
Transport BW Allocation, and Review of Network Routing

11/2/2009

Recap: Questions

$$\begin{array}{ll} \max & \sum_{f \in F} U_f(x_f) \\ \text{subject to} & \sum_{f: f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\ \text{over} & x \geq 0 \end{array}$$

- **Forward engineering:** which allocation objective?
- **Reverse engineering:** what are the objectives of TCP/Reno, TCP/Vegas?

Objective	Allocation (x1, x2, x3)		
TCP/Reno	0.26	0.74	0.74
TCP/Vegas	1/3	2/3	2/3
Max throughput	0	1	1
Max-min	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
Max sum log(x)	1/3	2/3	2/3
Max sum of $-1/(\text{RTT}^2 x)$	0.26	0.74	0.74

Recall: Resource Allocation Framework

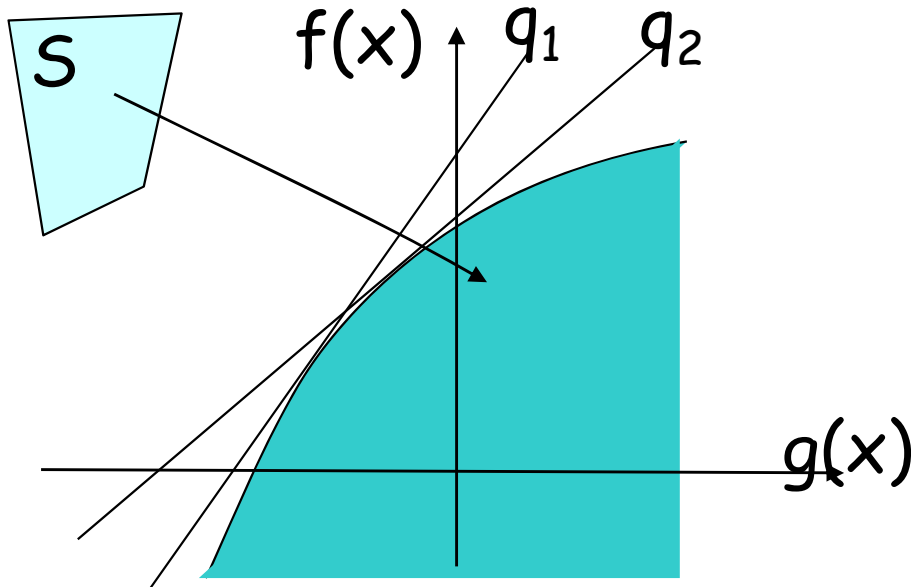
- Maximize aggregate utility, subject to capacity constraints

$$\begin{array}{ll} \text{max} & \sum_{f \in F} U_f(x_f) \\ \text{subject to} & \sum_{f: f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\ \text{over} & x \geq 0 \end{array}$$

A One-Slide Summary of Optimization Theory

$$\begin{array}{ll} \max & f(x) \\ \text{subject to} & g(x) \leq 0 \\ \text{over} & x \in S \end{array}$$

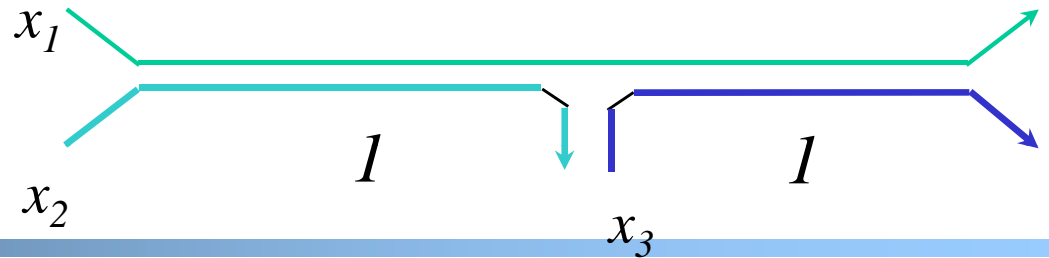
$f(x)$ concave
 $g(x)$ linear
 S is a convex set



$$D(q) = \max_{x \in S} (f(x) - qg(x))$$

- $D(q)$ is called the dual;
 $q (\succ= 0)$ are called prices in economics
- $D(q)$ provides an upper bound on obj.
- Then according to optimization theory: when $D(q)$ achieves minimum over all $q (\succ= 0)$, then the optimization objective is achieved.

Decomposition



- Assume each link l has non-negative congestion signal q_l , consider the dual $D(q)$

$$\begin{aligned} D(q) &= \max_{x_f \geq 0} \left(\sum_f U_f(x_f) - \sum_l q_l \left(\sum_{f: \text{uses } l} x_f - c_l \right) \right) \\ &= \max_{x_f \geq 0} \sum_f \left(U_f(x_f) - x_f \sum_{l: f \text{ uses } l} q_l \right) + \sum_l q_l c_l \\ &= \sum_f \max_{x_f \geq 0} \left(U_f(x_f) - x_f \sum_{l: f \text{ uses } l} q_l \right) + \sum_l q_l c_l \end{aligned}$$

Distributed Optimization: User Problem

- Given price signal per unit rate p_f (=sum of q_l along the path) flow f chooses rate x_f to maximize:

$$\begin{array}{ll} \max_{x_f} & U_f(x_f) - x_f p_f \\ \text{over} & x_f \geq 0 \end{array}$$

- Using the price signals, the optimization problem of each user is independent of each other

Distributed Optimization: User Problem

$$\begin{array}{ll} \max_{x_f} & U_f(x_f) - x_f p_f \\ \text{over} & x_f \geq 0 \end{array}$$

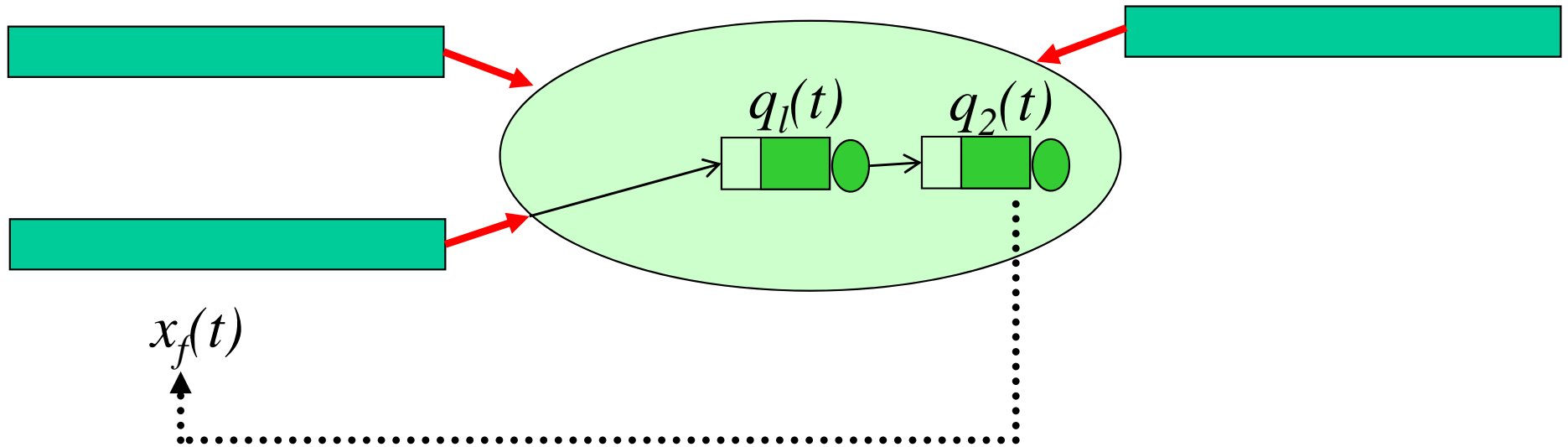
How should flow f adjust x_f locally?

$$\Delta x_f \propto U'_f(x_f) - p_f$$

At equilibrium (i.e., at optimal), x_f satisfies:

$$U'_f(x_f) - p_f = 0$$

Interpreting Congestion Measure



$$p_f = \sum_{f \text{ uses } l} q_l$$

$$\Delta x_f \propto U'_f(x_f) - p_f$$

Distributed Optimization: Network Problem

$$D(q) = \sum_f \max_{x_f \geq 0} \left(U_f(x_f) - x_f \sum_{l: f \text{ uses } l} q_l \right) + \sum_l q_l c_l$$

The network (i.e., link l) adjusts the link signals q_l (assume after all flows have picked their optimal rates given congestion signal)

$$\min_{q \geq 0} \tilde{D}(q) = \sum_l q_l (c_l - \sum_{f: f \text{ uses } l} x_f)$$

Distributed Optimization: Network Problem

$$\min_q D(q) = \sum_l q_l (c_l - \sum_{f: f \text{ uses } l} x_f)$$

how should link l adjust q_l locally?

$$\Delta q_l \propto - \frac{\partial D(q)}{q_l}$$

$$\frac{\partial}{\partial q_l} D(q) = c_l - \sum_{f: \text{uses } l} x_f$$

$$\Delta q_l \propto \sum_{f: \text{uses } l} x_f - c_l$$

Decomposition

□ SYSTEM(U):

$$\begin{array}{ll} \max & \sum_{f \in F} U_f(x_f) \\ \text{subject to} & \sum_{f: f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\ \text{over} & x \geq 0 \end{array}$$

□ USER_f:

$$\begin{array}{ll} \max_{x_f} & U_f(x_f) - x_f p_f \\ \text{over} & x_f \geq 0 \end{array}$$

□ NETWORK:


$$\min_{q \geq 0} \tilde{D}(q) = \sum_l q_l (c_l - \sum_{f: f \text{ uses } l} x_f)$$

Outline

- Bandwidth allocation framework
 - framework
 - Nash Bargaining Solution (NBS)
 - distributed computation
 - *TCP/Reno, TCP/Vegas revisited*

TCP/Reno Dynamics $\Delta x_f \propto U'_f(x_f) - p_f$

$$\Delta x = \frac{x^2}{2} \left(\frac{2}{RTT^2 x^2} - p \right)$$

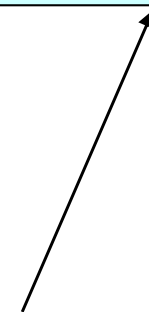
$$U'_f(x_f) - p_f$$


$$\Rightarrow U'_f(x_f) = \left(\frac{\sqrt{2}}{x_f RTT} \right)^2 \quad \Rightarrow U_f(x_f) = -\frac{2}{RTT^2 x_f}$$

TCP/Vegas Dynamics

$$\Delta x_f \propto U'_f(x_f) - p_f$$

$$\Delta x = \frac{x}{RTT^2} \left(\frac{\alpha}{x} - (RTT - RTT_{\min}) \right)$$

$$U'_f(x_f) - p_f$$


$$\Rightarrow U'_f(x_f) = \frac{\alpha}{x}$$

$$\Rightarrow U_f(x_f) = \alpha \log(x_f)$$

Summary: TCP/Vegas and TCP/Reno

□ Pricing signal is queueing delay T_{queueing}

$$x_f = \frac{\alpha}{T_{\text{queueing}}}$$

$$U'_f(x_f) = T_{\text{queueing}}$$

$$\Rightarrow U'_f(x_f) = \frac{\alpha}{x_f}$$

$$\Rightarrow U_f(x_f) = \alpha \log(x_f)$$

□ Pricing signal is loss rate p

$$x_f = \frac{\alpha}{RTT \sqrt{p}}$$

$$U'_f(x_f) = p$$

$$\Rightarrow U'_f(x_f) = \left(\frac{\alpha}{x_f RTT} \right)^2$$

$$\Rightarrow U_f(x_f) = -\frac{\alpha'}{RTT^2 x_f}$$

Summary: Global Network Bandwidth Allocation

- **Understand** protocols,
 - e.g., Reno, Vegas

- **Design** new e2e congestion control and router queue management protocols
 - e.g., FAST, XCP

$$\begin{array}{ll} \max & \sum_{f \in F} U_f(x_f) \\ \text{subject to} & Ax \leq C \\ \text{over} & x \geq 0 \end{array}$$

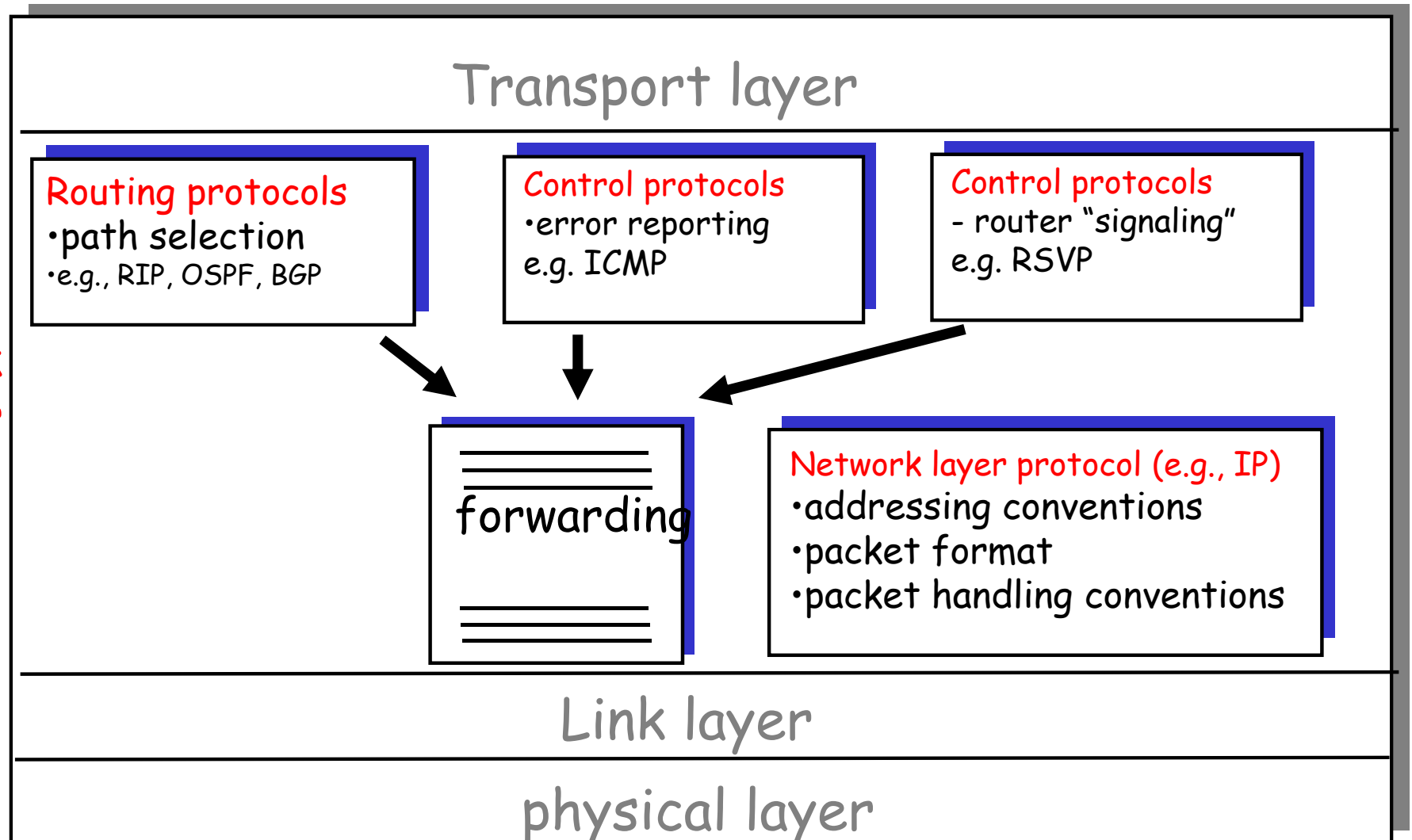
- **Pointer:** congestion control through game theory and economics
 - <http://www.statslab.cam.ac.uk/~frank/pf/>

Exam 1

- Covers lectures 1-16 and the preceding part of today's lecture
- A one-page "cheat" sheet

Recap: Network Layer: Protocols

Network layer functions:



Recap: Routing Design Space

- Robustness
- Optimality
- Simplicity

- Routing has a large design space
 - who decides routing?
 - how many paths from source s to destination d ?
 - will routing adapt to network traffic demand?
 - ...
- We focus mostly on network-based, single-path, static routing

Example: Cisco Proprietary Recommendation on Link Cost

□ Link metric:

$$\text{EIGRP}_{\text{metric}} = \{k1 \times \text{BW} + [(k2 \times \text{BW}) / (256 - \text{load})] + k3 \times \text{delay}\} \times \{k5 / (\text{reliability} + k4)\}$$

By default, $k1=k3=1$ and $k2=k4=k5=0$. The default composite metric for EIGRP, adjusted for scaling factors, is as follows:

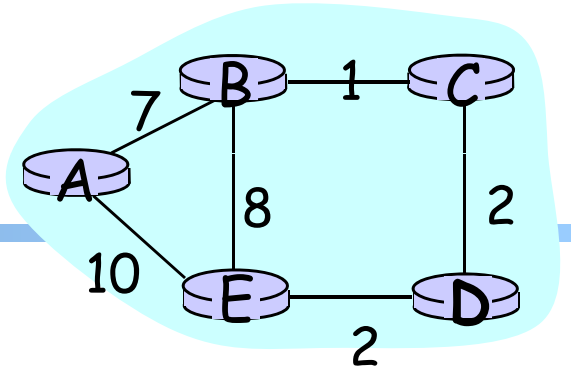
$$\text{EIGRP}_{\text{metric}} = 256 \times \{ [10^7 / \text{BW}_{\text{min}}] + [\text{sum_of_delays}] \}$$

EIGRP : Enhanced Interior Gateway Routing Protocol

EIGRP Link Cost

- ❑ BW_{\min} is in kbps and the sum of delays are in 10s of microseconds.
- ❑ **Example**
- ❑ The bandwidth and delay for an Ethernet interface are 10 Mbps and 1ms, respectively.
- ❑ The calculated EIGRP BW metric is as follows:
 - $256 \times 10^7 / BW = 256 \times 10^7 / 10,000$
 - $= 256 \times 10000$
 - $= 2560000$

Recap: Synchronous Bellman-Ford (SBF)

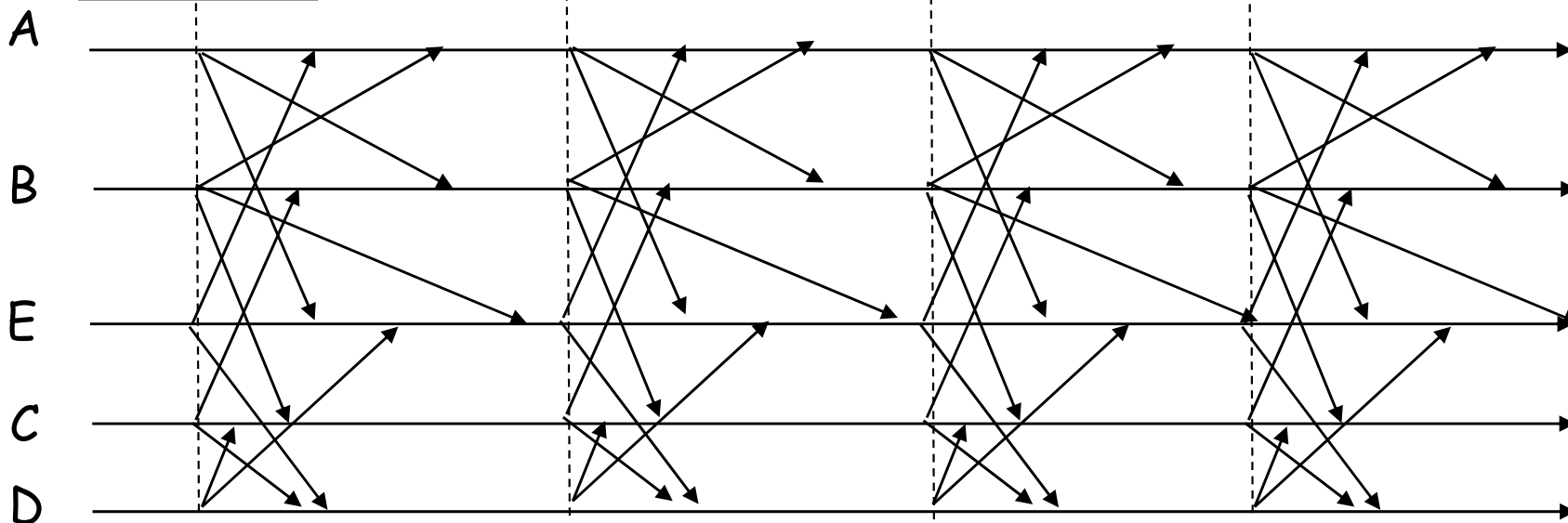


□ Nodes update in rounds:

- there is a global clock;
- at the beginning of each round, each node sends its estimate to all of its neighbors;
- at the end of the round, updates its estimation

$d(0) ?$

$$d_i(h+1) = \min_{j \in N(i)} (d_{ij} + d_j(h))$$



$$d_i(h+1) = \min_{j \in N(i)} (d_{ij} + d_j(h))$$

Recap: Properties of SBF

□ Monotonicity:

○ if $d(t) \geq d'(t) \Rightarrow d(t+1) \geq d'(t+1)$

- i.e., each node has a higher estimate in one scenario (d) than in another scenario (d'), then each node has a higher estimate in d than in d' after one round of synchronous update.

□ SBF/ ∞ computes the shortest path costs

□ Bellman equation has a unique solution

□ SBF/ -1 also computes the shortest path costs

Recap: Asynchronous Bellman-Ford (ABF)

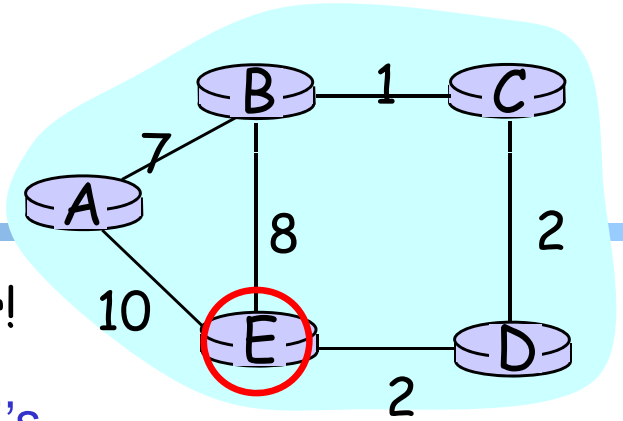
- No notion of global iterations
 - each node updates at its own pace
- Asynchronously each node i computes

$$d_i = \min_{j \in N(i)} (d_{ij} + d_j^i)$$

using last received value d_j^i from neighbor j .

- Asynchronously node j sends its estimate to its neighbor i :
 - there is an upper bound on the delay of estimate packets (no worry for out of order)

Distance Table: Example



Below is just one step! The protocol repeats forever!

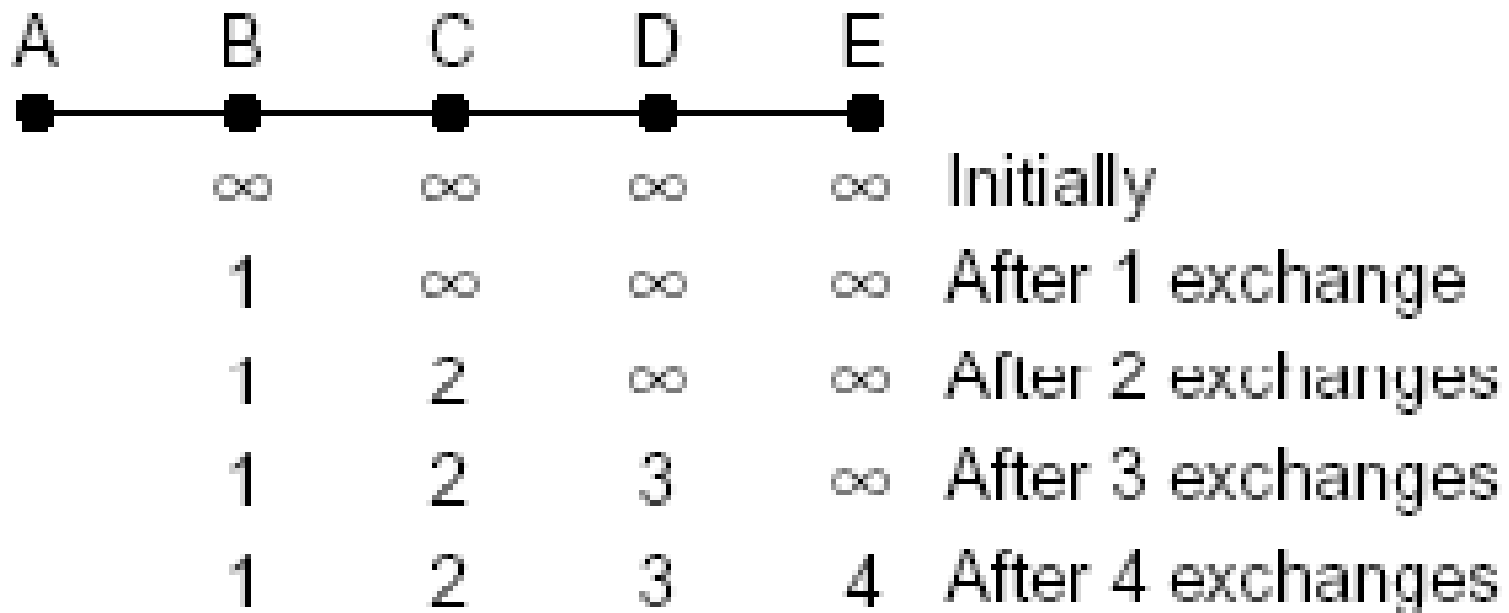
		distance tables from neighbors			computation			E's distance table	distance table E sends to its neighbors
$d_E()$		A	B	D	A	B	D		
destinations	A	0	7	∞	10	15	∞	A: 10	A: 10
	B	7	0	∞	17	8	∞	B: 8	B: 8
	C	∞	1	2	∞	9	4	D: 4	C: 4
	D	∞	∞	0	∞	∞	2	D: 2	D: 2
		10	8	2					E: 0

Recap: Asynchronous Bellman-Ford: Summary

- ❑ Distributed: each node communicates its routing table to its directly-attached neighbors
- ❑ Iterative: continues periodically or when link changes, e.g. detects a link failure
- ❑ Asynchronous: nodes need *not* exchange info/iterate in lock step!
- ❑ Convergence in finite steps, independent of initial condition if network is connected

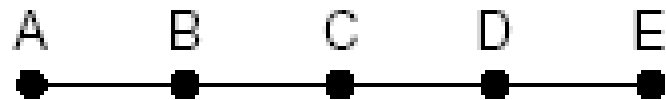
Properties of Distance-Vector Algorithms

- Good news propagate fast



Properties of Distance-Vector Algorithms

□ Bad news propagate slowly



A	B	C	D	E	
	1	2	3	4	Initially
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges
	7	8	7	8	After 6 exchanges
		⋮			
	∞	∞	∞	∞	

□ This is called the *counting-to-infinity* problem

Question: why does counting-to-infinity happen?

What is a Routing Loop?

- A routing loop is a global state (consisting of the nodes' local states) at a global moment (observed by an oracle) such that there exist nodes $A, B, C, \dots E$ such that A (locally) thinks B as down stream, B thinks C as down stream, ... E thinks A as down stream

Backup Slides

Network problem

- As if the network maximizes a logarithmic utility function, but with constants (w_f , $f \in F$) chosen by the users

$$\begin{array}{ll} \text{max} & \sum_{f \in F} w_f \log(x_f) \\ \text{subject to} & \sum_{f: f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\ \text{over} & x \geq 0 \end{array}$$

Three Optimization Problems

□ SYSTEM(U_f):

$$\begin{array}{ll} \max & \sum_{f \in F} U_f(x_f) \\ \text{subject to} & \sum_{f: f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\ \text{over} & x \geq 0 \end{array}$$

□ USER $_f$ (U_f ; p_f)

$$\begin{array}{ll} \max_{x_f} & U_f(x_f) - x_f p_f \\ \text{over} & x_f \geq 0 \end{array}$$

□ NETWORK(w_f)

Decomposition Theorem

- There exist vectors \mathbf{p} , \mathbf{w} and \mathbf{x} such that
 1. $w_f = p_f x_f$ for $f \in F$
 2. w_f solves $\text{USER}_f(U_f; p_f)$
 3. \mathbf{x} solves $\text{NETWORK}(\mathbf{w})$

- The vector \mathbf{x} then also solves $\text{SYSTEM}(U)$.

Another Decomposition

□ SYSTEM(U):

$$\begin{array}{ll} \max & \sum_{f \in F} U_f(x_f) \\ \text{subject to} & \sum_{f: f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\ \text{over} & x \geq 0 \end{array}$$

□ USER_f(U_f; p_f)

$$\begin{array}{ll} \max_{x_f} & U_f\left(\frac{w_f}{p_f}\right) - w_f \\ \text{over} & w_f \geq 0 \end{array}$$

□ NETWORK(w_f)

$$\begin{array}{ll} \max & \sum_{f \in F} w_f \log_f(x_f) \\ \text{subject to} & \sum_{f: f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\ \text{over} & x \geq 0 \end{array}$$

Decomposition Theorem

- There exist vectors p , w and x such that
 1. $w_f = p_f x_f$ for $f \in F$
 2. w_f solves $USER_f(U_f; p_f)$
 3. x solves $NETWORK(w)$

- The vector x then also solves $SYSTEM(U)$.

TCP/Reno: Equilibrium $U'_f(x_f) = p_f$

- Pricing signal is aggregated loss rate p

$$x_f = \frac{\alpha}{RTT \sqrt{p}}$$

$$\Rightarrow U'_f(x_f) = \left(\frac{\sqrt{2}}{x_f RTT} \right)^2$$

$$\Rightarrow U_f(x_f) = -\frac{2}{RTT^2 x_f}$$

TCP/Vegas: Equilibrium

$$U'_f(x_f) = p_f$$

□ Pricing signal is queueing delay T_{queueing}

$$x_f = \frac{\alpha}{T_{\text{queueing}}}$$

$$\Rightarrow U'_f(x_f) = \frac{\alpha}{x_f}$$

$$\Rightarrow U_f(x_f) = \alpha \log(x_f)$$

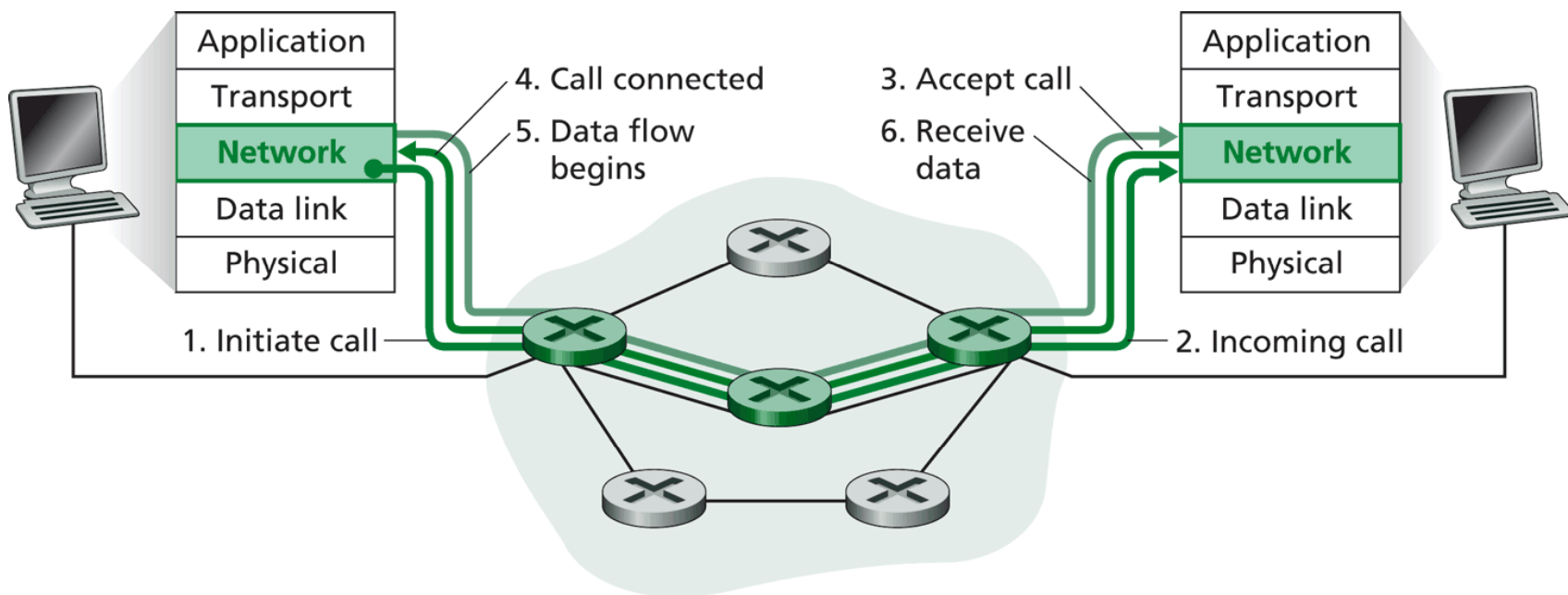
Summary: Network Bandwidth Allocation

$$\begin{array}{ll} \max & \sum_{f \in F} U_f(x_f) \\ \text{subject to} & \sum_{f: f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\ \text{over} & x \geq 0 \end{array}$$

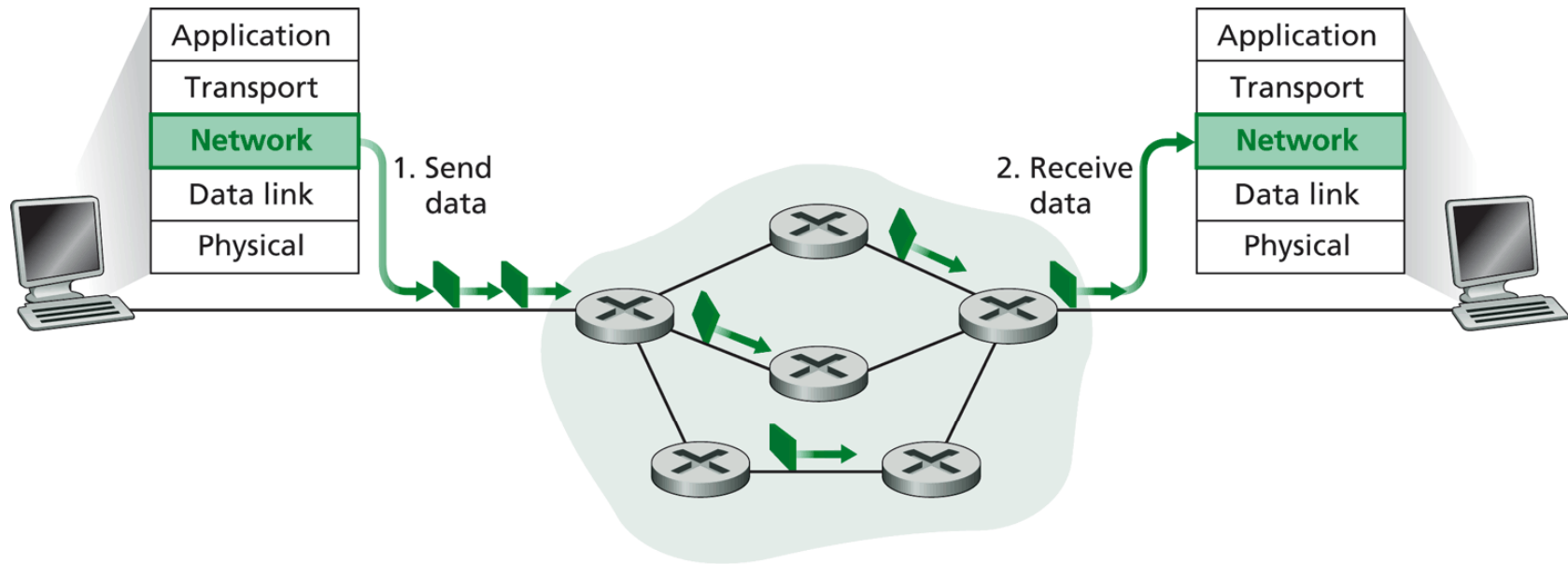
- Forward engineering:
 - how to determine objective functions: “first-principles” approach to derive utility functions
 - given objective, how to design effective alg (?)
- Reverse engineering: understand current protocols
- Discussion: problems of the framework

Using Virtual Circuit to Implement Network Services

- In order to provide some functionalities, a network may choose virtual circuit, e.g., Virtual Private Network (VPN)



Using Datagram to Implement the Most Basic Network Service

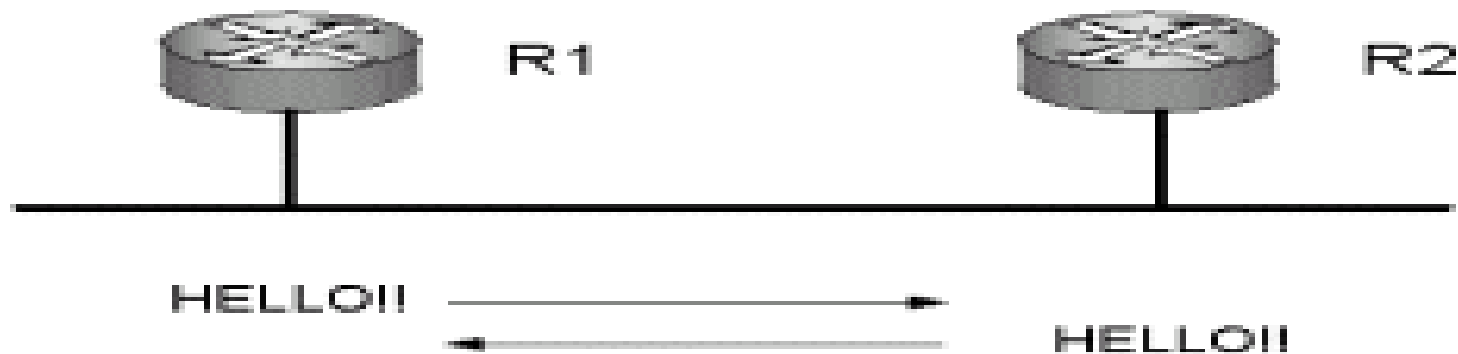


- ❑ A datagram network generally provides simple services: the forwarding of packets from src to dest.
- ❑ We will focus on datagram networks which provide best effort service
 - extensions to provide more services will be discussed in the multimedia networking part of the course

EIGRP Neighbor Discovery

I have a
neighbor called
R2!

I have a
neighbor called
R1!



- ❑ EIGRP routers actively establish relationships with their neighbors
- ❑ EIGRP routers establish adjacencies with neighbor routers by using small **hello** packets.
- ❑ The **Hello protocol** uses a multicast address of 224.0.0.10, and all routers periodically send hellos.

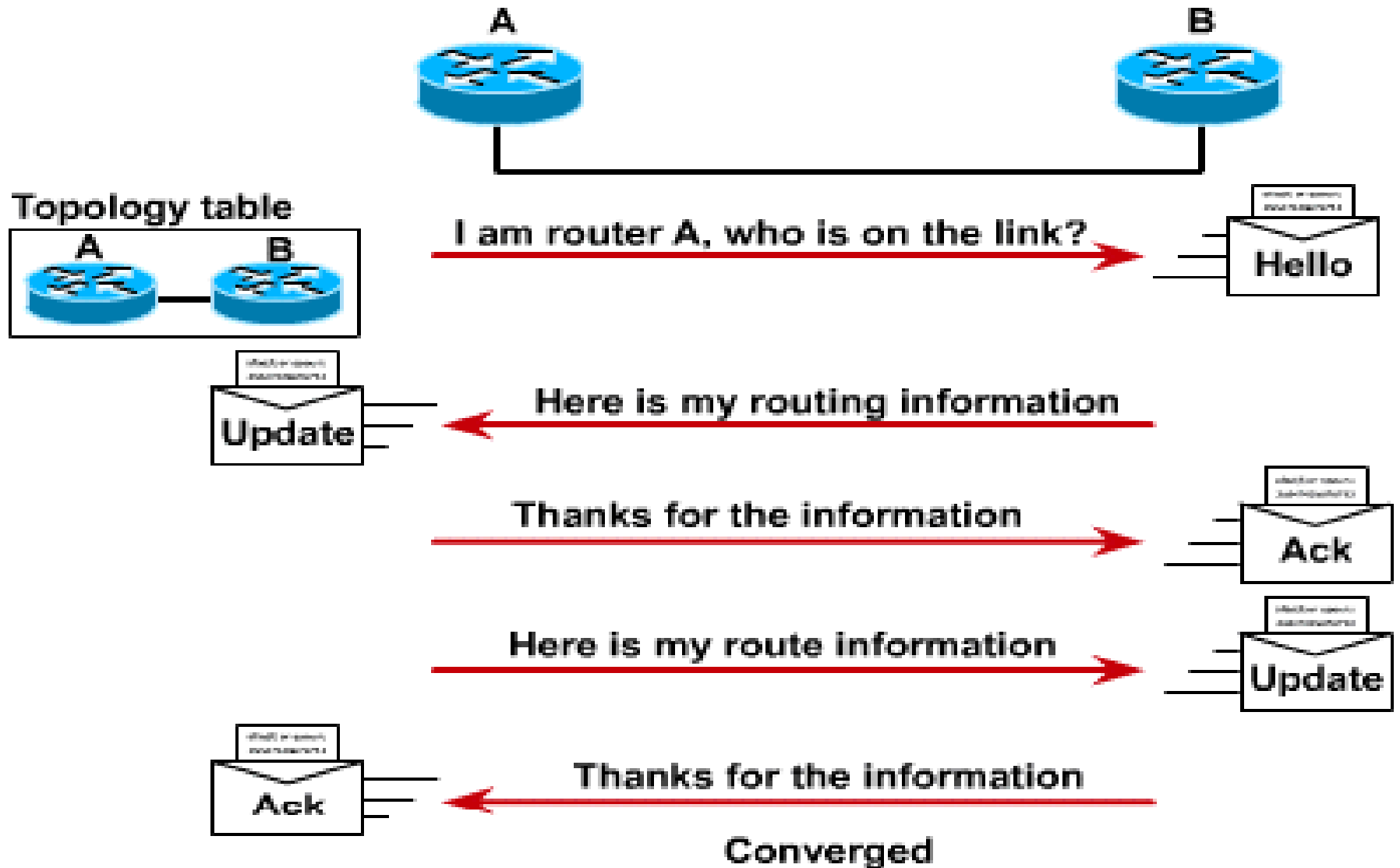
EIGRP Neighbor Discovery

- ❑ On hearing hellos, the router creates a table of its neighbors.
- ❑ The continued receipt of these packets maintains the neighbor table

By forming adjacencies, EIGRP routers do the following:

- ❑ Dynamically learn of new routes that join their network
- ❑ Identify routers that become either unreachable or inoperable
- ❑ Rediscover routers that had previously been unreachable

Neighbor Discovery - 3



Default Hello Intervals and Hold Time for EIGRP

Bandwidth	Example Link	Default Hello Interval	Default Hold Time
1.544 Mbps or less	Multipoint Frame Relay	60 seconds	180 seconds
Greater than 1.544 Mbps	T1, Ethernet	5 seconds	15 seconds