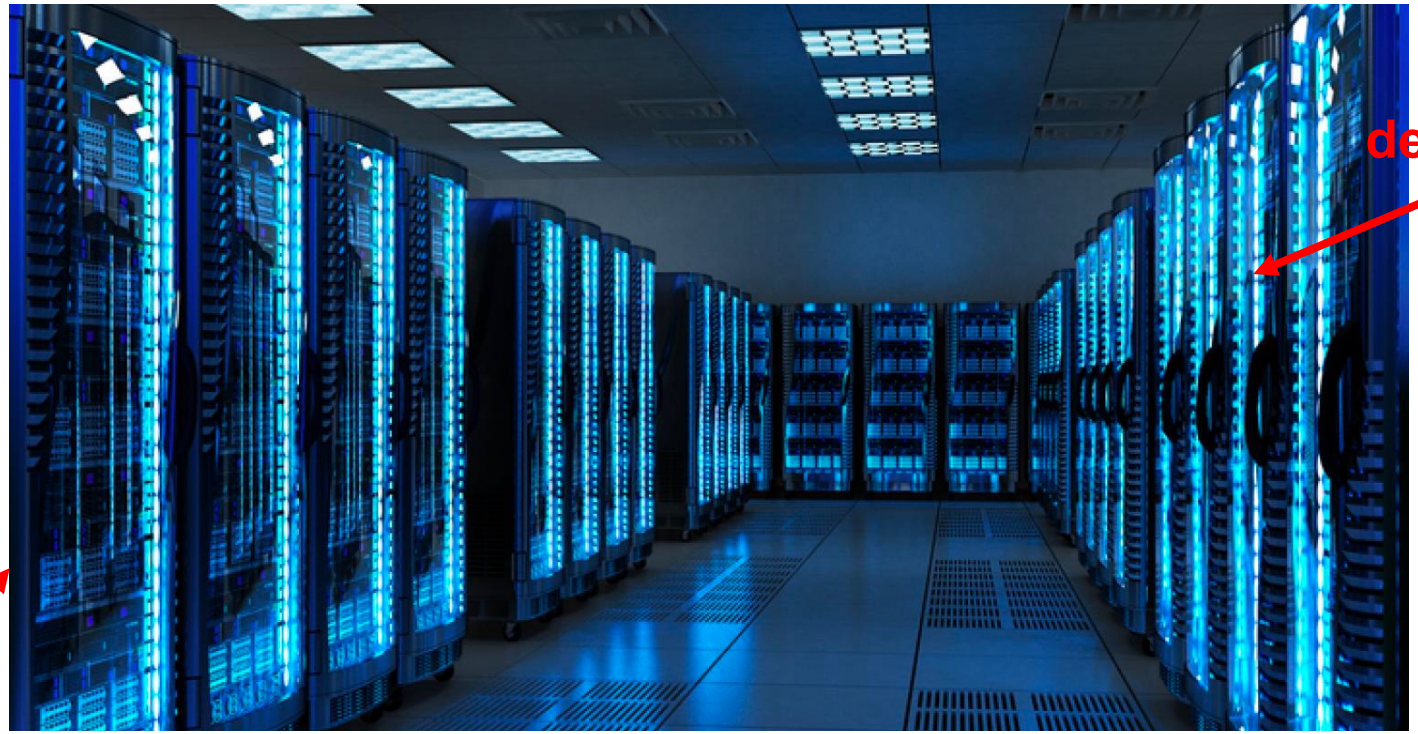
Data Structures
Programming Project #1

Data Center

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

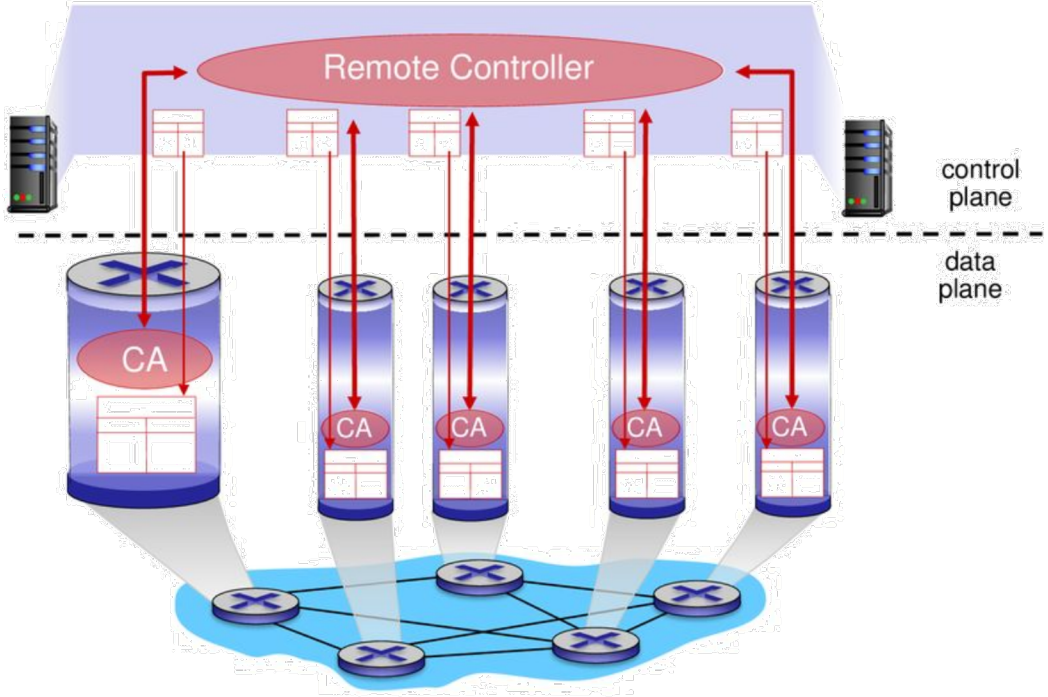


Switches

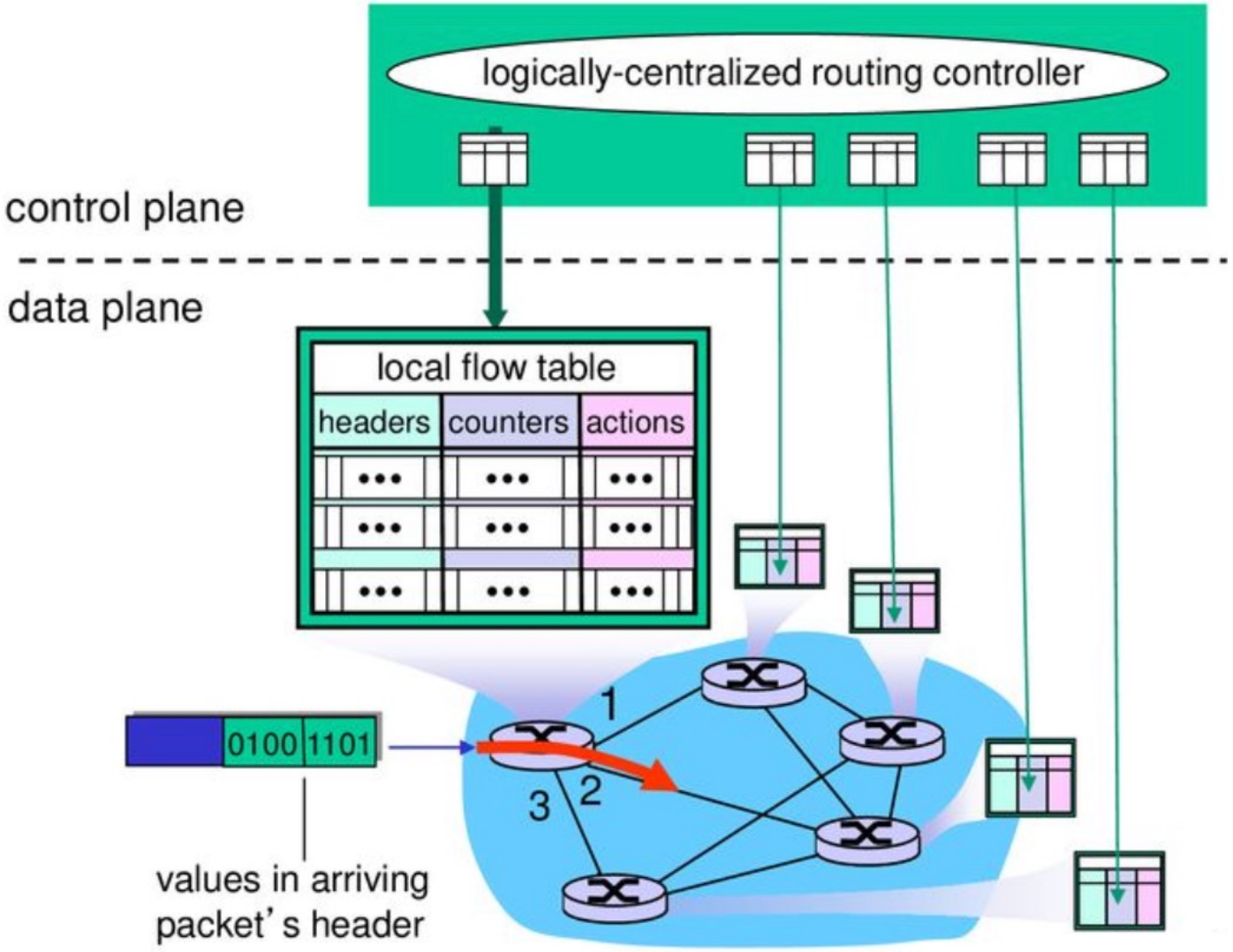
- Each switch has multiple ports
- Receive and forward the packets from a port to another port

SDN-enabled Switches

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

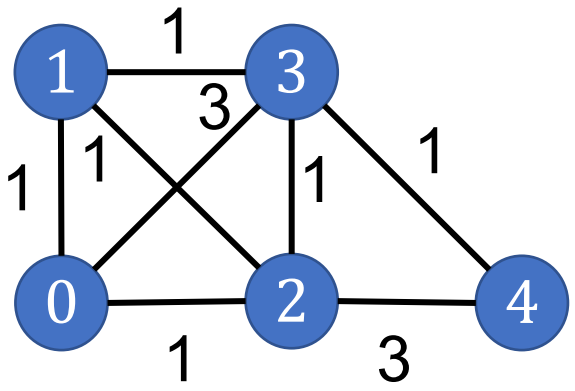


Installing Rules in the SDN-enabled Switches

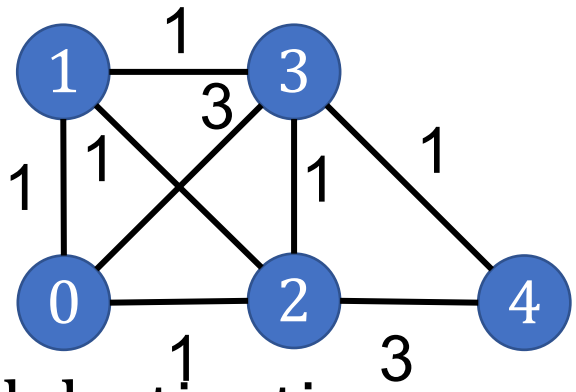


Routing Information

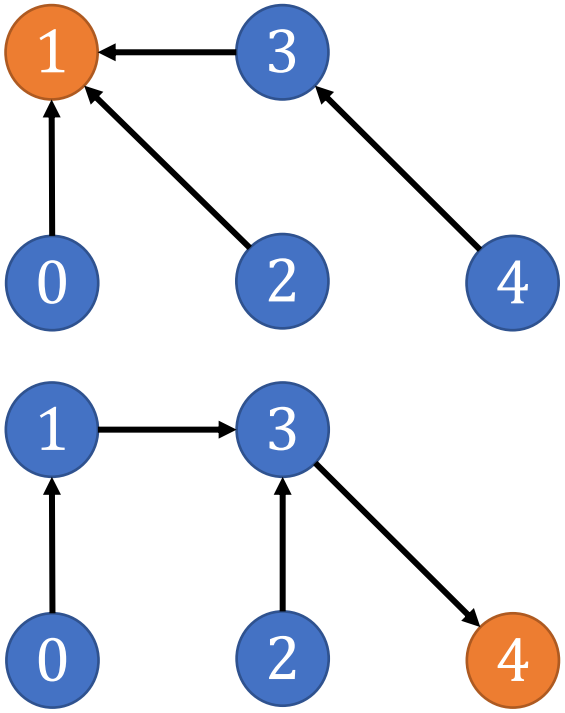
- Given: a graph with link weights and destinations
- Output: shortest paths towards all destinations
  - (Tie breaking) If two next nodes have the same hop, then choose the one with a smaller ID
- Then, store the information in each node's table



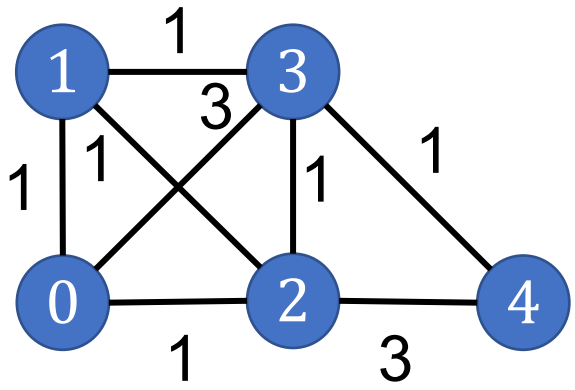
Routing Paths (Trees)



- Given: a graph with link weights and destinations
- Output: shortest paths towards all destinations
- Shortest path trees (SPTs)

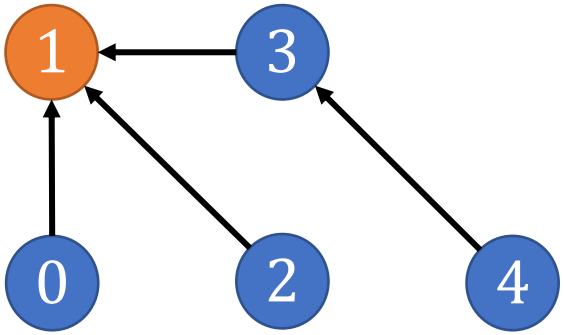


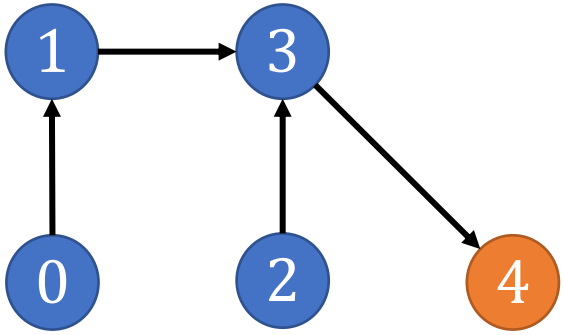
Routing Table



- Key: each destination
- Value: the next node (i.e., the output port)
- Node 0's table

Destination	Next Node
1	1
4	1





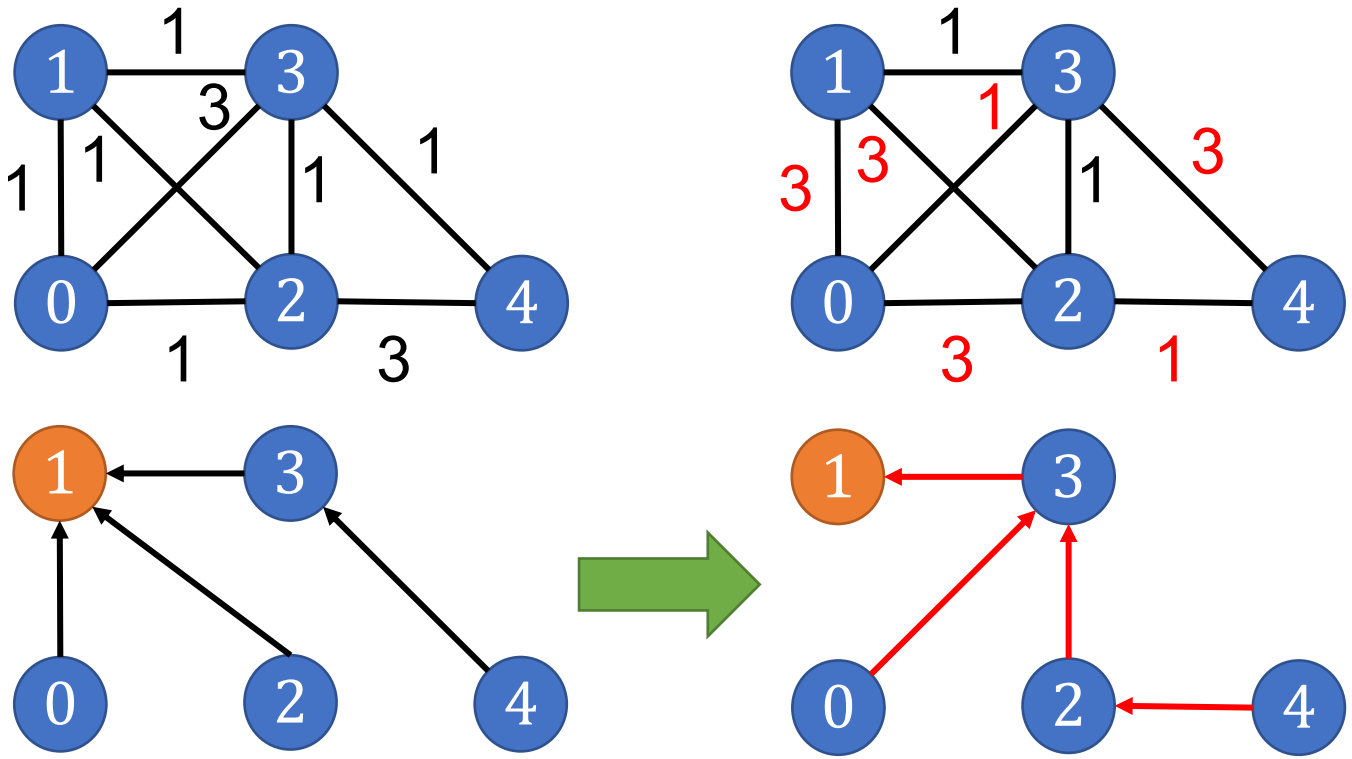
Note

- Every node only knows its neighbors
- Define your own node class and use public

  (i.e., just use class instead of struct)
- Each node has an unsigned int ID
- Use a vector<unsigned int> to store the neighbors' IDs in each node
- Use a map<unsigned int, unsigned int> to store each entry in the table (i.e., <destination ID, next node ID>)

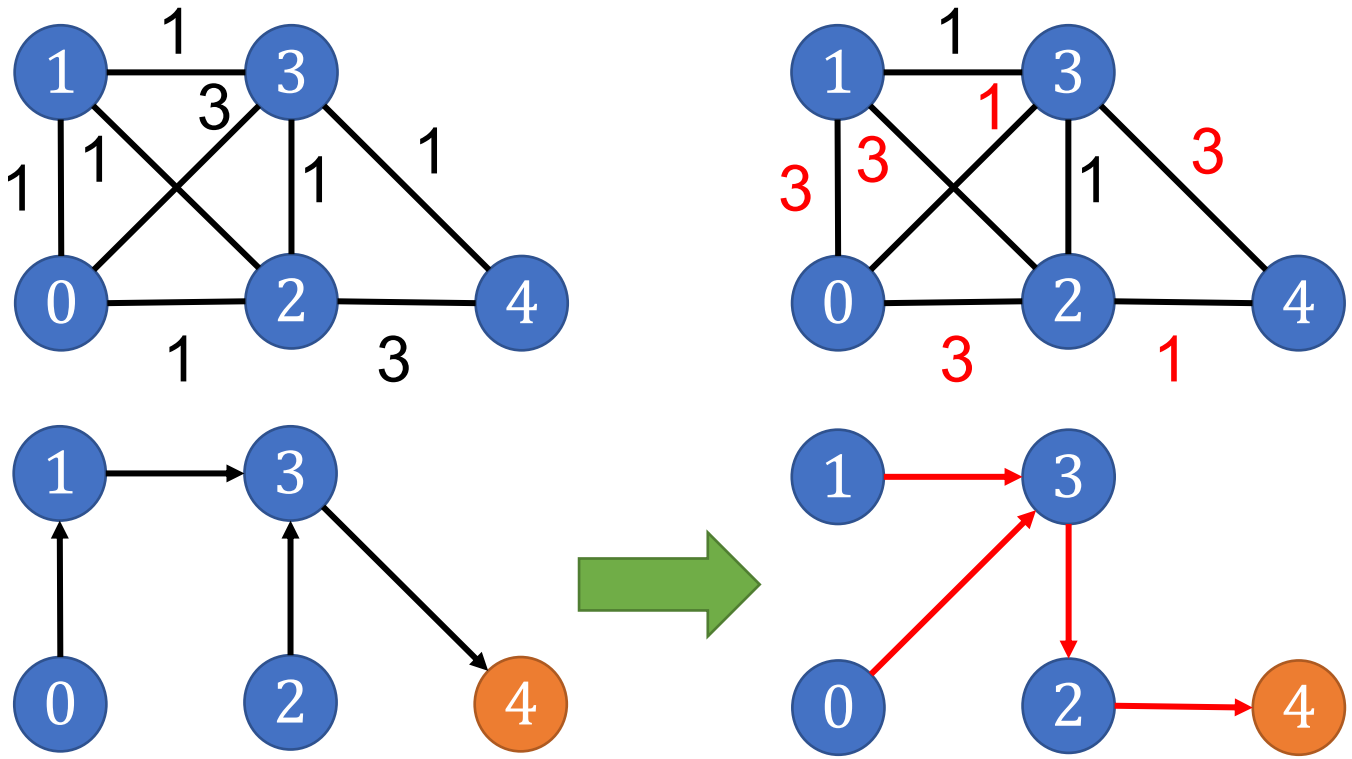
Routing Path Update (aka Network Update)

- Given: the same graph with new link weights
- Value: the new next node (i.e., the output port)



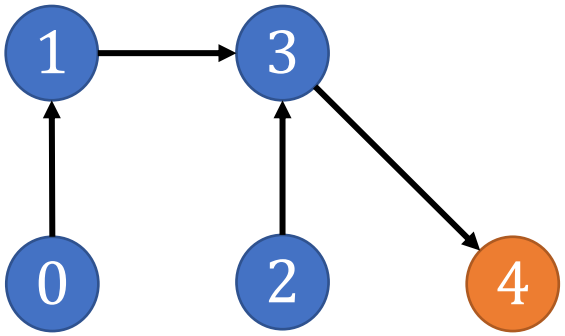
Routing Path Update (aka Network Update)

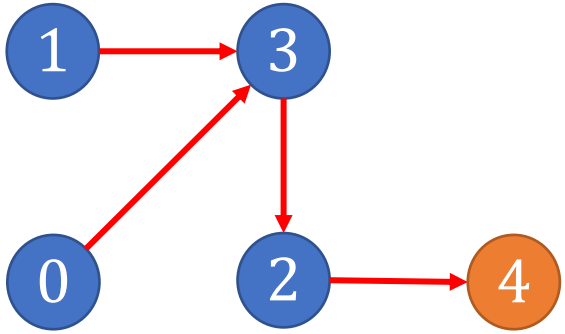
- Given: the same graph with new link weights
- Value: the new next node (i.e., the output port)



Routing Path Update (aka Network Update)

- For each destination, a new tree is generated
- The information in the table should be updated





Node ID	0	1	2	3	4
To 4	1	3	3	4	-1

Node ID	0	1	2	3	4
To 4	3	3	4	2	-1

Programming Project #1: Routing Table

- Input:
  - Numbers of nodes, destinations, and links
  - Destinations
  - Links with **old** weights
  - Links with new weights
- Procedure:
  - Compute **old** shortest paths to destinations
  - Compute new shortest paths to destinations
- Output:
  - Each node's old table
  - Each entry update in each node's table

Input Sample:

use cin

Format:

#Nodes #Dsts #Links

DstID

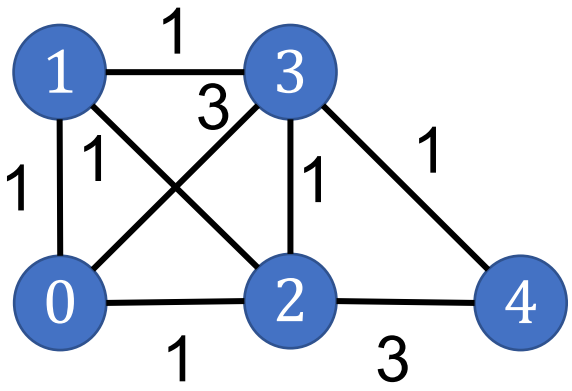
…

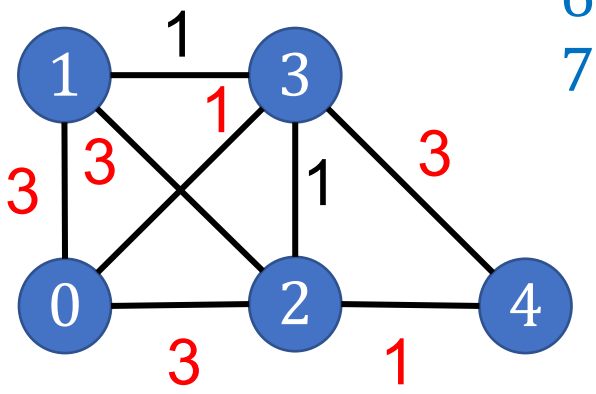
LinkID Node1 Node2 oldW newW

…

e.g.,

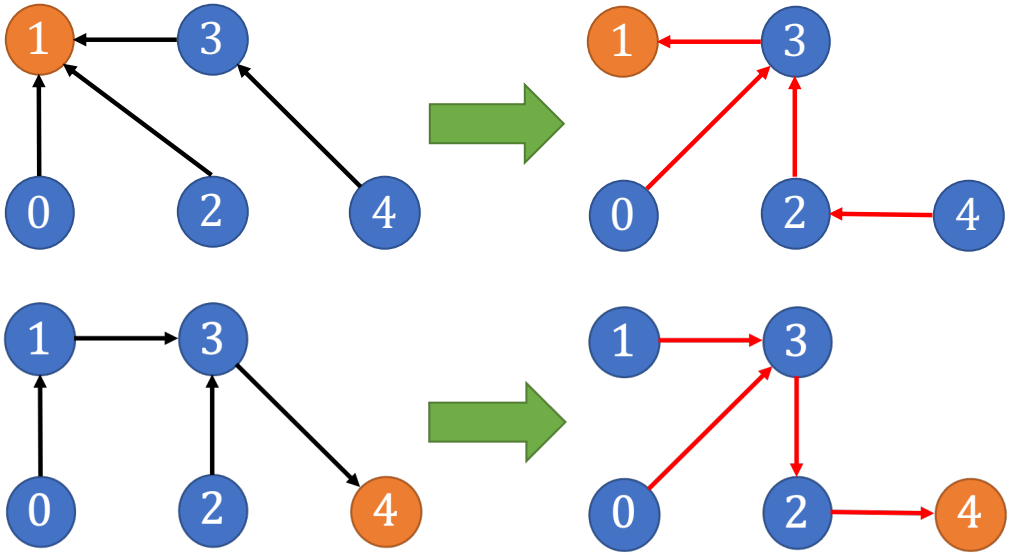
```
5 2 8
1
4
0 0 1 1 3
1 0 2 1 3
2 0 3 3 1
3 1 2 1 3
4 1 3 1 1
5 2 3 1 1
6 2 4 3 1
7 3 4 1 3
```





Output Sample:
use cout

Format:
NodeID
OldEntry
…
OnlyNodeID_withUpdate
NewEntry

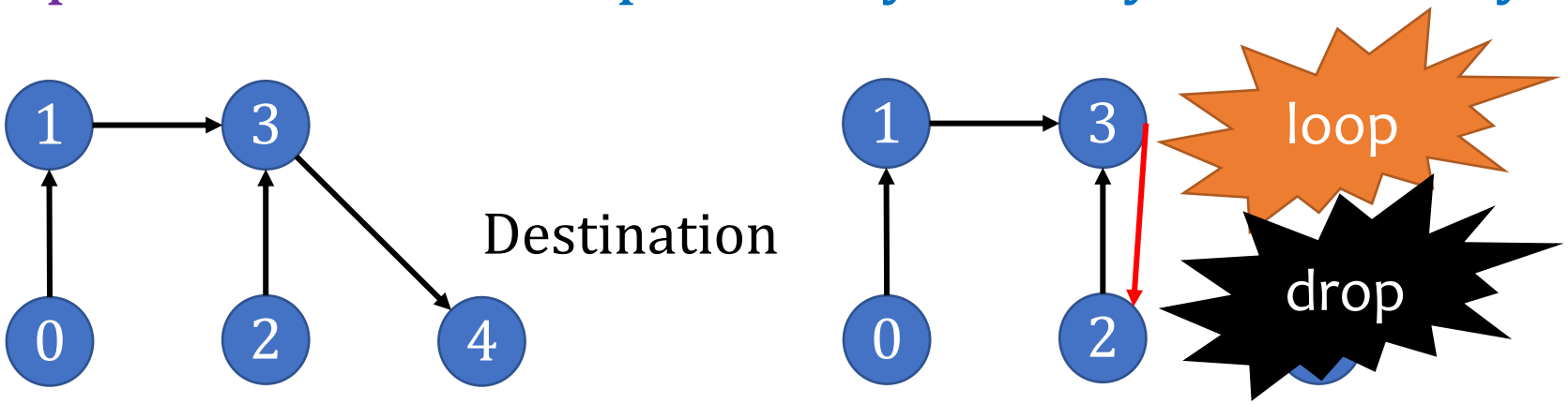


e.g.,

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

Discussion
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



Node ID	0	1	2	3	4
To	1	3	3	4	-1

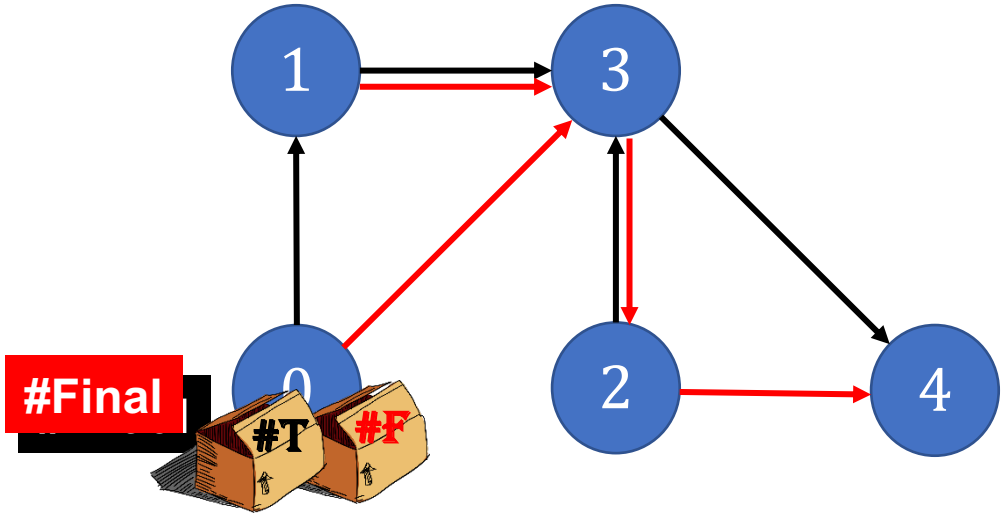
Node ID	0	1	2	3	4
To	1	3	3	2	-1

Discussion
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

Difficulty of Network Update in SDN

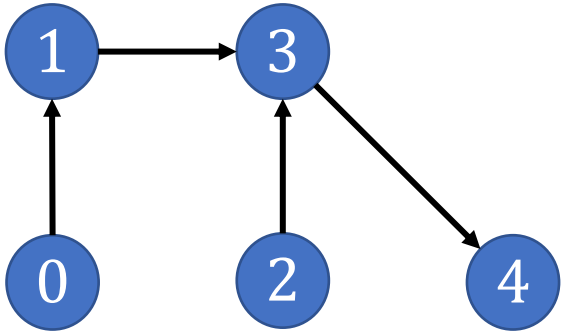
- Two-phase commit (SIGCOMM 2012)
- Drawback: waste the TCAM size during the update



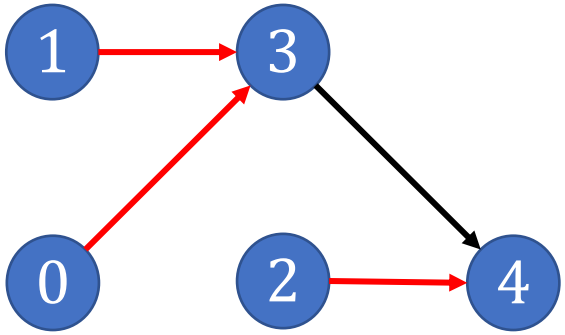
Difficulty of Network Update in SDN → The Next Project

- Round-based update

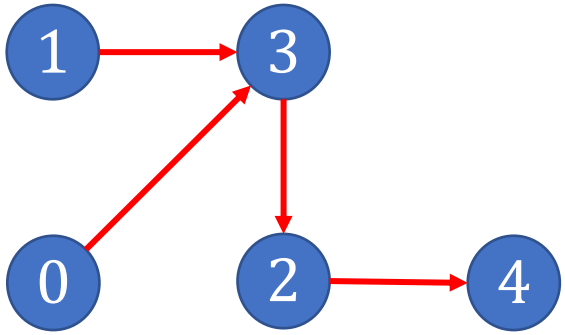
Old



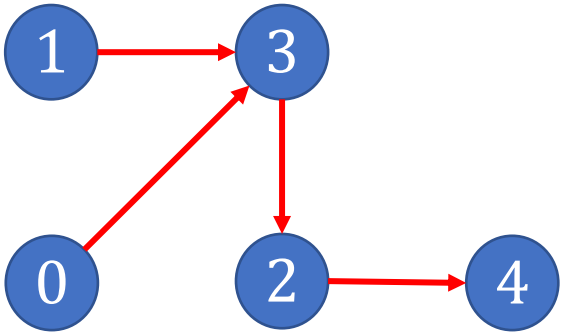
Step 1: Update 0 and 2



Step 2: Update 3



New



Note

- Superb deadline: 3/8 Tue
- Deadline: 3/15 Tue
- Pass the test of our online judge platform
- Submit your code to E-course2
- Demonstrate your code remotely with TA
- **C++ Source code (only C++; compiled with g++)**
  - **Include C++ library only (i.e., no more stdio, no stdlib, …)**
- Show a good programming style