CS438 Assignment 3

03/31/2023

Wang, Jie [jiew5]
Wu, Jiaxin [jiaxin19]

1. TCP RTT Estimation

One difficulty with the original TCP SRTT estimator is the choice of an initial value. In the absence of any special knowledge of network conditions, the typical approach is to pick an arbitrary value, such as 3 seconds, and hope this will converge quickly to an accurate value. If this estimate is too small, TCP will perform unnecessary retransmissions. If it is too large, TCP will wait a long time before retransmitting if the first segment is lost. Also, the convergence might be slow.

1. Choose $\alpha = 0.7$ and RTT-timeout$(0) = 1$ seconds, and assume all measured RTT values $= 0.5$ second with no packet loss. What is RTT-timeout(20)? Recall,

$$\text{RTT-timeout}(k+1) = \alpha \times \text{RTT-timeout}(k) + (1-\alpha) \times \text{RTT}(k+1)$$

   Describe your solution approach AND provide the numerical result (approximate to 4^{th} decimal place).

2. Using the same values as in above part, what happens if we use $\alpha = 0.5$ or $\alpha = 0.95$? Provide a numerical result for RTT-timeout(20) in both cases, then describe the effect of a larger or smaller α on the RTT estimation procedure.

(1) $\alpha = 0.7$

$$\begin{aligned} RTT\text{-}timeout(1) &= \alpha \times RTT\text{-}timeout(0) + (1-\alpha) \times RTT(1) \\ RTT\text{-}timeout(2) &= \alpha \times RTT\text{-}timeout(1) + (1-\alpha) \times RTT(2) \\ &= \alpha \times (\alpha \times RTT\text{-}timeout(0) + (1-\alpha) \times RTT(1)) + (1-\alpha) \times RTT(2) \\ &= \alpha^2 \times RTT(0) + \alpha(1-\alpha) \times RTT(1) + (1-\alpha) \times RTT(2) \\ Since\ RTT(1) &= RTT(2) = 0.5s, RTT\text{-}timeout(2) = \alpha^2 \times RTT(0) + (1-\alpha^2) \times 0.5, \end{aligned}$$

Similarly substitute **RTT-timeout(k)** with previous result when computing each **RTT-timeout(k+1),** we have ：

$$RTT\text{-}timeout(k) = \alpha^k \times RTT(0) + (1-\alpha^k) \times 0.5$$

Therefore,

$$RTT\text{-}timeout(20) = \alpha^{20} \times RTT(0) + (1-\alpha^{20}) \times 0.5 = 0.7^{20} \times 1 + (1-0.7^{20}) \times 0.5 \approx 0.5004s.$$

(2) $\alpha = 0.5 \& 0.95$, what differs?

Here we use same formula based on (1)

If α = 0.5, $RTT\text{-}timeout(20) = 0.5^{20} \times 1 + (1-0.5^{20}) \times 0.5 \approx 0.5000s.$
If α = 0.95, $RTT\text{-}timeout(20) = 0.95^{20} \times 1 + (1-0.95^{20}) \times 0.5 \approx 0.6792s.$
If α is larger, each measured RTT has a smaller effect on the estimated RTT value, so the estimated RTT value converges more slowly to an accurate RTT value.

If α is smaller, each measured RTT has a larger effect on the estimated RTT value, so the estimated RTT value converges faster to the accurate RTT value.

> Note: the conclusion here is based on the fact that measured RTT always stay the same

Python code implementation

```python
import numpy as np
def RTT_compute(k,alpha,RTT_mesaure):
    print("now start alpha == ",alpha)
    RTT_0 = 1
    RTT = RTT_0
    for _ in range(k):
        RTT = alpha*RTT + (1-alpha)*RTT_mesaure
```

```
        if np.abs( RTT - RTT_mesaure) < 0.00001:
            print("Premature, loop times is ",_)
            break
    return(RTT)

a1 = RTT_compute(20,0.7,0.5)
print("a1 is ",a1)
a2 = RTT_compute(20,0.5,0.5)
print("a2 is ",a2)
a3 = RTT_compute(20,0.95,0.5)
print("a3 is ",a3)
```

Output:

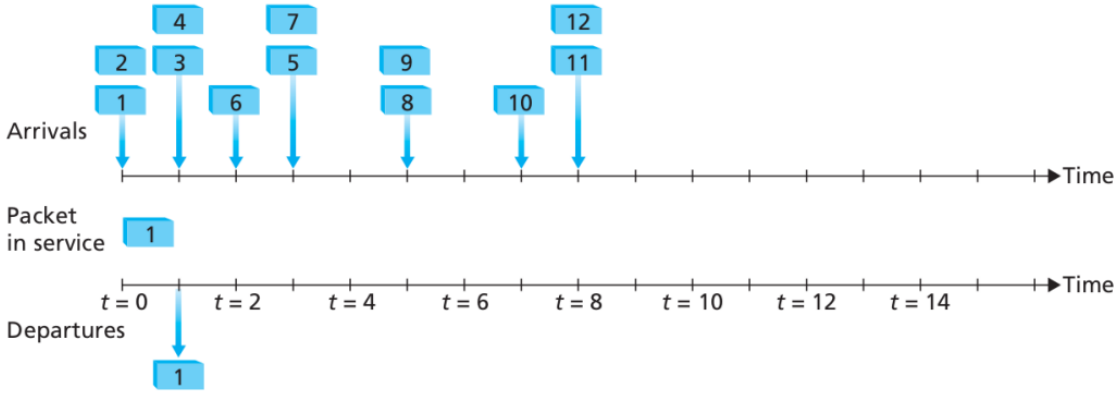
```
now start alpha ==  0.7
a1 is  0.5003989613314881
now start alpha ==  0.5
Premature, loop times is 15
a2 is  0.5000076293945312
now start alpha ==  0.95
a3 is  0.679242961204271
```

This approach further prove the correctness of our closed-form formula

2. Fair Queuing

Consider Fig. 1. Now consider the following scenarios:

Figure 1:



(1) Table 1: Packet Scheduling

Time of Departure	FIFO		Highest Priority		Round Robin		WFQ	
(t in sec)	Packet	Delay	Packet	Delay	Packet	Delay	Packet	Delay
1	1	1	1	1	1	1	1	1
2	2	2	3	1	2	2	4	1
3	3	2	2	3	4	2	2	3
4	4	3	5	1	3	3	3	3
5	6	3	7	2	6	3	6	3
6	5	3	9	1	5	3	9	1
7	7	4	4	6	7	4	7	4
8	8	3	6	6	8	3	10	1
9	9	4	11	1	11	1	5	6
10	10	3	8	5	9	5	12	2
11	11	3	10	4	12	3	8	6
12	12	4	12	4	10	5	11	4

(2) Performance Analysis

- For FIFO: Average delay is **2.9167**s.
- For highest priority:
  - Overall average delay is **2.9167**s.
  - average delay is **1.167s** for high priority class
  - **4.667s** for low priority class.
- For Round Robin
  - Overall average delay is **2.9167**s.
  - average delay is **3.286s** for class 1
  - average delay is **2.4s** for class 2.
- For WFQ:
  - Overall average delay is **2.9167**s.
  - average delay is **2.5s** for class 0,
  - average delay is **2s** for class 1
  - average delay is **4.25s** for class 2.

(3) Observation

- The **overall average delay** for the 4 scheduling policies are similar.
- In Highest priority and WFQ, the average delay of a **class with a higher priority** or a **larger weight** is smaller than other classes.

3. Forwarding and CIDR

1. Consider a router that interconnects three subnets: Subnet A, Subnet B and Subnet C. Suppose all of the interfaces in each of these three subnets are required to have the prefix 200.20.15.0/24. Also suppose that Subnet A is required to support up to 80 interfaces, and Subnets B and C are each required to support up to 25 interfaces. Provide three network addresses (of the form a.b.c.d/x) for each of the above subnet that satisfy these constraints.

(1) Classful IP Addressing

The prefix 200.20.15.0/24 means that the first 24 bits are fixed, and we have 8 bits to work with for subnetting.

Subnet A

Because A needs 80 interfaces, which is within $(2^6, 2^7)$, so the router should assign A with 7 bits.

Therefore, **Subnet A:** ***200.20.15.0/25***

(200.20.15.0 to 200.20.15.127)

Subnet B

Because B needs 25 interfaces, which is within $(2^4, 2^5)$, so the router should assign B with at least 5 host bits. Note we need to increase from A, the subnet mask should be ***/27*** .

Therefore, **Subnet B:** ***200.20.15.128/27***

(200.20.15.128 to 200.20.15.159)

Subnet C

Because C needs 25 interfaces, which is within $(2^4, 2^5)$, so the router should assign c with at least 5 host bits. Note we need to increase from b, the subnet mask should still be ***/27*** .

Therefore, **Subnet B:** ***200.20.15.160/27***

(200.20.15.160 to 200.20.15.191)

(2) IP-match pair

Suppose a router has built up the routing table shown below in Table. 2. CIDR addresses are used, with "/22" indicating a mask of 22 1's followed by 10 0's.

Net/Masklength	NextHop
128.174.240.0/20	Interface 1
128.174.240.128/25	R1
128.174.240.17	R2
128.174.252.0/22	Interface 3
128.174.240.16/29	R3
128.174.248.0/22	Interface 2
default	Interface 4

Table 2:

Based on Table-2, **How many individual IP addresses match each Net/Masklength pair?**

(Compute this for all entries in the table except the last one, i.e. for the default Masklength entry).

Ans:

Net/Masklength	NextHop	# of Individual IP addresses	IP range field
128.174.240.0/20	Interface 1	4096	128.174.240.0 to 128.174.255.255
128.174.240.128/25	R1	128	128.174.240.128 to 128.174.240.255
128.174.240.17	R2	1	A single IP address within the range of the first subnet
128.174.252.0/22	Interface 3	1024	128.174.252.0 to 128.174.255.255 (overlaps with the first subnet 128.174.240.0/20)
128.174.240.16/29	R3	8	128.174.240.16 to 128.174.240.23 (falls within the second subnet 128.174.240.128/25)
128.174.248.0/22	Interface 2	1024	128.174.248.0 to 128.174.251.255 (overlaps with the first subnet 128.174.240.0/20)
default	Interface 4	NULL	NULL (All the other ip left)

(3) NextHop jumping

The router can deliver packets directly over interfaces 1, 2, 3, 4, or it can forward to routers R1, R2, R3.

- **Specify the next hop for each of the following destinations**. Use the longest prefix match, i.e., if a destination matches more than one line of the table, the longest match is used.

IP address	destination hop
(a) 128.174.240.17	R2
(b) 128.174.245.17	Interface 1
(c) 128.174.250.17	Interface 2
(d) 128.174.254.17	Interface 4 (no pair)
(e) 128.174.225.17	Interface 1
(f) 128.174.240.18	R3

4. DHCP and NAT

Alice and Bob are neighbors and they each buy a new home wireless router. After connecting each of their laptops to their own router they each enter the command `hostname -i`, which prints out their IP address.

1. Briefly describe the process how they get the IP address.
2. After doing this, is it possible that they get the same IP address, if the routers use private subnet addresses?
3. Give one reason that wide-spread deployment of IPv6 would let them get rid of the NAT service provided by thier routers.
4. Give one reason that they might want to continue using the router's NAT service even if they could use IPv6.

(1) How do they fetch ip

The whole procedure is based on Dynamic Host Configuration Protocol (DHCP):

- Alice and Bob's laptops (hosts) request IP address separately from their own router.
  - Hosts broadcasts "*DHCP discover*" msg
- Routers use their integrated DHCP server to respond the latptops,
  - DHCP server respond with "*DHCP offer*" msg
- Laptops know there is a router, then request for an IP
  - Host request IP with "*DHCP request*" msg
- And these routers obtain the IP address from network server when joining the network (If it is the first time connecting network).
- DHCP server send laptops with available IP address:
  - DHCP server respond with "*DHCP ack*" msg

After router connecting to the internet, it creates a close subnet and connect with the new laptop via private subnet addresses.

(2) Same ip?

Yes it is possible they fetch same private subnet IP address. Especially if the router use NAT(network address translation) on them, Alice and Bob may have same local ip address 192.168.56.101.

However, there won't be any collision because two laptops are in different network, which cannot be routed externally.

If the hosts need to route outside internet, it would use NAT service to translate their IP and request for information through the router.

(3) IPv6 development?

Since IPv6 has 128-bit IP addressability, Alice and Bob may not need NAT service. Because the IPv6 is large enough to allocate globally distinct address with any devices connecting to the internet.

This would simplify end-to-end communication and potentially improve the performance of certain applications, such as P2P file sharing.

(4) Why still NAT?

- To begin with, the IPv4 is still widely used around the world, most of the web service are still based on it. Connecting IPv6 is not enough for them to surf the internet.
- Secondly, NAT provides a basic level of security. By translating private IP addresses to a single public IP address, NAT hides the internal network structure and makes it more difficult for external devices to directly access devices within the home network
- Additionally, NAT can help conserve global IP addresses by allowing multiple devices to share a single public IP address. Maybe in the future, the huge addressability of IPv6 can still be used up, and NAT will help to solve this problem a lot.

5. Distance Vector Routing

(1) Shortest Path Tree

> **Derive and draw the shortest-path tree from nodes E using the Dijkstra's algorithm**.
>
> Show how the cost of path and predecessor node along path converge step-by-step in a table as demonstrated in the lecture slides.

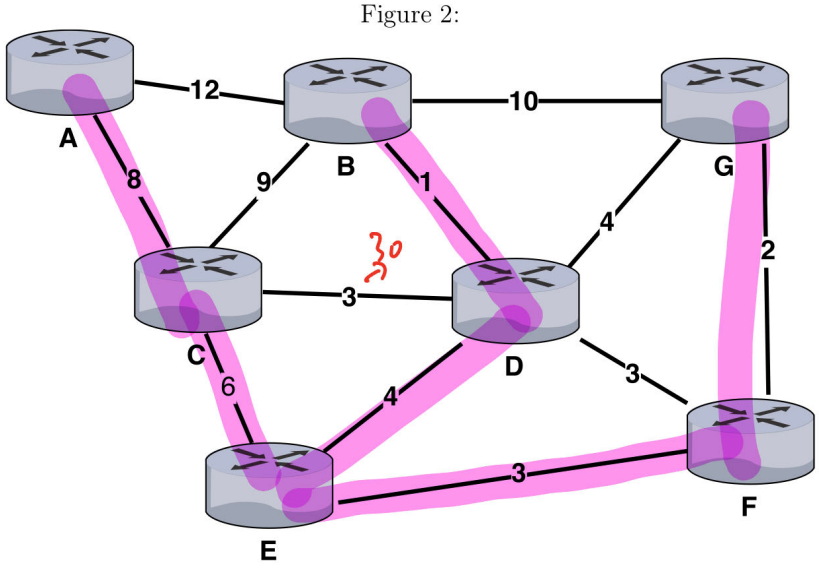
- Dijkstra's Table (Here the node sequence is based on BFS, it can also be turned into dictionary order).

Step	Node Visited	Dist(C)	Dist(D)	Dist(F)	Dist(A)	Dist(B)	Dist(G)
0	E	6,E	4,E	**3,E**	∞	∞	∞
1	EF	6,E	**4,E**		∞	∞	5,F
2	EFDG	6,E			∞	5, D	**5,F**
3	EFDG	6,E			∞	**5,D**	
4	EFDGB	**6,E**			17, B		
5	EFDGBC				**14, C**		
6	EFDGBCA						

Note: **Bold** means the time one node reaches its shortest path from E

The shortest path tree drawn as the Figure 2 below:

Figure 2:



(2) Routing table using DV algo

> **Compute routing tables for nodes B, D and E using the distance vector algorithm.**
>
> The table rows should include a Destination, Distance, and Next Hop.

(3)Edge increase case

> Now consider a situation in which link between the D-C becomes congested and its costs become 30. List a sequence of updates to routing tables at nodes B and D, in order, until the routing tables converge

6. Routing Algorithms

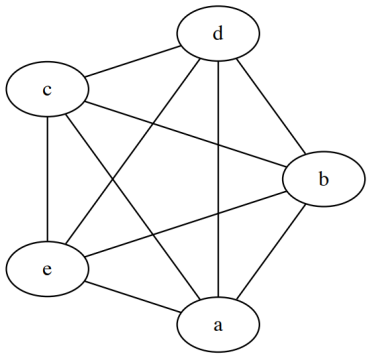
(1) Known one path

1. Consider the network of routers as shown in Fig. 3. Suppose you are told that traffic from a to b is routed by the path $a \to c \to e \to b$

   (a) List all the possible routes from d to e that could coexist with the above route from a to b on this network, if routing is performed using the link-state algorithm. Hint: Consider what the routing decision $a \to c \to e \to b$ says about the relative costs of individual edges. Can you use them to rule some items out.

   (b) List all the possible routes from d to a that could coexist with the above route from a to b on this network, if routing is performed using software- defined networking.

Figure 3:



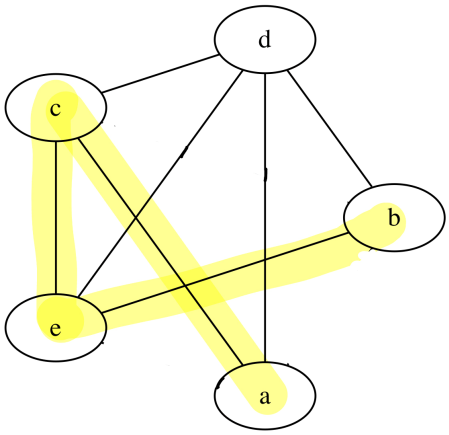
(a) Possible moves from d ->e

> List all the possible routes from d to e that could coexist with the above route from a to b on this network, if routing is performed using the link-state algorithm. Hint: Consider what the routing decision a → c → e → b says about the relative costs of individual edges. Can you use them to rule some items out.

Based on the knowledge of shortest path (a-c-e-b), we can build a new graph

$G' = G - (a \leftrightarrow b) - (c \leftrightarrow b) - (a \leftrightarrow e)$,

because $a \leftrightarrow b$, $(a \leftrightarrow e)$ and $c \leftrightarrow b$ is too long. We have G' below:



Then, the possible path from d to e is :

- d->e

- d->c->e
- d->b->e
- d->a->c->e

(b) SDN-based routing from d to a

> List all the possible routes from d to a that could coexist with the above route from a to b on this network, if routing is performed using software- defined networking.

In a software-defined networking (SDN) scenario, the routing decisions are made by a centralized controller, which has complete knowledge of the network topology and can dynamically adapt to network changes.

As a result, the SDN controller can make use of any available path between two nodes, even if it's not the shortest path. Therefore, we don't need to build a new graph G' in this case and can use any available path without a loop from d to a.

1.d->a
2.d->b->a
3.d->b->c->a
4.d->b->e->a
5.d->b->c->e->a
6.d->b->e->c->a
7.d->c->a
8.d->c->b->a
9.d->c->e->a
10.d->c->b->e->a
11.d->c->e->b->a
12.d->e->a
13.d->e->b->a
14.d->e->c->a
15.d->e->b->c->a
16.d->e->c->b->a

(2) Routing Protocol Evaluation

> Sort the following protocols by the amount of state each node maintains, and give clear explanation.
> • Link State
> • Distance Vector
> • Path Vector

Answer: **Distance Vector < Link State < Path vector**
Explanation:
• Distance Vector algorithm:
Each node estimates its cost to the destination, and maintains cost to each of its neighbors and the distance vector of each of its neighbors.
• Link State algorithm:
Each node maintains the least cost paths from itself to all other nodes, as well as the next hop in the least cost path, so each node maintains more state information than Distance Vector algorithm.
• Path Vector algorithm:
Each node maintains a path vector for each destination which contains the entire path

information to the destination node, so each node maintains more state information than Link State algorithm.

ECE 438 T5

(2): First Iteration.

[initial step: only know own neighbor]

1

A	A	B	C	D	E	F	G
A	0	12	8	∞	∞	∞	∞
B							
C							

B	A	B	C	D	E	F	G
A							
B	12	0	9	1	∞	∞	10
C							
D							
G							

C	A	B	C	D	E	F	G
A							
B							
C	8	9	0	3	6	∞	∞
D							
E							

D	A	B	C	D	E	F	G
B							
C							
D	∞	1	3	0	4	3	4
E							
F							
G							

E	A	B	C	D	E	F	G
C							
D							
E	∞	∞	6	4	0	3	∞
F							

F	A	B	C	D	E	F	G
D							
E							
F	∞	∞	∞	3	3	0	2
G							

G	A	B	C	D	E	F	G
B							
D							
F							
G	∞	10	∞	4	∞	2	0

Note:

1° If not sepecified, the nexthop is direct to itself

2° If the distance vector changes, we circle it

3° We only list connectable neighbor point in distance vector

Second Iteration:

A	A	B	C	D	E	F	G
				C	C		B
A	0	12	8	11	14	∞	22
B	12	0	9	1	∞	∞	10
C	8	9	0	3	6	∞	∞

B	A	B	C	D	E	F	G
			D		D	D	D
A	0	12	8	∞	∞	∞	∞
B	12	0	9	1	5	4	5
C	8	9	0	3	6	∞	∞
D	∞	1	3	0	4	3	4
G	∞	10	∞	4	∞	2	0

C	A	B	C	D	E	F	G
		D				D	D
A	0	12	8	∞	∞	∞	∞
B	12	0	9	1	∞	∞	10
C	8	4	0	3	6	6	7
D	∞	1	3	0	4	3	4
E	∞	∞	6	4	0	3	∞

D	A	B	C	D	E	F	G
			C				
B	12	0	9	1	∞	∞	10
C	8	9	0	3	6	∞	∞
D	11	1	3	0	4	3	4
E	∞	∞	6	4	0	3	∞
F	∞	∞	∞	3	3	0	2
G	∞	10	∞	4	∞	2	0

E	A	B	C	D	E	F	G
	C	D					F
C	8	9	0	3	6	∞	∞
D	∞	1	3	0	4	3	4
E	14	5	6	4	0	3	5
F	∞	∞	∞	3	3	0	2

F	A	B	C	D	E	F	G
		D	D				
D	∞	1	3	0	4	3	4
E	∞	∞	6	4	0	3	∞
F	∞	46	3	3	0	2	
G	∞	10	∞	4	∞	2	0

G	A	B	C	D	E	F	G
		B	D	D		F	
B	12	0	9	1	∞	∞	10
D	∞	1	3	0	4	3	4
F	∞	∞	∞	3	3	0	2
G	22	5	7	4	5	2	0

Note:

- Blue means updated value
- Red character means nexthop

Third Iteration:

A	A	B	C	D	E	F	G
				C	C	C	
A	0	12	8	11	14	14	15
B	12	0	4	1	5	4	5
C	8	4	0	3	6	6	7

B	A	B	C	D	E	F	G
			D		D	D	D
A	0	12	8	11	14	∞	22
B	12	0	4	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	3	0	4	3	4
G	22	5	7	4	5	2	0

C	A	B	C	D	E	F	G
		D				D	D
A	0	12	8	11	14	∞	22
B	12	0	4	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	3	0	4	3	4
E	4	5	6	4	0	3	5

D	A	B	C	D	E	F	G
			C				
B	12	0	4	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	3	0	4	3	4
E	4	5	6	4	0	3	5
F	∞	4	6	3	3	0	2
G	22	5	7	4	5	2	0

E	A	B	C	D	E	F	G
	C	D					F
C	8	4	0	3	6	6	7
D	11	1	3	0	4	3	4
E	14	5	6	4	0	3	5
F	∞	4	6	3	3	0	2

F	A	B	C	D	E	F	G
	D	D	D				
D	11	1	3	0	4	3	4
E	14	5	6	4	0	3	5
F	14	4	6	3	3	0	2
G	22	5	7	4	5	2	0

G	A	B	C	D	E	F	G
	D	D	D		F		
B	12	0	4	1	5	4	5
D	11	1	3	0	4	3	4
F	∞	4	6	3	3	0	2
G	15	5	7	4	5	2	0

Note:

- Blue means updated value
- Red character means nexthop

Fourth Iteration:

A	A	B	C	D (C)	E (C)	F (C)	G
A	0	12	8	11	14	14	15
B	12	0	4	1	5	4	5
C	8	4	0	3	6	6	7

C	A	B (D)	C	D	E	F (D)	G (D)
A	0	12	8	11	14	14	15
B	12	0	4	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	3	0	4	3	4
E	14	5	6	4	0	3	5

E	A (C)	B (D)	C	D	E	F	G (F)
C	8	4	0	3	6	6	7
D	11	1	3	0	4	3	4
E	14	5	6	4	0	3	5
F	14	4	6	3	3	0	2

G	A	B (D)	C (D)	D	E	F (F)	G
B	12	0	4	1	5	4	5
D	11	1	3	0	4	3	4
F	14	4	6	3	3	0	2
G	15	5	7	4	5	2	0

All 7 distance vector converge, the algorithm ends

B	A	B	C (D)	D	E (D)	F (D)	G (D)
A	0	12	8	11	14	14	15
B	12	0	4	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	3	0	4	3	4
G	15	5	7	4	5	2	0

D	A (C)	B	C	D	E	F	G
B	12	0	4	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	3	0	4	3	4
E	14	5	6	4	0	3	5
F	14	4	6	3	3	0	2
G	15	5	7	4	5	2	0

F	A (D)	B (D)	C (D)	D	E	F	G
D	11	1	3	0	4	3	4
E	14	5	6	4	0	3	5
F	14	4	6	3	3	0	2
G	15	5	7	4	5	2	0

Finally, we have B, D, E's routing table:

Table B

Destination	Distance	Next Hop
A	12	A
B	0	B
C	4	D
D	1	D
E	5	D
F	4	D
G	5	D

Table D

Destination	Distance	Next Hop
A	11	C
B	1	B
C	3	C
D	0	D
E	4	E
F	3	F
G	4	G

Table E

Destination	Distance	Next Hop
A	14	C
B	5	D
C	6	C
D	4	D
E	0	E
F	3	F
G	5	F

(3) Update CD from 3 to 30

Updated at C and D:

C	A	B	C	D	E	F	G
		D		B		D	D
A	0	12	8	11	14	14	15
B	12	0	4	1	5	4	5
C	8	4	0	10	6	6	7
D	11	1	3	0	4	3	4
E	14	5	6	4	0	3	5

D	A	B	C	D	E	F	G
		C		B			
B	12	0	4	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	5	0	4	3	4
E	14	5	6	4	0	3	5
F	14	4	6	3	3	0	2
G	15	5	7	4	5	2	0

updates in B and D:

①

B	A	B	C	D	E	F	G
			D		D	D	D
A	0	12	8	11	14	14	15
B	12	0	6	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	5	0	4	3	4
G	15	5	7	4	5	2	0

D	A	B	C	D	E	F	G
		C		B			
B	12	0	6	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	7	0	4	3	4
E	14	5	6	4	0	3	5
F	14	4	6	3	3	0	2
G	15	5	7	4	5	2	0

②

B	A	B	C	D	E	F	G
			D		D	D	D
A	0	12	8	11	14	14	15
B	12	0	8	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	7	0	4	3	4
G	15	5	7	4	5	2	0

D	A	B	C	D	E	F	G
		C		B			
B	12	0	8	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	9	0	4	3	4
E	14	5	6	4	0	3	5
F	14	4	6	3	3	0	2
G	15	5	7	4	5	2	0

③

B	A	B	C	D	E	F	G
					D	D	D
A	0	12	8	11	14	14	15
B	12	0	9	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	9	0	4	3	4
G	15	5	7	4	5	2	0

D	A	B	C	D	E	F	G
		C		B			
B	12	0	9	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	10	0	4	3	4
E	14	5	6	4	0	3	5
F	14	4	6	3	3	0	2
G	15	5	7	4	5	2	0

④

B	A	B	C	D	E	F	G
					D	D	D
A	0	12	8	11	14	14	15
B	12	0	9	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	10	0	4	3	4
G	15	5	7	4	5	2	0

D	A	B	C	D	E	F	G
		C		B			
B	12	0	9	1	5	4	5
C	8	4	0	3	6	6	7
D	11	1	10	0	4	3	4
E	14	5	6	4	0	3	5
F	14	4	6	3	3	0	2
G	15	5	7	4	5	2	0

The routing tables converge.

DV algo stops

Finally, we have B, D's routing table

Table B

Destination	Distance	Next Hop
A	12	A
B	0	B
C	9	C
D	1	D
E	5	D
F	4	D
G	5	D

Table D

Destination	Distance	Next Hop
A	11	C
B	1	B
C	10	13
D	0	D
E	4	E
F	3	F
G	4	G