Distance vector protocol

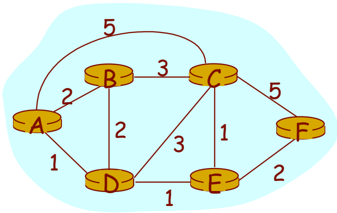
Irene Finocchi

finocchi@di.uniroma1.it

Routing

Routing protocol

Goal: determine "good" path (sequence of routers) thru network from source to dest.



Graph abstraction for routing algorithms:

- graph nodes are routers
- graph edges are "physical" links
  - link cost: delay, $cost, congestion level

"Good" path:
- minimum cost path
- other def's possible

Classification of routing algorithms

View: global or local

- Global: info about entire network (routers, links) [link state]
- Local: partial knowledge of remote parts of network [distance vector]

Centralized or decentralized

- one node maintains view, and distributes routes to other nodes
- all nodes maintain view

Static or dynamic?

Static:
- infrequent route changes
- infrequent view update; static link costs (e.g. up/down)

Dynamic:
- frequent periodic route changes
- frequent view update; dynamic link costs (e.g. delay)

Distance vector routing algorithm

Distributed, asynchronous implementation of the algorithm by Bellman & Ford

- Distributed: each node communicates *only* with directly-attached neighbors
- Asynchronous: nodes need *not* exchange info or iterate in lock step (synchronized)!
- Iterative:
  - continues until no nodes exchange info
  - *self-terminating*: no "signal" to stop
- Decentralized, local, dynamic

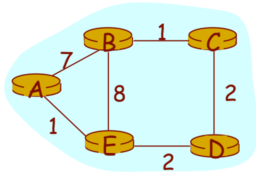
Distance table data structure

- Each node has its own distance table
- One row for each possible destination
- One column for each directly-attached neighbor of the node (outgoing links)

Example: at node S, for destination T via neighbor X:

$$D^S(T,X) = \text{distance } \textit{from } S \textit{ to } T, \textit{ via } X \text{ as next hop}$$
$$= w(S,X) + \min_Y D^X(T,Y)$$

Distance table: an example



$$D^E(C,D) = w(E,D) + \min_Y \{D^D(C,Y)\}$$
$$= 2+2 = 4$$

$$D^E(A,D) = w(E,D) + \min_Y \{D^D(A,Y)\}$$
$$= 2+3 = 5$$ loop!

$$D^E(A,B) = w(E,B) + \min_Y \{D^B(A,Y)\}$$
$$= 8+6 = 14$$ loop!

neighbors

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destinations

Distance table gives routing table

neighbors

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destinations

	Outgoing link to use, cost
A	A, 1
B	D, 5
C	D, 4
D	D, 4

destinations

Distance table → Routing table

Distance vector routing: an overview

Iterative, asynchronous

Each local iteration caused by:

- local link cost change
- message from neighbor v: a shortest path with source v has changed

Distributed

- each node notifies neighbors *only* when a shortest path to any destination changes
  - neighbors then notify their neighbors if necessary

Each node:

wait for (msg from neighbor / change in local link cost)

↓

update distance table

↓

if shortest path to any dest has changed, *notify* neighbors

Assumption

- For the time being, don't consider link cost changes: we'll remove this assumption later
- In the next slides we show:
  - How does the algorithm work
  - Why it stabilizes and produces in a finite amount of time the correct distances

Each node:

initialization

wait for (msg from neighbor)

update distance table

if shortest path to any dest has changed, *notify* neighbors

Distance vector algorithm: initialization

At node S:

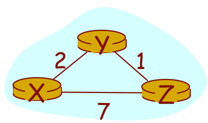
```
for all adjacent nodes y:
      D^S(.,y) = + ∞
      D^S(y,y) = w(S,y)

for all destinations t
send min_y D^S(t,y) to each neighbor
     /* y over all neighbors of S */
```

Distance vector algorithm: main loop

At node S:

```
loop
  wait (until S receives a message from a neighbor V)

  let m = ( V, T, C ) be the message received from V
  /* a path from V to T of cost C has been discovered */

  update: D^S (T,V) = w(S,V) + C

  if  min_Y  D^S (T,Y) changes, send its new value to all
                              the neighbors of S

forever
```

Distance vector algorithm: an example



D^X (dest.)	Y (neighbors)	Z (neighbors)
Y	2	∞
Z	∞	7

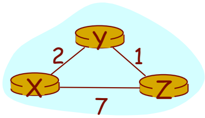
Distance table at node X just after the initialization

Msg destination	Msg/path source	Path destination	Cost
Y	X	Y	2
Z	X	Y	2
Y	X	Z	7
Z	X	Z	7

If msg destination = path destination, message is useless: Y is not a possible destination in its own distance table, and the info carried by the message cannot be used to update any entry in D^Y

Won't consider these messages any further

Distance vector algorithm: an example



D^X	neighbors Y	neighbors Z
dest. Y	2	∞
dest. Z	∞	7

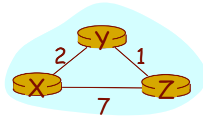
D^Y	neighbors X	neighbors Z
dest. X	2	∞
dest. Z	∞	1

D^Z	neighbors X	neighbors Y
dest. X	7	∞
dest. Y	∞	1

Msg destination	Msg/path source	Path destination	Cost
Z	X	Y	2
Y	X	Z	7
X	Y	Z	1
Z	Y	X	2
X	Z	Y	1
Y	Z	X	7

Messages generated during the initialization

Distance vector algorithm: an example



D^X	neighbors Y	neighbors Z
dest. Y	2	∞
dest. Z	∞	7

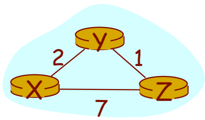
D^Y	neighbors X	neighbors Z
dest. X	2	∞
dest. Z	∞	1

D^Z	neighbors X	neighbors Y
dest. X	7	∞
dest. Y	9	1

Msg destination	Msg/path source	Path destination	Cost
Z	X	Y	2
Y	X	Z	7
X	Y	Z	1
Z	Y	X	2
X	Z	Y	1
Y	Z	X	7

$w(Z,X) + C = 7 + 2 = 9 \Rightarrow D^Z(Y,X) = 9$

Distance vector algorithm: an example



D^X	neighbors Y	neighbors Z
dest. Y	2	∞
dest. Z	∞	7

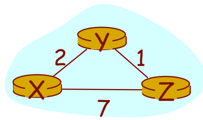
D^Y	neighbors X	neighbors Z
dest. X	2	∞
dest. Z	9	1

D^Z	neighbors X	neighbors Y
dest. X	7	∞
dest. Y	9	1

Msg destination	Msg/path source	Path destination	Cost
Y	X	Z	7
X	Y	Z	1
Z	Y	X	2
X	Z	Y	1
Y	Z	X	7

$w(Y,X) + C = 2 + 7 = 9 \Rightarrow D^Y(Z,X) = 9$

Distance vector algorithm: an example



D^X	neighbors Y	neighbors Z
dest. Y	2	∞
dest. Z	3	7

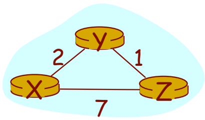
D^Y	neighbors X	neighbors Z
dest. X	2	∞
dest. Z	9	1

D^Z	neighbors X	neighbors Y
dest. X	7	∞
dest. Y	9	1

Msg destination	Msg/path source	Path destination	Cost
X	Y	Z	1
Z	Y	X	2
X	Z	Y	1
Y	Z	X	7
Y	X	Z	3

$w(X,Y) + C = 2 + 1 = 3 \Rightarrow D^X(Z,Y) = 3$

Distance vector algorithm: an example



D^X (dest. \ neighbors)	Y	Z
Y	2	∞
Z	3	7

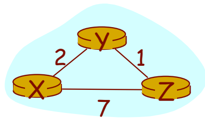
D^Y (dest. \ neighbors)	X	Z
X	2	∞
Z	9	1

D^Z (dest. \ neighbors)	X	Y
X	7	3
Y	9	1

Msg destination	Msg/path source	Path destination	Cost
Z	Y	X	2
X	Z	Y	1
Y	Z	X	7
Y	X	Z	3
Y	Z	X	3

$w(Z,Y) + C = 1 + 2 = 3 \Rightarrow D^Z(X,Y) = 3$

Distance vector algorithm: an example



D^X (dest. \ neighbors)	Y	Z
Y	2	8
Z	3	7

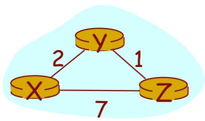
D^Y (dest. \ neighbors)	X	Z
X	2	∞
Z	9	1

D^Z (dest. \ neighbors)	X	Y
X	7	3
Y	9	1

Msg destination	Msg/path source	Path destination	Cost
X	Z	Y	1
Y	Z	X	7
Y	X	Z	3
Y	Z	X	3

$w(X,Z) + C = 7 + 1 = 8 \Rightarrow D^X(Y,Z) = 8$

Distance vector algorithm: an example



D^X (dest. \ neighbors)	Y	Z
Y	2	8
Z	3	7

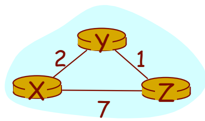
D^Y (dest. \ neighbors)	X	Z
X	2	8
Z	9	1

D^Z (dest. \ neighbors)	X	Y
X	7	3
Y	9	1

Msg destination	Msg/path source	Path destination	Cost
Y	Z	X	7
Y	X	Z	3
Y	Z	X	3

$w(Y,Z) + C = 1 + 7 = 8 \Rightarrow D^Y(X,Z) = 8$

Distance vector algorithm: an example



D^X (dest. \ neighbors)	Y	Z
Y	2	8
Z	3	7

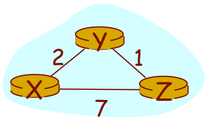
D^Y (dest. \ neighbors)	X	Z
X	2	8
Z	5	1

D^Z (dest. \ neighbors)	X	Y
X	7	3
Y	9	1

Msg destination	Msg/path source	Path destination	Cost
Y	X	Z	3
Y	Z	X	3

$w(Y,X) + C = 2 + 3 = 5 \Rightarrow D^Y(Z,X) = 5$

Distance vector algorithm: an example



neighbors

D^X	Y	Z
dest. Y	2	8
dest. Z	3	7

neighbors

D^Y	X	Z
dest. X	2	4
dest. Z	5	1

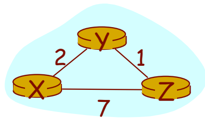
neighbors

D^Z	X	Y
dest. X	7	3
dest. Y	9	1

Msg destination	Msg/path source	Path destination	Cost
Y	Z	X	3

w(Y,Z) + C = 1 + 3 = 4 ⇨ D^Y (X,Z) = 4

Distance vector algorithm: an example



neighbors

D^X	Y	Z
dest. Y	2	8
dest. Z	3	7

neighbors

D^Y	X	Z
dest. X	2	4
dest. Z	5	1

neighbors

D^Z	X	Y
dest. X	7	3
dest. Y	9	1

Msg destination	Msg/path source	Path destination	Cost

No more messages remain: the algorithm has found a stable configuration

Correctness (1/2)

Does the algorithm stabilize and produce (in a finite amount of time) the correct distances?

```
let m = ( V, T, C ) be the message received from V
/* a path from V to T of cost C has been discovered */
update: DS(T,V) = w(S,V) + C
if  min  DS(T,Y) changes, send messages
     Y
```

This is just a relaxation!

$\min_Y D^S(T,Y) = D_{ST}$ $C = D_{VT}$

The update is equivalent to the relaxation $D_{ST} = w(S,V) + D_{VT}$

Correctness (2/2)

We previously proved the following:

If: 1) Distance estimate D_{xy} corresponds to the length of an existing path from x to y

2) Bellman's conditions $D_{xz} \leq w(x,y) + D_{zy}$ locally satisfied for each $(x,y) \in E$

Then $D_{xy} = d_{xy}$ for every $x,y \in V$

The algorithm always uses existing arcs ⇨ 1) OK

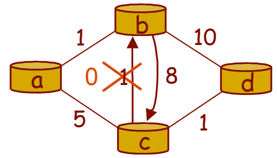
If at some point D_{zy} decreases, this is notified with a message to x, that restores the condition if necessary (and sends in turn messages to its own neighbors) ⇨ 2) OK

Dealing with link cost changes

At node S:

```
loop
  wait (until S receives a message from a neighbor V
        or the cost of a link (S,U) changes )

  if w(S,U) changes by δ
     /* change cost to all dest's via neighbor U by δ */
     /* δ may be positive or negative */
     for all destinations T: D^S(T,U) = D^S(T,U) + δ

  else if m = ( V, T, C ) is the message received from V
     /* a path from V to T of cost C was discovered */
     update: D^S(T,V) = w(S,V) + C

  for all destinations T:
       if  min  D^S(T,Y) changes, send its new value to
            Y
                                     all the neighbors of S
forever
```

Decreasing the cost of a link: an example



D^b (dest. \ neighbors)	a	c	d
a	1	10	13
c	6	8	11
d	7	9	10

D^c (dest. \ neighbors)	a	b	d
a	5	1	4
b	6	0	3
d	11	7	1

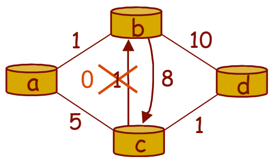
Msg dest.	Msg source	Path dest.	Cost
b	c	a	1
d	c	a	1
a	c	b	0
d	c	b	0

D^a (dest. \ neighbors)	b	c
b	1	6
c	7	5
d	8	6

D^d (dest. \ neighbors)	b	c
a	11	3
b	10	2
c	16	1

In D^C, decrease all entries in column **b** by **1**

Decreasing the cost of a link: an example



D^b (dest. \ neighbors)	a	c	d
a	1	9	13
c	6	8	11
d	7	9	10

D^c (dest. \ neighbors)	a	b	d
a	5	1	4
b	6	0	3
d	11	7	1

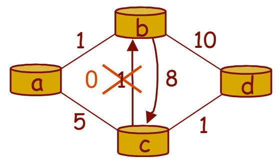
Msg dest.	Msg source	Path dest.	Cost
b	c	a	1
d	c	a	1
a	c	b	0
d	c	b	0

D^a (dest. \ neighbors)	b	c
b	1	6
c	7	5
d	8	6

D^d (dest. \ neighbors)	b	c
a	11	3
b	10	2
c	16	1

$w(b,c) + C = 8 + 1 = 9 \Rightarrow D^b(a,c) = 9$

Decreasing the cost of a link: an example



D^b (dest. \ neighbors)	a	c	d
a	1	9	13
c	6	8	11
d	7	9	10

D^c (dest. \ neighbors)	a	b	d
a	5	1	4
b	6	0	3
d	11	7	1

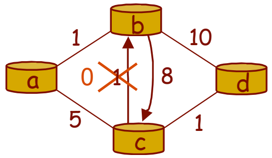
Msg dest.	Msg source	Path dest.	Cost
d	c	a	1
a	c	b	0
d	c	b	0
b	d	a	2
c	d	a	2

D^a (dest. \ neighbors)	b	c
b	1	6
c	7	5
d	8	6

D^d (dest. \ neighbors)	b	c
a	11	2
b	10	2
c	16	1

$w(d,c) + C = 1 + 1 = 2 \Rightarrow D^d(a,c) = 2$

Decreasing the cost of a link: an example



neighbors

D^b (dest.)	a	c	d
a	1	9	13
c	6	8	11
d	7	9	10

neighbors

D^c (dest.)	a	b	d
a	5	1	4
b	6	0	3
d	11	7	1

Msg dest.	Msg source	Path dest.	Cost
a	c	b	0
d	c	b	0
b	d	a	2
c	d	a	2

neighbors

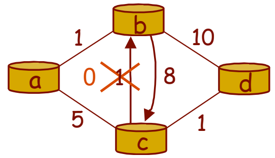
D^a (dest.)	b	c
b	1	5
c	7	5
d	8	6

neighbors

D^d (dest.)	b	c
a	11	2
b	10	2
c	16	1

$w(a,c) + C = 5 + 0 = 5 \Rightarrow D^a(b,c) = 5$

Decreasing the cost of a link: an example



neighbors

D^b (dest.)	a	c	d
a	1	9	13
c	6	8	11
d	7	9	10

neighbors

D^c (dest.)	a	b	d
a	5	1	4
b	6	0	3
d	11	7	1

Msg dest.	Msg source	Path dest.	Cost
d	c	b	0
b	d	a	2
c	d	a	2
c	d	b	1

neighbors

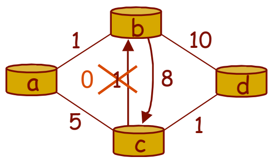
D^a (dest.)	b	c
b	1	5
c	7	5
d	8	6

neighbors

D^d (dest.)	b	c
a	11	2
b	10	1
c	16	1

$w(d,c) + C = 1 + 0 = 1 \Rightarrow D^d(b,c) = 1$

Decreasing the cost of a link: an example



neighbors

D^b (dest.)	a	c	d
a	1	9	12
c	6	8	11
d	7	9	10

neighbors

D^c (dest.)	a	b	d
a	5	1	4
b	6	0	3
d	11	7	1

Msg dest.	Msg source	Path dest.	Cost
b	d	a	2
c	d	a	2
c	d	b	1

neighbors

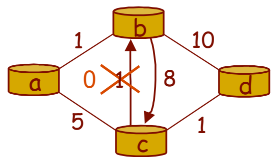
D^a (dest.)	b	c
b	1	5
c	7	5
d	8	6

neighbors

D^d (dest.)	b	c
a	11	2
b	10	1
c	16	1

$w(b,d) + C = 10 + 2 = 12 \Rightarrow D^b(a,d) = 12$

Decreasing the cost of a link: an example



neighbors

D^b (dest.)	a	c	d
a	1	9	12
c	6	8	11
d	7	9	10

neighbors

D^c (dest.)	a	b	d
a	5	1	3
b	6	0	3
d	11	7	1

Msg dest.	Msg source	Path dest.	Cost
c	d	a	2
c	d	b	1

neighbors

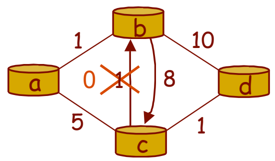
D^a (dest.)	b	c
b	1	5
c	7	5
d	8	6

neighbors

D^d (dest.)	b	c
a	11	2
b	10	1
c	16	1

$w(c,d) + C = 1 + 2 = 3 \Rightarrow D^c(a,d) = 3$

Decreasing the cost of a link: an example



neighbors

D^b	a	c	d
dest. a	1	9	12
c	6	8	11
d	7	9	10

neighbors

D^c	a	b	d
dest. a	5	1	3
b	6	0	2
d	11	7	1

Msg dest.	Msg source	Path dest.	Cost
c	d	b	1

neighbors

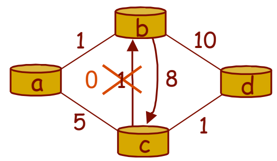
D^a	b	c
dest. b	1	5
c	7	5
d	8	6

neighbors

D^d	b	c
dest. a	11	2
b	10	1
c	16	1

$w(c,d) + C = 1 + 1 = 2 \Rightarrow D^c(b,d) = 2$

Decreasing the cost of a link: an example



neighbors

D^b	a	c	d
dest. a	1	9	12
c	6	8	11
d	7	9	10

neighbors

D^c	a	b	d
dest. a	5	1	3
b	6	0	2
d	11	7	1

Msg dest.	Msg source	Path dest.	Cost

No more messages remain: the algorithm has found a stable configuration

neighbors

D^a	b	c
dest. b	1	5
c	7	5
d	8	6

neighbors

D^d	b	c
dest. a	11	2
b	10	1
c	16	1

Decreasing links: "good news travel fast"

Why does the algorithm stabilize?

- (S,X) = arc that we decreased by the amount δ
- T = any destination
- Assume for simplicity that all cycles have cost >0
  - ⇒ Shortest path between any pair of nodes must be simple
- Consider the path $S \rightarrow \underbrace{X \rightsquigarrow T}_{}$
  - shortest path from X to T ⇒ simple
  - ⇒ cannot contain (S,X)

⇒ The cost of $S \rightarrow X \rightsquigarrow T$ decreases by exactly δ

Decreasing links: "good news travel fast"

The cost of $\Pi_{ST} = S \rightarrow X \rightsquigarrow T$ decreases by exactly δ

1) If Π_{ST} was shortest, the cost of the shortest path from S to T changes

2) If Π_{ST} was not shortest, it may become preferable to the old shortest path from S to T

In any case, if Π_{ST} is shortest after decreasing (S,X), it must be simple (no cycle has cost 0), it is a path really existing in G and we know exactly its cost

⇒ The messages sent to the neighbors of S contain correct info

With similar arguments: the neighbors of S will correctly propagate this information backwards

Increasing links: “bad news travel slow”

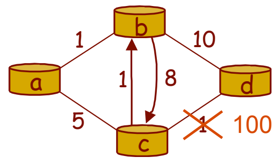
- (S,X) = arc that we increased by the amount δ
- T = any destination
- If $S \rightarrow X \rightsquigarrow T$ was shortest before the update, it may no longer be the shortest path after increasing (S,X)
- Replacement path = minimum in row T of D^S
- The replacement path may not be simple and may contain (S,X)

$$S \rightarrow Y \rightsquigarrow S \rightarrow X \rightsquigarrow T$$

In this case we should increase its cost by δ, but the algorithm doesn’t know when this is necessary

⇨ The messages sent to the neighbors of S may contain **wrong info** and the algorithm may not stabilize!

Increasing links: sending wrong info



D^b (dest. \ neighbors)	a	c	d
a	1	10	13
c	6	8	11
d	7	9	10

D^c (dest. \ neighbors)	a	b	d
a	5	2	4
b	6	1	3
d	11	8	1

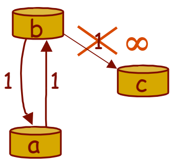
In $\mathbf{D^c}$ and $\mathbf{D^d}$, increase all entries in columns **d** and **c** by **99**

In $\mathbf{D^c}$, $\mathbf{D^c}$(d,b)=8 is the new minimum in row d of D^c, but there is no path in the graph from c to d of cost 8!

D^a (dest. \ neighbors)	b	c
b	1	6
c	7	5
d	8	6

D^d (dest. \ neighbors)	b	c
a	11	3
b	10	2
c	16	1

“Count to infinity” problem



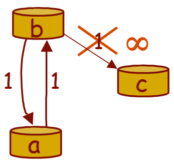
D^a (dest. \ neighbors)	b
b	1
c	2

D^b (dest. \ neighbors)	a	c
a	1	∞
c	3	∞

D^c (dest. \ neighbors)	
a	
b	

Msg destination	Msg/path source	Path destination	Cost
a	b	c	3

“Count to infinity” problem



D^a (dest. \ neighbors)	b
b	1
c	4

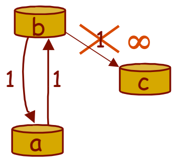
D^b (dest. \ neighbors)	a	c
a	1	∞
c	3	∞

D^c (dest. \ neighbors)	
a	
b	

Msg destination	Msg/path source	Path destination	Cost
a	b	c	3
b	a	c	4

$w(a,b) + C = 1 + 3 = 4$ ⇨ $D^a(c,b) = 4$

“Count to infinity” problem



D^a	neighbors b
dest. b	1
dest. c	4

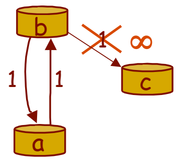
D^b	neighbors a	neighbors c
dest. a	1	∞
dest. c	5	∞

D^c	neighbors
dest. a	
dest. b	

Msg destination	Msg/path source	Path destination	Cost
a	b	c	3
b	a	c	4
a	b	c	5

$w(b,a) + C = 1 + 4 = 5 \Rightarrow D^b(c,a) = 5$

“Count to infinity” problem



D^a	neighbors b
dest. b	1
dest. c	6

D^b	neighbors a	neighbors c
dest. a	1	∞
dest. c	5	∞

D^c	neighbors
dest. a	
dest. b	

Msg destination	Msg/path source	Path destination	Cost
a	b	c	3
b	a	c	4
a	b	c	5
b	a	c	6

• • •

$w(a,b) + C = 1 + 5 = 6 \Rightarrow D^a(c,b) = 6$

How to make things work

- Different solutions proposed to solve the problem (poisoned reverse, …
- None of them really general
- To solve the problem completely we should keep information about the entire path to a destination (path vector protocols)
- But messages in that case are much bigger

Hierarchical Routing

Our routing study thus far - idealization
all routers identical
network “flat”
… *not* true in practice

scale: with 50 million destinations:

- can’t store all dest’s in routing tables!
- routing table exchange would swamp links!

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

Hierarchical Routing

- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

gateway routers

- special routers in AS
- run intra-AS routing protocol with all other routers in AS
- *also* responsible for routing to destinations outside AS
  - run *inter-AS routing* protocol with other gateway routers