COMS20001 lecture: week #21

Continued from last lecture ...



▶ We now know enough to write a (rough) algorithm for processing IP packets:

#### Algorithm (IP packet processing)

Given a packet provided by the lower, link layer:

- 1. validate header (e.g., use checksum to check for errors),
- 2. process options in header,
- 3. check destination address: if the packet is for this host
  - 3.1 buffer fragments and apply reassembly process, then eventually
  - 3.2 provide payload to a higher layer (per protocol field)

otherwise, assuming we want to forward the packet

- 3.3 check TTL, and drop if exceeded,
- 3.4 look-up next hop in forwarding table,
- 3.5 apply fragmentation and header update processes, (e.g., decrement TTL, recompute checksum),
- 3.6 transmit packet(s) via lower, link layer

and if/when errors occur, signal them appropriately.

▶ We now know enough to write a (rough) algorithm for processing IP packets:

#### Algorithm (IP packet processing)

Given a packet provided by the lower, link layer:

- 1. validate header (e.g., use checksum to check for errors),
- process options in header,
- 3. check destination address: if the packet is for this host
  - 3.1 buffer fragments and apply reassembly process, then eventually
  - 3.2 provide payload to a higher layer (per protocol field)

otherwise, assuming we want to forward the packet

- 3.3 check TTL, and drop if exceeded,
- 3.4 look-up next hop in forwarding table,
- 3.5 apply fragmentation and header update processes, (e.g., decrement TTL, recompute checksum),
- 3.6 transmit packet(s) via lower, link layer

and if/when errors occur, signal them appropriately.

- Question: how does the forwarding table get populated with entries?
- ▶ Answer: routing, which is a big topic so we'll focus on uni-cast
  - 1. distance vector routing, and
  - 2. link state routing

and hence ignore various alternatives.

### Concepts (1)

#### ► Recall:

- routing is the act of deciding a path used when forward packets from a given source to a
  given destination,
- forwarding is a *local* process, routing is *global* in the sense it involves the whole (inter-)network,
- goal is to make best use of connectivity and thus bandwidth: it can be viewed as a form of resource allocation.

#### ► Solution(s):

- 1. static (or fixed) routing, i.e., hard-code routing information by hand,
- 2. source routing, i.e., let the source pre-determine routing decisions, or
- 3. adaptive routing, e.g., i.e., use a distributed routing algorithm (or routing protocol).

### Concepts (2)

Good news: we can reason about routing by noting

 $network \equiv graph \implies graph theory \subset data structures and algorithms,$ 

#### and that

- ▶ a network graph will be weighted to capture the properties of each connection,
- ▶ we could use directed graphs (e.g., to capture uni-directional connection properties) ...
- ... but for simplicity we'll consider undirected graphs only

st. our network is modelled by

$$G = (V, E = \langle (u_0, v_0, d_0), (u_1, v_1, d_1), \dots, (u_{m-1}, v_{m-1}, d_{m-1}) \rangle)$$

where |V| = n and |E| = m.

#### Concepts (3)

Bad news: any algorithm (ideally) needs to be

correct ⇒ find paths that provide end-to-end connectivity

efficient ⇒ make good use of resources

fair ⇒ will not "stave" nodes of bandwidth

convergent ⇒ initialises/recovers quickly, e.g., after change to topology

scalable  $\Rightarrow$  remains efficient even with large n and/or m

#### and must be decentralised: the model is that

- 1. no (global) controller node exists,
- 2. nodes typically start with only local knowledge of topology,
- 3. nodes communicate (concurrently) with neighbours only, and
- 4. nodes and links can fail!

### Concepts (4)

- Recall: we have already assumed
  - routers make (global) routing decisions, whereas
  - hosts communicate locally, or forward to nearest router

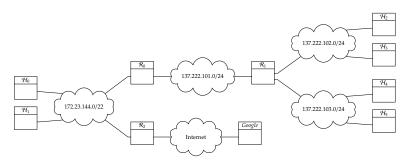
so we can route wrt. **routing units** (or regions), *not* per-node destinations.

- The strategy is to address scalablity using hierarchy, so
  - 1. route to region, then
  - 2. route in region

e.g., by leveraging IP prefixes to coalesce multiple destinations into one region (or block) ...

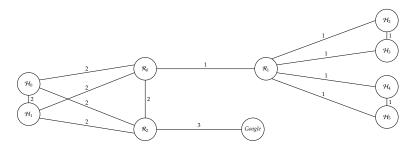
### Concepts (5)

• ... so we *significantly* simplify the problem to:



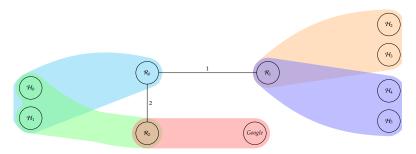
### Concepts (5)

• ... so we *significantly* simplify the problem to:



### Concepts (5)

• ... so we *significantly* simplify the problem to:



### Routing Algorithms (1)

- ► Idea: in general, we'll have each routing algorithm
  - 1. maintain some state, i.e., a routing table,
  - periodically transmit, receive and integrate information via (local) communication with neighbouring nodes,
  - periodically translate the routing table into forwarding table, e.g., via some form of (local) computation

#### keeping in mind that

- periodically means at regular intervals, plus when a change is topology occurs, and
- a cost of ∞ means a link does not exist or has failed: either way, avoid it!

► Idea:

#### **distance vector routing** $\simeq$ distributed Bellman-Ford,

in the sense each node *u* 

1. maintains a distance vector

$$\langle (v_0, d_0), (v_1, d_1), \dots, (v_{l-1}, d_{l-1}) \rangle$$

of next hop and cost tuples, initialised st.

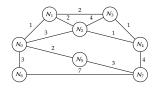
$$d_i = \begin{cases} 0 & \text{if } v_i = u \\ \infty & \text{otherwise} \end{cases},$$

- 2. periodically transmits the distance vector to all neighbours,
- 3. periodically updates the distance vector with new information, st.

$$dist(u,v) = \min_{\forall w, \ (u,w,d) \in E} \left[ dist(w,v) + d \right].$$

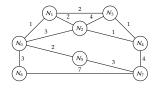
- **Example:** Routing Information Protocol (RIP) [5].
  - ▶ uses hop count as a cost function, assuming  $\infty \equiv 16$ ,
  - ► transmits distance vectors every ~ 30s with ~ 180s failure time-out.

Recalling that the graph in question is,



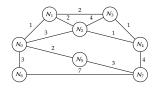
		$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$			$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	N
$N_0$	dist =	0	00	00	00	00	00	00	00	, d	dist =	0	1	3	00	00	2	3	00
/ <b>V</b> ()	hop =	1	1	1	1	1	1	1	1	$N_0$	hop =	1	$N_1$	$N_2$	1	1	$N_5$	$N_6$	1
$N_1$	dist =	00	0	00	00	00	00	00	00	$N_1$	dist =	1	0	2	2	00	00	8	00
/*1	hop =	1	1	1	1	1	1	1	1	71	hop =	$N_0$	1	$N_2$	$N_3$	1	1	1	1
N <sub>2</sub>	dist =	00	00	0	00	00	00	00	00	$N_2$	dist =	3	2	0	4	1	00	00	00
742	hop =	1	1	1	1	1	1	1	1	712	hop =	$N_0$	$N_1$	1	$N_3$	$N_4$	1	1	1
$N_3$	dist =	00	00	00	0	00	00	00	00		dist =	00	2	4	0	1	00	00	00
, •3	hop =	1	1	1	1	1	1	1	1	. ,,,	hop =	1	$N_1$	$N_2$	1	$N_4$	1	1	1
$N_4$	dist =	00	00	00	00	0	00	00	00	$N_4$	dist =	00	00	1	1	0	00	00	4
, , 4	hop =	1	1	1	1	1	_	1	_	/*4	hop =	1	1	$N_2$	$N_3$	1	1	1	$N_7$
$N_5$	dist =	00	00	00	00	00	0	00	00	N <sub>5</sub>	dist =	2	00	00	00	00	0	00	3
, •5	hop =	1	_	1	_	1	$\pm$	1	$\perp$	/•5	hop =	$N_0$	1	1	1	1	1	1	$N_7$
$N_6$	dist =	00	00	00	00	00	00	0	00	$N_6$	dist =	3	00	00	00	00	00	0	7
, *6	hop =	1	$\perp$	1	$\perp$	1	1	1	$\perp$	746	hop =	$N_0$	1	1	1	1	1	1	$N_7$
$N_7$	dist =	00	00	00	00	00	00	00	0	$N_7$	dist =	00	00	00	00	4	3	7	0
, •7	hop =	1	1	1	1	1	1	1	1	717	hop =	1	1	1	1	$N_4$	$N_5$	$N_6$	1

Recalling that the graph in question is,



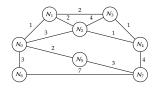
		$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$				$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	N <sub>7</sub>
$N_0$	dist =	0	1	3	00	00	2	3	00		$N_0$	dist =	0	1	3	3	4	2	3	5
/ <b>V</b> ()	hop =	1	$N_1$	$N_2$	1	1	$N_5$	$N_6$	1		/¥0	hop =	1	$N_1$	$N_2$	$N_1$	$N_2$	$N_5$	$N_6$	$N_5$
$N_1$	dist =	1	0	2	2	00	00	00	00		$N_1$	dist =	1	0	2	2	3	3	4	N <sub>5</sub> ∞ ⊥ 5 N <sub>4</sub> 5 N <sub>4</sub> 4 N <sub>7</sub>
/*1	hop =	$N_0$	1	$N_2$	$N_3$	1	1	1	1		741	hop =	$N_0$	1	$N_2$	$N_3$	$N_2$	$N_0$	$N_0$	
$N_2$	dist =	3	2	0	4	1	00	00	00		$N_2$	dist =	3	2	0	2	1	5	6	5
742	hop =	$N_0$	$N_1$	1	$N_3$	$N_4$	1	1	1		742	hop =	$N_0$	$N_1$	1	$N_4$	$N_4$	$N_0$	$N_0$	
$N_3$	dist =	00	2	4	0	1	00	00	00	$\sim$	$N_3$	dist =	3	2	2	0	1	00	00	5
743	hop =	1	$N_1$	$N_2$	1	$N_4$	1	1	1	-,	743	hop =	$N_1$	$N_1$	$N_4$	1	$N_4$	1	1	$N_4$
$N_4$	dist =	00	00	1	1	0	00	00	4		$N_4$	dist =	4	3	1	1	0	7	11	
/14	hop =	1	1	$N_2$	$N_3$	1	1	1	$N_7$		744	hop =	$N_2$	$N_2$	$N_2$	$N_3$	1	$N_7$	$N_7$	
$N_5$	dist =	2	00	00	00	00	0	00	3		N <sub>5</sub>	dist =	2	3	5	00	7	0	5	
/ 15	hop =	$N_0$	$\perp$	1	$\perp$	1	$\perp$	1	$N_7$		745	hop =	$N_0$	$N_0$	$N_0$	1	$N_7$	1	$N_0$	
$N_6$	dist =	3	00	00	00	00	00	0	7		$N_6$	dist =	3	4	6	00	11	5	0	
746	hop =	$N_0$	$\perp$	1	$\perp$	1	$\perp$	1	$N_7$		746	hop =	$N_0$	$N_0$	$N_0$	1	$N_7$	$N_0$	1	$N_7$
$N_7$	dist =	00	00	00	00	4	3	7	0		N <sub>7</sub>	dist =	5	00	5	5	4	3	7	0
747	hop =	1	$\perp$	1	$\perp$	$N_4$	$N_5$	$N_6$	1		747	hop =	$N_5$	1	$N_4$	$N_4$	$N_4$	$N_5$	$N_6$	1

Recalling that the graph in question is,



		$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$			$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	
$N_0$	dist =	0	1	3	3	4	2	3	5	$N_0$	dist =	0	1	3	3	4	2	3	ŀ
/ <b>V</b> ()	hop =	1	$N_1$	$N_2$	$N_1$	$N_2$	$N_5$	$N_6$	$N_5$	/NO	hop =	1	$N_1$	N <sub>2</sub>	$N_1$	$N_2$	$N_5$	$N_6$	l
$N_1$	dist =	1	0	2	2	3	3	4	00	$\mathcal{N}_1$	dist =	1	0	2	2	3	3	4	Π
/V1	hop =	$N_0$	1	$N_2$	$N_3$	$N_2$	$N_0$	$N_0$	1	N1	hop =	$N_0$	1	$N_2$	$N_3$	$N_2$	$N_0$	$N_0$	İ
$N_2$	dist =	3	2	0	2	1	5	6	$\begin{array}{c c} \hline \\ \hline 5 \\ N_4 \\ \hline 5 \\ N_4 \\ \end{array} \longrightarrow \begin{array}{c} N \\ N \\ \end{array}$	dist =	3	2	0	2	1	5	6	T	
/12	hop =	$N_0$	$N_1$	1	$N_4$	$N_4$	$N_0$	$N_0$	$N_4$	/N <sub>2</sub>	hop =	$N_0$	$N_1$	1	$N_4$	$N_4$	$N_0$	$N_0$	İ
$N_3$	dist =	3	2	2	0	1	00	00	5	∧	dist =	3	2	2	0	1	5	6	T
743	hop =	$N_1$	$N_1$	$N_4$	1	$N_4$	1	1	$N_4$		hop =	$N_1$	$N_1$	N <sub>4</sub>	1	$N_4$	$N_1$	$N_1$	١.
$N_4$	dist =	4	3	1	1	0	7	11	4	$N_4$	dist =	4	3	1	1	0	6	7	Γ
/14	hop =	$N_2$	$N_2$	$N_2$	$N_3$	1	$N_7$	$N_7$	$N_7$	714	hop =	$N_2$	$N_2$	$N_2$	$N_3$	1	$N_2$	$N_2$	l
$N_5$	dist =	2	3	5	00	7	0	5	3	N <sub>5</sub>	dist =	2	3	5	5	6	0	5	Γ
/ <b>v</b> 5	hop =	$N_0$	$N_0$	$N_0$	1	$N_7$	1	$N_0$	$N_7$	/N <sub>5</sub>	hop =	$N_0$	$N_0$	$N_0$	$N_0$	$N_0$	1	$N_0$	l
$N_6$	dist =	3	4	6	00	11	5	0	7	$N_6$	dist =	3	4	6	6	7	5	0	Τ
/ <b>v</b> 6	hop =	$N_0$	$N_0$	$N_0$	1	$N_7$	$N_0$	1	$N_7$	N <sub>6</sub>	hop =	$N_0$	$N_0$	$N_0$	$N_0$	$N_0$	$N_0$	1	l
$N_7$	dist =	5	00	5	5	4	3	7	0	N <sub>7</sub>	dist =	5	6	5	5	4	3	7	Γ
117	hop =	$N_5$	1	$N_4$	$N_4$	$N_4$	$N_5$	$N_6$	1	/N <sub>7</sub>	hop =	$N_5$	$N_5$	$N_4$	$N_4$	$N_4$	$N_5$	$N_6$	ı

Recalling that the graph in question is,



		$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$			$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	
$N_0$	dist =	0	1	3	3	4	2	3	5		dist =	0	1	3	3	4	2	3	1
/ <b>V</b> ()	hop =	1	$N_1$	$N_2$	$N_1$	$N_2$	$N_5$	$N_6$	$N_5$	$N_0$	hop =	1	$N_1$	$N_2$	$N_1$	$N_2$	$N_5$	$N_6$	ı
$N_1$	dist =	1	0	2	2	3	3	4	6	$\mathcal{N}_1$	dist =	1	0	2	2	3	3	4	1
741	hop =	$N_0$	1	$N_2$	$N_3$	$N_2$	$N_0$	$N_0$	$N_0$	71	hop =	$N_0$	1	$N_2$	$N_3$	$N_2$	$N_0$	$N_0$	١
$N_2$	dist =	3	2	0	2	1	5	6	5	$N_2$	dist =	3	2	0	2	1	5	6	İ
742	hop =	$N_0$	$N_1$	1	$N_4$	$N_4$	$N_0$	$N_0$	$N_4$	712	hop =	$N_0$	$N_1$	1	$N_4$	$N_4$	$N_0$	$N_0$	١
$N_3$	dist =	3	2	2	0	1	5	6	5	$\sim \rightarrow N_3$	dist =	3	2	2	0	1	5	6	İ
743	hop =	$N_1$	$N_1$	$N_4$	1	$N_4$	$N_1$	$N_1$	$N_4$	-, ,,,3	hop =	$N_1$	$N_1$	$N_4$	1	$N_4$	$N_1$	$N_1$	١
$N_4$	dist =	4	3	1	1	0	6	7	4	$N_4$	dist =	4	3	1	1	0	6	7	Ī
/*4	hop =	$N_2$	$N_2$	$N_2$	$N_3$	1	$N_2$	$N_2$	$N_7$	714	hop =	$N_2$	$N_2$	$N_2$	$N_3$	1	$N_2$	$N_2$	١
$N_5$	dist =	2	3	5	5	6	0	5	3	N <sub>5</sub>	dist =	2	3	5	5	6	0	5	Ī
/15	hop =	$N_0$	$N_0$	$N_0$	$N_0$	$N_0$	1	$N_0$	$N_7$	715	hop =	$N_0$	$N_0$	$N_0$	$N_0$	$N_0$	1	$N_0$	١
$N_6$	dist =	3	4	6	6	7	5	0	7	$N_6$	dist =	3	4	6	6	7	5	0	I
, v <sub>6</sub>	hop =	$N_0$	$N_0$	$N_0$	$N_0$	$N_0$	$N_0$	1	$N_7$	Ν <sub>6</sub>	nop =	$N_0$	$N_0$	$N_0$	$N_0$	$N_0$	$N_0$	1	
$N_7$	dist =	5	6	5	5	4	3	7	0	N <sub>7</sub>	dist =	5	6	5	5	4	3	7	1
747	hop =	$N_5$	$N_5$	$N_4$	$N_4$	$N_4$	$N_5$	$N_6$	1	747	hop =	$N_5$	$N_5$	$N_4$	$N_4$	$N_4$	$N_5$	$N_6$	1

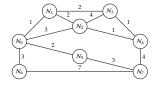
► Idea:

**link state routing** 
$$\simeq$$
 flooding + Dijkstra,

in the sense each node u

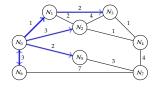
- floods network with link state packets (i.e., neighbouring nodes plus link costs) yielding global topology, then
- 2. use Dijkstra to compute shortest paths.
- ► Examples:
  - Open Shortest Path First (OSPF) [7], and
  - ► Intermediate System to Intermediate System (IS-IS) [8].

Recalling that the graph in question is,



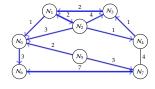
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



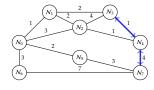
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



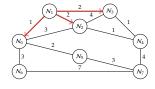
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



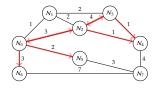
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



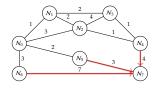
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



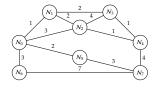
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



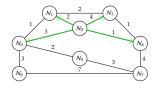
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



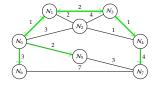
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



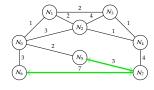
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



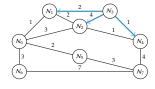
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



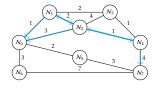
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



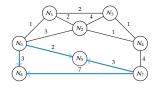
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



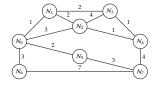
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



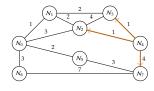
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



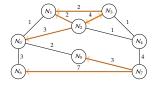
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



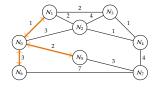
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



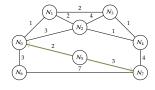
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



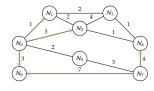
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



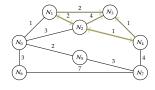
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



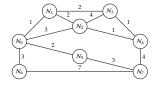
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



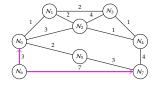
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



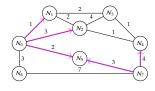
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



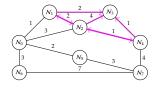
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



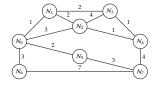
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



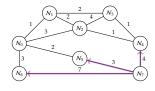
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



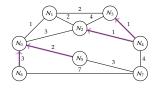
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



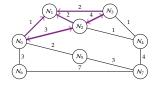
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



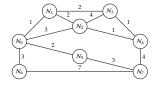
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



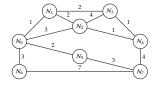
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



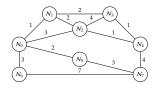
- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

Recalling that the graph in question is,



- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra.

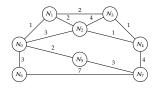
Recalling that the graph in question is,



- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra, i.e.,

		$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$	
	dist =	0	00	00	00	00	00	00	00	((A) (A) (A) (A) (A) (A) (A) (A) (A) (A
$N_0$	hop =	1	1	1	1	1	1	1	1	queue = $\langle (N_0, 0), (N_1, \infty), (N_2, \infty), (N_3, \infty), (N_4, \infty), (N_5, \infty), (N_6, \infty), (N_7, \infty) \rangle$
$N_1$	dist =	00	0	00	00	00	00	00	00	queue = $\langle (N_0, \infty), (N_1, 0), (N_2, \infty), (N_3, \infty), (N_4, \infty), (N_5, \infty), (N_6, \infty), (N_7, \infty) \rangle$
/ 1	hop =	1	1	1	1	1	1	1	1	$ q_{1}  = ((v_0, \omega), (v_1, \omega), (v_2, \omega), (v_3, \omega), (v_4, \omega), (v_5, \omega), (v_6, \omega), (v_7, \omega))$
N <sub>2</sub>	dist =	00	00	0	00	00	00	00	00	queue = $\langle (N_0, \infty), (N_1, \infty), (N_2, 0), (N_3, \infty), (N_4, \infty), (N_5, \infty), (N_6, \infty), (N_7, \infty) \rangle$
742	hop =	1	1	1	1	1	1	1	1	
$N_3$	dist =	00	00	00	0	00	00	00	00	queue = $\langle (N_0, \infty), (N_1, \infty), (N_2, \infty), (N_3, 0), (N_4, \infty), (N_5, \infty), (N_6, \infty), (N_7, \infty) \rangle$
743	hop =	1	1	1	1	1	1	1	1	
$N_4$	dist =	00	00	00	00	0	00	00	00	queue = $\langle (N_0, \infty), (N_1, \infty), (N_2, \infty), (N_3, \infty), (N_4, 0), (N_5, \infty), (N_6, \infty), (N_7, \infty) \rangle$
744	hop =	1	1	1	1	1	1	1	1	quene = \(\(\cute{\cutee{\cute{\cute{\cute\cute{\cute{\cute{\cute{\cute{\cute{\cute{\cute{\cute{\cute{\cut
$N_5$	dist =	00	00	00	00	00	0	00	00	queue = $\langle (N_0, \infty), (N_1, \infty), (N_2, \infty), (N_3, \infty), (N_4, \infty), (N_5, 0), (N_6, \infty), (N_7, \infty) \rangle$
7.5	hop =	1	1	1	1	1	1	1	1	
$N_6$	dist =	00	00	00	00	00	00	0	00	queue = $\langle (N_0, \infty), (N_1, \infty), (N_2, \infty), (N_3, \infty), (N_4, \infty), (N_5, \infty), (N_6, 0), (N_7, \infty) \rangle$
746	hop =	1	1	1	1	1	1	1	1	
$N_7$	dist =	00	00	00	00	00	00	00	0	queue = $\langle (N_0, \infty), (N_1, \infty), (N_2, \infty), (N_3, \infty), (N_4, \infty), (N_5, \infty), (N_6, \infty), (N_7, 0) \rangle$
7.47	hop =	1	1	1	1	1	1	1	1	queue = \(\(\nu_0,\infty),\(\nu_1,\infty),\(\nu_2,\infty),\(\nu_3,\infty),\(\nu_4,\infty),\(\nu_5,\infty),\(

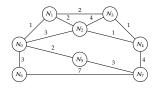
Recalling that the graph in question is,



- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra, i.e.,

		$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$	
	dist =	0	1	3	00	00	2	3	00	1
$N_0$	hop =	1	$N_1$	$N_2$	1	1	$N_5$	$N_6$	1	queue = $\langle (N_1, 1), (N_2, 3), (N_3, \infty), (N_4, \infty), (N_5, 2), (N_6, 3), (N_7, \infty) \rangle$
$N_1$	dist =	1	0	2	2	00	00	00	00	((A) 1) (A) 2) (A) 2) (A)> (A)> (A)> (A)
/ <b>V</b> 1	hop =	$N_0$	1	$N_2$	$N_3$	1	1	1	1	queue = $\langle (N_0, 1), (N_2, 2), (N_3, 2), (N_4, \infty), (N_5, \infty), (N_6, \infty), (N_7, \infty) \rangle$
N <sub>2</sub>	dist =	3	2	0	4	1	00	00	00	((A( 2) (A( 2) (A( 4) (A( 4) (A() (A() (A())
/ <b>V</b> 2	hop =	$N_0$	$N_1$	1	$N_3$	$N_4$	1	1	1	queue = $\langle (N_0, 3), (N_1, 2), (N_3, 4), (N_4, 1), (N_5, \infty), (N_6, \infty), (N_7, \infty) \rangle$
$N_3$	dist =	00	2	4	0	1	00	00	00	((A() (A( 2) (A( A) (A( 1) (A() (A() (A())
/ <b>V</b> 3	hop =	1	$N_1$	$N_2$	1	$N_4$	1	1	1	queue = $\langle (N_0, \infty), (N_1, 2), (N_2, 4), (N_4, 1), (N_5, \infty), (N_6, \infty), (N_7, \infty) \rangle$
	dist =	00	00	1	1	0	00	00	4	((A() (A() (A( 4) (A( 4) (A() (A() (A( 4))
$N_4$	hop =	1	1	$N_2$	$N_3$	1	1	1	$N_7$	queue = $\langle (N_0, \infty), (N_1, \infty), (N_2, 1), (N_3, 1), (N_5, \infty), (N_6, \infty), (N_7, 4) \rangle$
.,	dist =	2	00	00	00	00	0	00	3	1
$N_5$	hop =	$N_0$	1	1	1	1	1	1	N <sub>7</sub>	queue = $\langle (N_0, 2), (N_1, \infty), (N_2, \infty), (N_3, \infty), (N_4, \infty), (N_6, \infty), (N_7, 3) \rangle$
.,	dist =	3	00	00	00	00	00	0	7	1
$N_6$	hop =	$N_0$	1	1	1	1	1	1	$N_7$	queue = $\langle (N_0, 3), (N_1, \infty), (N_2, \infty), (N_3, \infty), (N_4, \infty), (N_5, \infty), (N_7, 7) \rangle$
.,	dist =	00	00	00	00	4	3	7	0	1
$N_7$	hop =	1	1	1	1	$N_4$	$N_5$	$N_6$	1	queue = $\langle (N_0, \infty), (N_1, \infty), (N_2, \infty), (N_3, \infty), (N_4, 4), (N_5, 3), (N_6, 7) \rangle$

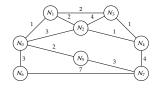
Recalling that the graph in question is,



- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra, i.e.,

		$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$	
$N_0$	dist =	0	1	3	3	00	2	3	00	queue = $\langle (N_2, 3), (N_3, 3), (N_4, \infty), (N_5, 2), (N_6, 3), (N_7, \infty) \rangle$
/¥0	hop =	1	$N_1$	$N_2$	$N_1$	1	$N_5$	$N_6$	1	queue = ((N2, 3), (N3, 3), (N4, \infty), (N5, 2), (N6, 3), (N7, \infty)
$N_1$	dist =	1	0	2	2	00	3	4	00	queue = $\langle (N_2, 2), (N_3, 2), (N_4, \infty), (N_5, 3), (N_6, 4), (N_7, \infty) \rangle$
/*1	hop =	$N_0$	1	$N_2$	$N_3$	1	$N_0$	$N_0$	1	queue = \(\(\nu_2, 2\), \(\nu_3, 2\), \(\nu_4, \infty), \((\nu_5, 3\), \((\nu_6, \nu_7), \((\nu_7, \infty)\)
$N_2$	dist =	3	2	0	2	1	00	00	5	queue = $\langle (N_0, 3), (N_1, 2), (N_3, 2), (N_5, \infty), (N_6, \infty), (N_7, 5) \rangle$
742	hop =	$N_0$	$N_1$	1	$N_4$	$N_4$	1	1	$N_4$	queue = \(\(\mathbf{v}_0\),\(\mathbf{v}_1\),\(\mathbf{v}_1\),\(\mathbf{v}_1\),\(\mathbf{v}_3\),\(\mathbf{v}_5\),\(\mathbf{v}_5\),\(\mathbf{v}_5\),\(\mathbf{v}_6\),\(\mathbf{v}_7\),\(\mathbf{v}_
$N_3$	dist =	00	2	2	0	1	00	00	5	queue = $\langle (N_0, \infty), (N_1, 2), (N_2, 2), (N_5, \infty), (N_6, \infty), (N_7, 5) \rangle$
743	hop =	1	$N_1$	$N_4$	1	$N_4$	1	1	$N_4$	queue = \(\(\mathbf{v}_0\), \(\mathbf{v}_1\), \(\mathbf{v}_2\), \(\mathbf{v}_2\), \(\mathbf{v}_3\), \(\mathbf{v}_6\), \(\mathbf{v}_7\), \(
$N_4$	dist =	4	3	1	1	0	00	00	4	queue = $\langle (N_0, 4), (N_1, 3), (N_3, 1), (N_5, \infty), (N_6, \infty), (N_7, 4) \rangle$
/*4	hop =	$N_2$	$N_2$	$N_2$	$N_3$	1	1	1	$N_7$	queue = \(\(\sigma_0, \pi_3, (\sigma_1, \sigma_1, \sigma_1, \sigma_3, \pi_1), (\sigma_5, \infty), (\sigma_5, \infty), (\sigma_7, \pi_1)
$N_5$	dist =	2	3	5	00	00	0	5	3	queue = $\langle (N_1, 3), (N_2, 5), (N_3, \infty), (N_4, \infty), (N_6, 5), (N_7, 3) \rangle$
745	hop =	$N_0$	$N_0$	$N_0$	1	1	1	$N_0$	$N_7$	queue = \(\(\varphi_1, 5\), \(\varphi_2, 5\), \(\varphi_3, \omega), \((\varphi_4, \omega), \((\varphi_6, 5\), \((\varphi_7, 5\))\)
$N_6$	dist =	3	4	6	00	00	5	0	7	queue = $\langle (N_1, 4), (N_2, 6), (N_3, \infty), (N_4, \infty), (N_5, 5), (N_7, 7) \rangle$
746	hop =	$N_0$	$N_0$	$N_0$	1	1	$N_0$	1	$N_7$	queue = \(\(\mathbf{v}_1\), \(\mathbf{v}_2\), \(\mathbf{v}_3\), \(\mathbf{v}_3\), \(\mathbf{v}_4\), \(\mathbf{v}_3\), \(
$N_7$	dist =	5	00	00	00	4	3	7	0	queue = $\langle (N_0, 5), (N_1, \infty), (N_2, \infty), (N_3, \infty), (N_4, 4), (N_6, 7) \rangle$
747	hop =	$N_5$	1	1	1	$N_4$	$N_5$	$N_6$	1	queue = ((\vi_0, 5), (\vi_1, \infty), (\vi_2, \infty), (\vi_3, \infty), (\vi_4, \vi_1), (\vi_6, \vi_1)

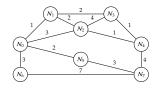
Recalling that the graph in question is,



- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra, i.e.,

		$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$	
. , d	list =	0	1	3	3	00	2	3	5	]
$N_0$ h	юр =	1	$N_1$	$N_2$	$N_1$	1	$N_5$	$N_6$	$N_5$	queue = $\langle (N_2, 3), (N_3, 3), (N_4, \infty), (N_6, 3), (N_7, 5) \rangle$
$N_1$ d	list =	1	0	2	2	3	3	4	00	((A) 2) (A) 2) (A) 2) (A) (A) (A) (A)
$N_1$ h	юр =	$N_0$	1	$N_2$	$N_3$	$N_2$	$N_0$	$N_0$	1	queue = $\langle (N_3, 2), (N_4, 3), (N_5, 3), (N_6, 4), (N_7, \infty) \rangle$
No di	list =	3	2	0	2	1	00	00	5	((A) 2) (A) 2) (A)> (A)> (A)>
N <sub>2</sub> h	юр =	$N_0$	$N_1$	1	$N_4$	$N_4$	1	1	$N_4$	queue = $\langle (N_0, 3), (N_3, 2), (N_5, \infty), (N_6, \infty), (N_7, 5) \rangle$
N <sub>3</sub> d	list =	3	2	2	0	1	00	00	5	queue = $\langle (N_0, 3), (N_2, 2), (N_5, \infty), (N_6, \infty), (N_7, 5) \rangle$
N <sub>3</sub> h	юр =	$N_1$	$N_1$	$N_4$	1	$N_4$	1	1	$N_4$	queue = $((N_0, 3), (N_2, 2), (N_5, \infty), (N_6, \infty), (N_7, 3))$
$N_4$ d	list =	4	3	1	1	0	00	00	4	((A) (A) (A) (A) () (A) () (A)
N <sub>4</sub> h	iop =	$N_2$	$N_2$	$N_2$	$N_3$	1	1	1	$N_7$	queue = $\langle (N_0, 4), (N_1, 3), (N_5, \infty), (N_6, \infty), (N_7, 4) \rangle$
. , d	list =	2	3	5	5	00	0	5	3	1
$N_5$ $h$	юр =	$N_0$	$N_0$	$N_0$	$N_1$	1	1	$N_0$	N <sub>7</sub>	queue = $\langle (N_2, 5), (N_3, 5), (N_4, \infty), (N_6, 5), (N_7, 3) \rangle$
. , d	list =	3	4	6	6	00	5	0	7	1
N <sub>6</sub>	юр =	$N_0$	$N_0$	$N_0$	$N_1$	1	$N_0$	1	$N_7$	queue = $\langle (N_2, 6), (N_3, 6), (N_4, \infty), (N_5, 5), (N_7, 7) \rangle$
., d:	list =	5	00	5	5	4	3	7	0	(/A/ E) (A/) (A/ E) (A/ E) (A/ E)
$N_7$ h	iop =	$N_5$	1	$N_4$	$N_4$	$N_4$	$N_5$	$N_6$	1	queue = $\langle (N_0, 5), (N_1, \infty), (N_2, 5), (N_3, 5), (N_6, 7) \rangle$

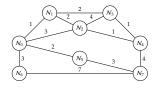
Recalling that the graph in question is,



- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra, i.e.,

		$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$	
	dist =	0	1	3	3	4	2	3	5	((4( 2) (4( 4) (4( 2) (4( 5))
$N_0$	hop =	1	$N_1$	$N_2$	$N_1$	$N_2$	$N_5$	$N_6$	$N_5$	queue = $\langle (N_3, 3), (N_4, 4), (N_6, 3), (N_7, 5) \rangle$
$N_1$	dist =	1	0	2	2	3	3	4	00	queue = $\langle (N_4, 3), (N_5, 3), (N_6, 4), (N_7, \infty) \rangle$
/ 1	hop =	$N_0$	1	$N_2$	$N_3$	$N_2$	$N_0$	$N_0$	1	queue = ((14,3), (145,3), (146,4), (147, 20))
N <sub>2</sub>	dist =	3	2	0	2	1	00	00	5	queue = $\langle (N_0, 3), (N_5, \infty), (N_6, \infty), (N_7, 5) \rangle$
/ <b>v</b> 2	hop =	$N_0$	$N_1$	1	$N_4$	$N_4$	1	1	$N_4$	$queue = ((1 \mathbf{v}_0, 3), (1 \mathbf{v}_5, \infty), (1 \mathbf{v}_6, \infty), (1 \mathbf{v}_7, 3))$
$N_3$	dist =	3	2	2	0	1	00	00	5	queue = $\langle (N_0, 3), (N_5, \infty), (N_6, \infty), (N_7, 5) \rangle$
/ <b>v</b> 3	hop =	$N_1$	$N_1$	$N_4$	1	$N_4$	1	1	$N_4$	queue = ((1,0,3), (1,0,0), (1,0,0), (1,0,0))
$N_4$	dist =	4	3	1	1	0	00	00	4	queue = $\langle (N_0, 4), (N_5, \infty), (N_6, \infty), (N_7, 4) \rangle$
784	hop =	$N_2$	$N_2$	$N_2$	$N_3$	1	1	1	$N_7$	queue = ((/v <sub>0</sub> , 4), (/v <sub>5</sub> , ∞), (/v <sub>6</sub> , ∞), (/v <sub>7</sub> , 4))
$N_5$	dist =	2	3	5	5	7	0	5	3	queue = $\langle (N_2, 5), (N_3, 5), (N_4, 7), (N_6, 5) \rangle$
/ <b>V</b> 5	hop =	$N_0$	$N_0$	$N_0$	$N_1$	$N_7$	1	$N_0$	$N_7$	queue = ((/v2, 3), (/v3, 3), (/v4, 7), (/v6, 3))
$N_6$	dist =	3	4	6	6	00	5	0	7	queue = $\langle (N_2, 6), (N_3, 6), (N_4, \infty), (N_7, 7) \rangle$
746	hop =	$N_0$	$N_0$	$N_0$	$N_1$	1	$N_0$	1	$N_7$	queue = ((1\v2, 6), (1\v3, 6), (1\v4, \infty), (1\v7, 7))
$N_7$	dist =	5	6	5	5	4	3	7	0	queue = $\langle (N_1, 6), (N_2, 5), (N_3, 5), (N_6, 7) \rangle$
/17	hop =	$N_5$	$N_0$	$N_4$	$N_4$	$N_4$	$N_5$	$N_6$	1	queue = ((\v_1, 0), (\v_2, 3), (\v_3, 3), (\v_6, 1))

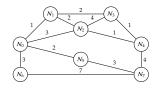
Recalling that the graph in question is,



- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra, i.e.,

		$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$	
$N_0$	dist =	0	1	3	3	4	2	3	5	20000 = /(A/ 4) (A/ 2) (A/ E)
/ <b>v</b> <sub>0</sub>	hop =	1	$N_1$	$N_2$	$N_1$	$N_2$	$N_5$	$N_6$	$N_5$	queue = $\langle (N_4, 4), (N_6, 3), (N_7, 5) \rangle$
$N_1$	dist =	1	0	2	2	3	3	4	7	queue = $\langle (N_5, 3), (N_6, 4), (N_7, 7) \rangle$
741	hop =	$N_0$	1	$N_2$	$N_3$	$N_2$	$N_0$	$N_0$	$N_4$	queue = ((145,5),(146,4),(147,7))
$N_2$	dist =	3	2	0	2	1	5	6	5	queue = $\langle (N_5, 5), (N_6, 6), (N_7, 5) \rangle$
742	hop =	$N_0$	$N_1$	1	$N_4$	$N_4$	$N_0$	$N_0$	$N_4$	quene = ((145,5), (146,6), (147,5))
$N_3$	dist =	3	2	2	0	1	5	6	5	queue = $\langle (N_5, 5), (N_6, 6), (N_7, 5) \rangle$
743	hop =	$N_1$	$N_1$	$N_4$	1	$N_4$	$N_0$	$N_0$	$N_4$	quene = ((145,5), (146,6), (147,5))
$N_4$	dist =	4	3	1	1	0	6	7	4	queue = $\langle (N_5, 6), (N_6, 7), (N_7, 4) \rangle$
744	hop =	$N_2$	$N_2$	$N_2$	$N_3$	1	$N_0$	$N_0$	$N_7$	quene = ((145,0),(146,1),(147,4))
$N_5$	dist =	2	3	5	5	6	0	5	3	queue = $\langle (N_3, 5), (N_4, 6), (N_6, 5) \rangle$
715	hop =	$N_0$	$N_0$	$N_0$	$N_1$	$N_2$	1	$N_0$	$N_7$	quene = ((143, 5), (144, 6), (146, 5))
$N_6$	dist =	3	4	6	6	7	5	0	7	queue = $\langle (N_3, 6), (N_4, 7), (N_7, 7) \rangle$
716	hop =	$N_0$	$N_0$	$N_0$	$N_1$	$N_2$	$N_0$	1	$N_7$	queue = ((143,0),(144,1),(147,1))
$N_7$	dist =	5	6	5	5	4	3	7	0	queue = $\langle (N_1, 6), (N_3, 5), (N_6, 7) \rangle$
747	hop =	$N_5$	$N_0$	$N_4$	$N_4$	$N_4$	$N_5$	$N_6$	1	queue = ((141,0),(143,5),(146,1))

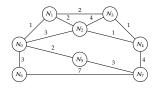
Recalling that the graph in question is,



- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra, i.e.,

		$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$	
$N_0$	dist =	0	1	3	3	4	2	3	5	queue = $\langle (N_4, 4), (N_7, 5) \rangle$
/¥0	hop =	1	$N_1$	$N_2$	$N_1$	$N_2$	$N_5$	$N_6$	$N_5$	queue = \((/\v4,4),(/\v7,3)/
$N_1$	dist =	1	0	2	2	3	3	4	6	queue = $\langle (N_6, 4), (N_7, 6) \rangle$
/*1	hop =	$N_0$	1	$N_2$	$N_3$	$N_2$	$N_0$	$N_0$	$N_5$	quene = \((\vec{v}_6, \vec{v}_7, (\vec{v}_7, 0))
N <sub>2</sub>	dist =	3	2	0	2	1	5	6	5	queue = $\langle (N_6, 6), (N_7, 5) \rangle$
/ 1/2	hop =	$N_0$	$N_1$	1	$N_4$	$N_4$	$N_0$	$N_0$	$N_4$	quene = \((/*6,0),(/*/,5)/
$N_3$	dist =	3	2	2	0	1	5	6	5	queue = $\langle (N_6, 6), (N_7, 5) \rangle$
743	hop =	$N_1$	$N_1$	$N_4$	1	$N_4$	$N_0$	$N_0$	$N_4$	quene = \((\vec{v}_6,0),(\vec{v}_7,3))
$N_4$	dist =	4	3	1	1	0	6	7	4	queue = $\langle (N_5, 6), (N_6, 7) \rangle$
7*4	hop =	$N_2$	$N_2$	$N_2$	$N_3$	1	$N_0$	$N_0$	$N_7$	queue = ((/45,0),(/46,7)/
$N_5$	dist =	2	3	5	5	6	0	5	3	queue = $\langle (N_4, 6), (N_6, 5) \rangle$
7.5	hop =	$N_0$	$N_0$	$N_0$	$N_1$	$N_2$	1	$N_0$	$N_7$	quene = ((/44,0),(/46,5))
$N_6$	dist =	3	4	6	6	7	5	0	7	queue = $\langle (N_4, 7), (N_7, 7) \rangle$
7.0	hop =	$N_0$	$N_0$	$N_0$	$N_1$	$N_2$	$N_0$	1	$N_7$	queue = ((/•4,/),(/•/,/)/
$N_7$	dist =	5	6	5	5	4	3	7	0	queue = $\langle (N_1, 6), (N_6, 7) \rangle$
747	hop =	$N_5$	$N_0$	$N_4$	$N_4$	$N_4$	$N_5$	$N_6$	1	queue = \((/*1,0),(/*6,7)/

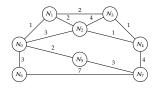
Recalling that the graph in question is,



- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra, i.e.,

		$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$	
	dist =	0	1	3	3	4	2	3	5	(/ 1/ 5)
$N_0$	hop =	1	$N_1$	$N_2$	$N_1$	$N_2$	$N_5$	$N_6$	$N_5$	$queue = \langle (N_7, 5) \rangle$
$N_1$	dist =	1	0	2	2	3	3	4	6	$queue = \langle (N_7, 6) \rangle$
/¥1	hop =	$N_0$	1	$N_2$	$N_3$	$N_2$	$N_0$	$N_0$	$N_5$	queue = \((/\v7,0)/
$N_2$	dist =	3	2	0	2	1	5	6	5	$queue = \langle (N_6, 6) \rangle$
/¥2	hop =	$N_0$	$N_1$	1	$N_4$	$N_4$	$N_0$	$N_0$	$N_4$	queue = \((/\mathbf{v}_6, 0))
$N_3$	dist =	3	2	2	0	1	5	6	5	queue = $\langle (N_6, 6) \rangle$
/ <b>v</b> 3	hop =	$N_1$	$N_1$	$N_4$	1	$N_4$	$N_0$	$N_0$	$N_4$	queue = \((/\mathbf{v}_6, 0))
$N_4$	dist =	4	3	1	1	0	6	7	4	queue = $\langle (N_6, 7) \rangle$
744	hop =	$N_2$	$N_2$	$N_2$	$N_3$	1	$N_0$	$N_0$	$N_7$	queue = \((/\mathbf{v}_6,7))
$N_5$	dist =	2	3	5	5	6	0	5	3	$queue = \langle (N_4, 6) \rangle$
/ <b>V</b> 5	hop =	$N_0$	$N_0$	$N_0$	$N_1$	$N_2$	1	$N_0$	$N_7$	queue = ((/ <b>v</b> 4, 6)/
$N_6$	dist =	3	4	6	6	7	5	0	7	$queue = \langle (N_7, 7) \rangle$
746	hop =	$N_0$	$N_0$	$N_0$	$N_1$	$N_2$	$N_0$	1	$N_7$	queue = \((/\v7,7))
N <sub>7</sub>	dist =	5	6	5	5	4	3	7	0	augus = //A/ 700
/ 17	hop =	$N_5$	$N_0$	$N_4$	$N_4$	$N_4$	$N_5$	$N_6$	1	$queue = \langle (N_6, 7) \rangle$

Recalling that the graph in question is,



- 1. uses flooding to communicate the topology to all nodes,
- 2. has each node (in parallel) compute shortest paths using Dijkstra, i.e.,

		$N_0$	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$	
	dist =	0	1	3	3	4	2	3	5	queue = Ø
$N_0$	hop =	1	$N_1$	$N_2$	$N_1$	$N_2$	$N_5$	$N_6$	$N_5$	queue – v
$N_1$	dist =	1	0	2	2	3	3	4	6	queue = Ø
/V <sub>1</sub>	hop =	$N_0$	1	$N_2$	$N_3$	$N_2$	$N_0$	$N_0$	$N_5$	queue = v
	dist =	3	2	0	2	1	5	6	5	a
$N_2$	hop =	$N_0$	$N_1$	1	$N_4$	$N_4$	$N_0$	$N_0$	$N_4$	queue = Ø
$N_3$	dist =	3	2	2	0	1	5	6	5	queue = Ø
/ <b>V</b> 3	hop =	$N_1$	$N_1$	$N_4$	1	$N_4$	$N_0$	$N_0$	$N_4$	queue = v
$N_4$	dist =	4	3	1	1	0	6	7	4	queue = Ø
/44	hop =	$N_2$	$N_2$	$N_2$	$N_3$	1	$N_0$	$N_0$	$N_7$	queue – v
$N_5$	dist =	2	3	5	5	6	0	5	3	queue = Ø
/ <b>v</b> 5	hop =	$N_0$	$N_0$	$N_0$	$N_1$	$N_2$	1	$N_0$	$N_7$	queue – v
$N_6$	dist =	3	4	6	6	7	5	0	7	queue = Ø
,¥6	hop =	$N_0$	$N_0$	$N_0$	$N_1$	$N_2$	$N_0$	1	N <sub>7</sub>	queue - v
$N_7$	dist =	5	6	5	5	4	3	7	0	queue = Ø
, <b>v</b> 7	hop =	$N_5$	$N_0$	$N_4$	$N_4$	$N_4$	$N_5$	$N_6$	1	queue - v

#### Conclusions

▶ To summarise the two approaches covered, we can say

Metric	Distance Vector	Link State
correct	distributed Bellman-Ford	replicated Dijkstra
efficient	approximately	approximately
fair	approximately	approximately
convergent	slow: many exchanges	fast: flood then recompute
scalable	excellent	reasonable

i.e., selection is basically a trade-off between convergence and scalablity  $\dots$ 

- ... *plus* we need to cater for various corner cases:
  - distance vector routing can fail if
    - if network is partitioned (cf. graph cut, meaning "count to infinity" problem),
       while
  - ▶ link state routing can fail if
    - flooding sequence numbers can overflow or be corrupted,
    - nodes can fail and reset flooding sequence number,
    - if network is partitioned and then re-joined.

#### Conclusions

#### Take away points:

- Using graph theory to study routing is advantageous in terms of designing and reasoning about solutions ...
- ... even so, translating theory into practice is *still* a significant challenge wrt. function and efficiency.
   Pouting protocols are instances of more concret concerns protocols whereby (distributed)
- Routing protocols are instances of more general consensus protocols whereby (distributed) parties need to agree on a shared state.
- ► Additional topics: a (non-exhaustive) list could include at least
  - the Border Gateway Protocol (BGP) [6], which is important for large(r) scale inter-networking, and
  - techniques such as multi-path routing [10] which allows packets to make use of multiple routes to a given destination.

#### References

- Wikipedia: Distance vector routing protocol. http://en.wikipedia.org/wiki/Distance-vector\_routing\_protocol.
- [2] Wikipedia: Link-state routing protocol. http://en.wikipedia.org/wiki/Link-state\_routing\_protocol.
- [3] Wikipedia: Routing. http://en.wikipedia.org/wiki/Routing.
- [4] Wikipedia: Routing protocol. http://en.wikipedia.org/wiki/Routing\_protocol.
- [5] C. Hedrick.
   Routing Information Protocol.
   Internet Engineering Task Force (IETF) Request for Comments (RFC) 1058, 1988.
   http://tools.ietf.org/html/rfc1058.
- [6] K. Loughheed and Y. Rekhter. A Border Gateway Protocol (BGP). Internet Engineering Task Force (IETF) Request for Comments (RFC) 1105, 1989. http://tools.ietf.org/html/rfc1105.
- J. Moy.
   OSPF specification.
   Internet Engineering Task Force (IETF) Request for Comments (RFC) 1131, 1989.
   http://tools.ietf.org/html/rfc1131.

#### References

#### [8] D. Oran.

OSI IS-IS Intra-domain Routing Protocol.

Internet Engineering Task Force (IETF) Request for Comments (RFC) 1142, 1990. http://tools.ietf.org/html/rfc1142.

#### [9] W. Stallings.

Chapter 13: Routing in switched data networks.

In Data and Computer Communications. Pearson, 9th edition, 2010.

#### [10] D. Thaler and C. Hopps.

Multipath issues in unicast and multicast next-hop selection.

Internet Engineering Task Force (IETF) Request for Comments (RFC) 2991, 2000. http://tools.ietf.org/html/rfc2991.