	3	4	5	6	7	8	9	10	d
4	2	9	10	5	3	d	8	1	3	d	-1

s	1	2	3	4	5	6	7	8	9	10	d
4	6	1	10	5	3	d	8	1	3	d	-1

s	1	2	3	4	5	6	7	8	9	10	d
4	6	1	2	5	3	d	8	1	3	d	-1

s	1	2	3	4	5	6	7	8	9	10	d
4	6	1	2	5	3	d	-1	-1	-1	-1	-1

e.g.,

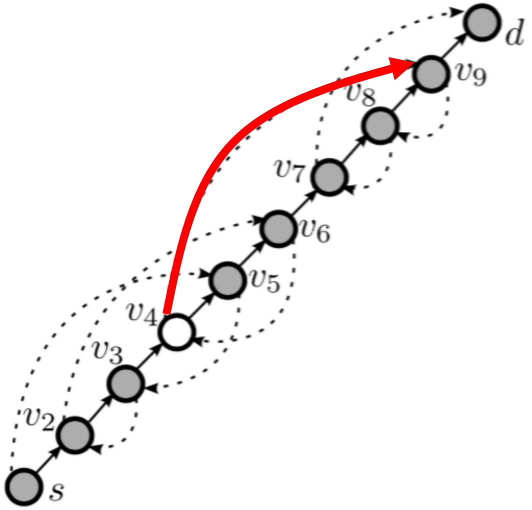
```
6
7    2    9    10   -1   -1   -1   8    1    3    11   -1
7    2    9    10   5    3    11   8    1    3    11   -1
4    2    9    10   5    3    11   8    1    3    11   -1
4    6    1    10   5    3    11   8    1    3    11   -1
4    6    1    2    5    3    11   8    1    3    11   -1
4    6    1    2    5    3    11   -1   -1   -1   -1   -1
```

Note

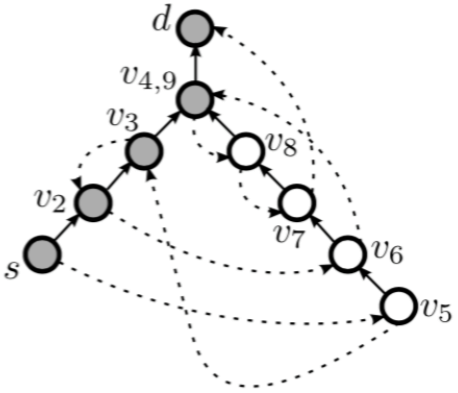
- Superb deadline: 10/31 Tue
- Deadline: 11/7 Tue
- Pass the test of our online judge platform
- Submit your code to E-course2
- Demonstrate your code remotely or in person with TA
- **C Source code (i.e., only .c)**
- Show a good programming style

Note: Round-Based Update Algorithm (1/3)

- Shortcut phase: used in odd rounds
- In each round, we iteratively select the edge that has the farthest reaching distance and does not interfere with the selected edge until there is no such edge
- # selected edges >= 1 → Update the selected edges



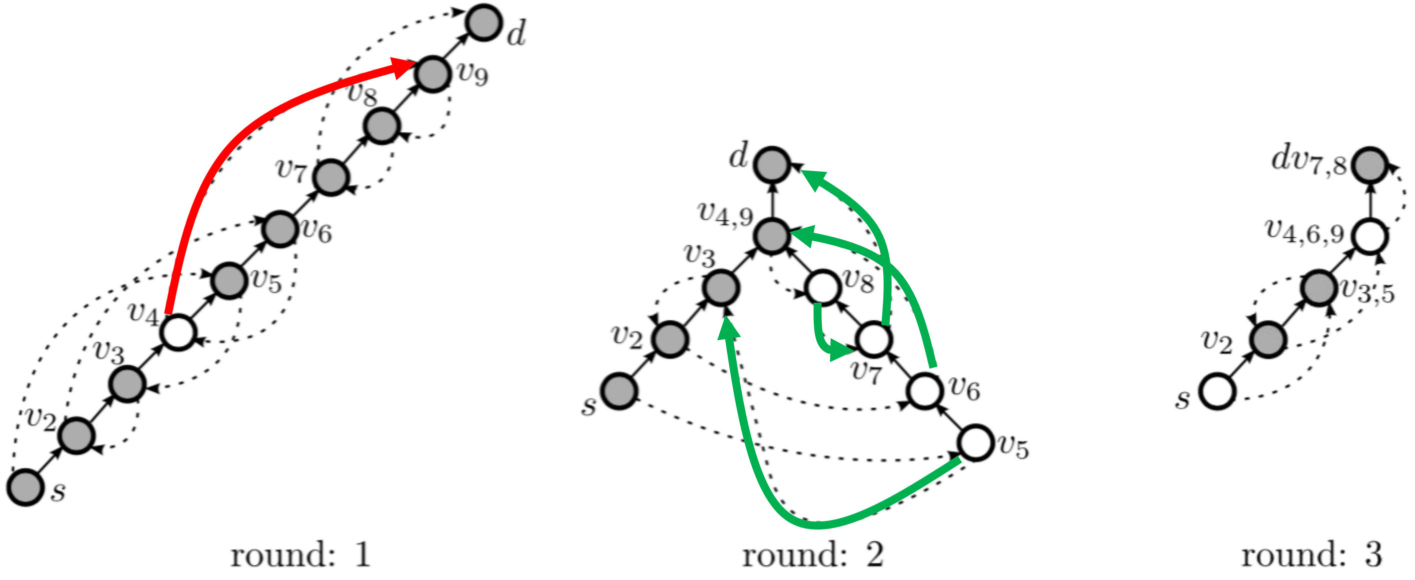
round: 1



round: 2

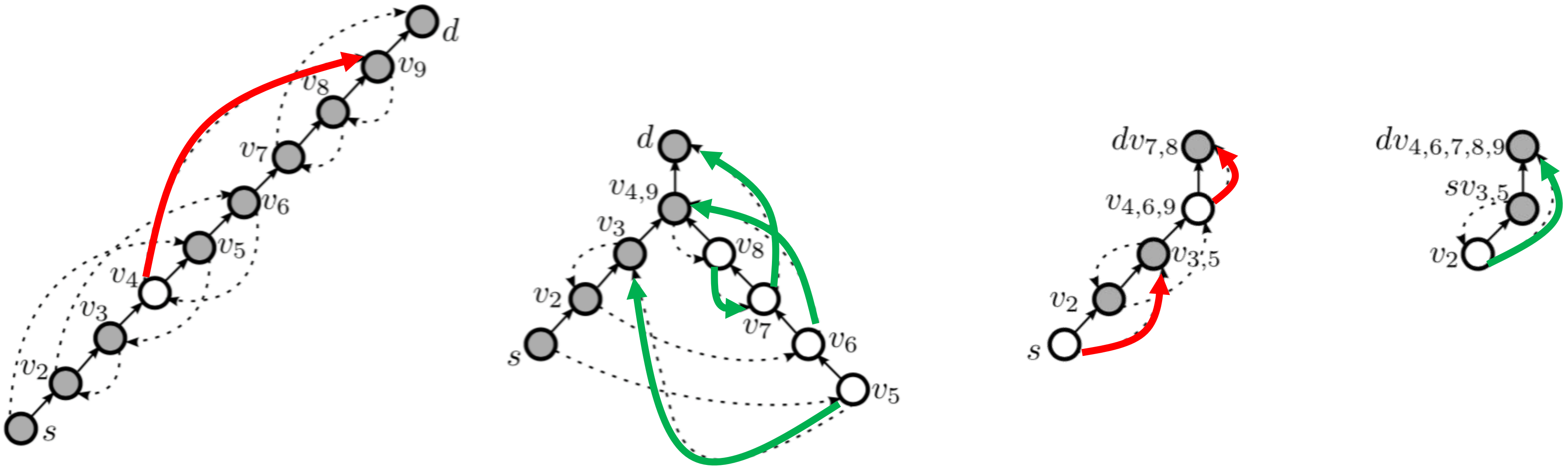
Note: Round-Based Update Algorithm (2/3)

- Prune phase: used in even rounds
- Update all nodes that are not on the current path from the source to the destination
- They can be updated in the same round since they don't receive any packet after the 1st round



Note: Round-Based Update Algorithm (3/3)

- The algorithm
  Repeat the two phases until all nodes are updated
- Shortcut phase: used in odd rounds
- Prune phase: used in even rounds



round: 1 round: 2 round: 3 round: 4