Midterm Review

15-441 Fall 2017

Profs Peter Steenkiste & **Justine Sherry**

Thanks to Scott Shenker, Sylvia Ratnasamay, Peter Steenkiste, and Srini Seshan for slides.

Carnegie Mellon University

sli.do time…

Midterm Facts

- On Tuesday, in class
- No cheatsheet allowed
- We will start right on time. Please arrive a few minutes before class to make sure you have every minute!

Material on the Midterm

- Everything up to this Tuesday
- Topics covered:
  - Protocols
  - Packet Switching
  - IP Addressing and Hierarchy
  - Routing
  - DNS
  - NATs
  - BGP
  - TCP
  - Circuits and Virtual Routing
  - The Web

Studying for the Midterm

- Materials you have: lecture slides, homeworks, textbook, practice problems which I am handing out today
- When I was a student I crawled the Internet for other textbooks and other universities' courses for extra practice problems before my exams.
  - I have done this for you instead — this is the jankiest guide ever but it is some of the best questions I found from other schools.

Dictionary

janky

jan·ky

/ˈjaNGkē/

adjective NORTH AMERICAN *informal*

of extremely poor or unreliable quality.
"the software is pretty janky"

Translations, word origin, and more definitions

*It's ugly, but these are the best practice questions I could find from UCSD, UW, Berkeley, CMU, and two textbooks you don't have.

These practice problems are not an assignment and you do not have to turn them in.

Today: Review

- As requested:
  - MPLS vs VLAN
  - Heirarchy/Prefixes
  - BGP
  - TCP
- Open Q&A if time…

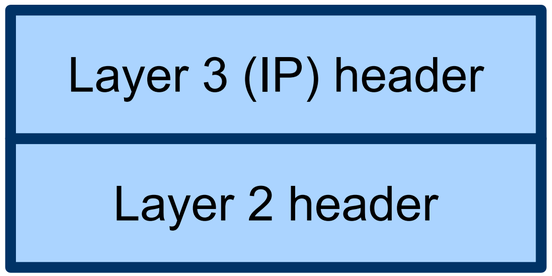
“Can we go over the practical differences between VLAN and MPLS?”

Quick comparison

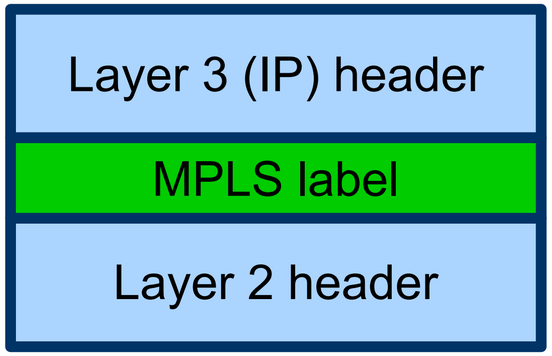
VLAN	MPLS
Link Layer	Network Layer
Creates a virtual network	Creates a virtual route/path
Motivation: Limit broadcast, Make management easier, provide security isolation	Motivation: QoS, Traffic Engineering, Faster Forwarding

MPLS

- Multi-Protocol Label Switching
- Bringing virtual circuit concept into IP
- Driven by multiple forces
  - QoS
  - Traffic engineering
  - High performance forwarding
  - VPN

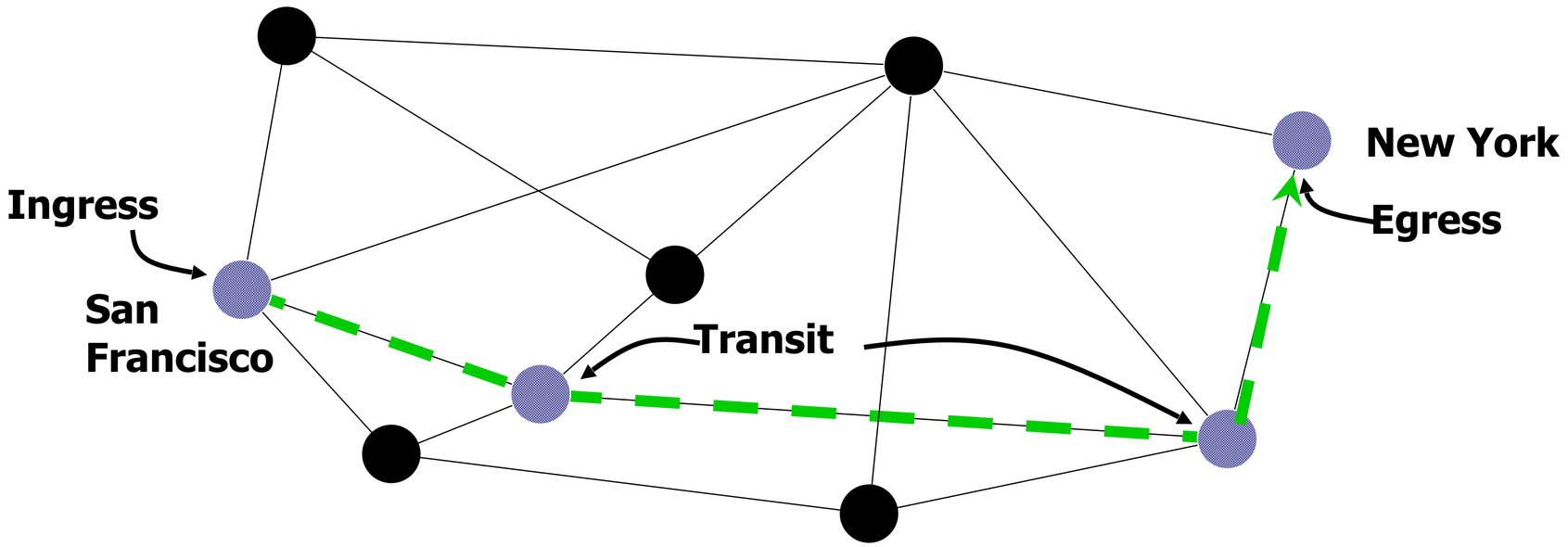


Some MPLS slides from H. Zhang



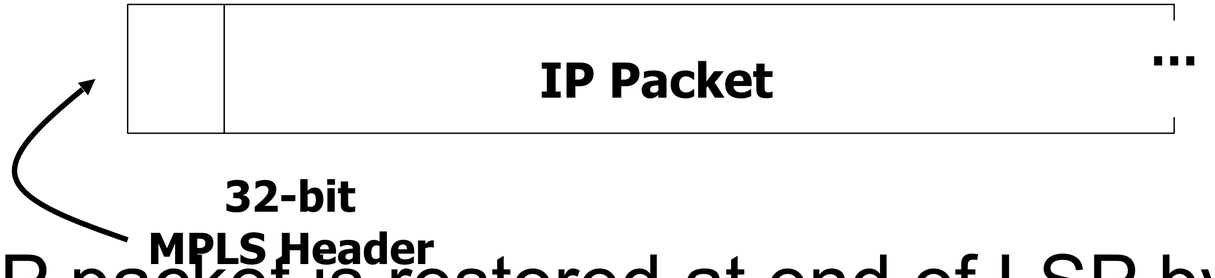
MPLS Vocabulary: LSR

- Label-switching router (LSR)
  - Performs LSP setup and MPLS packet forwarding
  - Label Edge Router (LER): LSP ingress or egress
  - Transit Router: swaps MPLS label, forwards packet



MPLS Header

- IP packet is encapsulated in MPLS header and sent down LSP



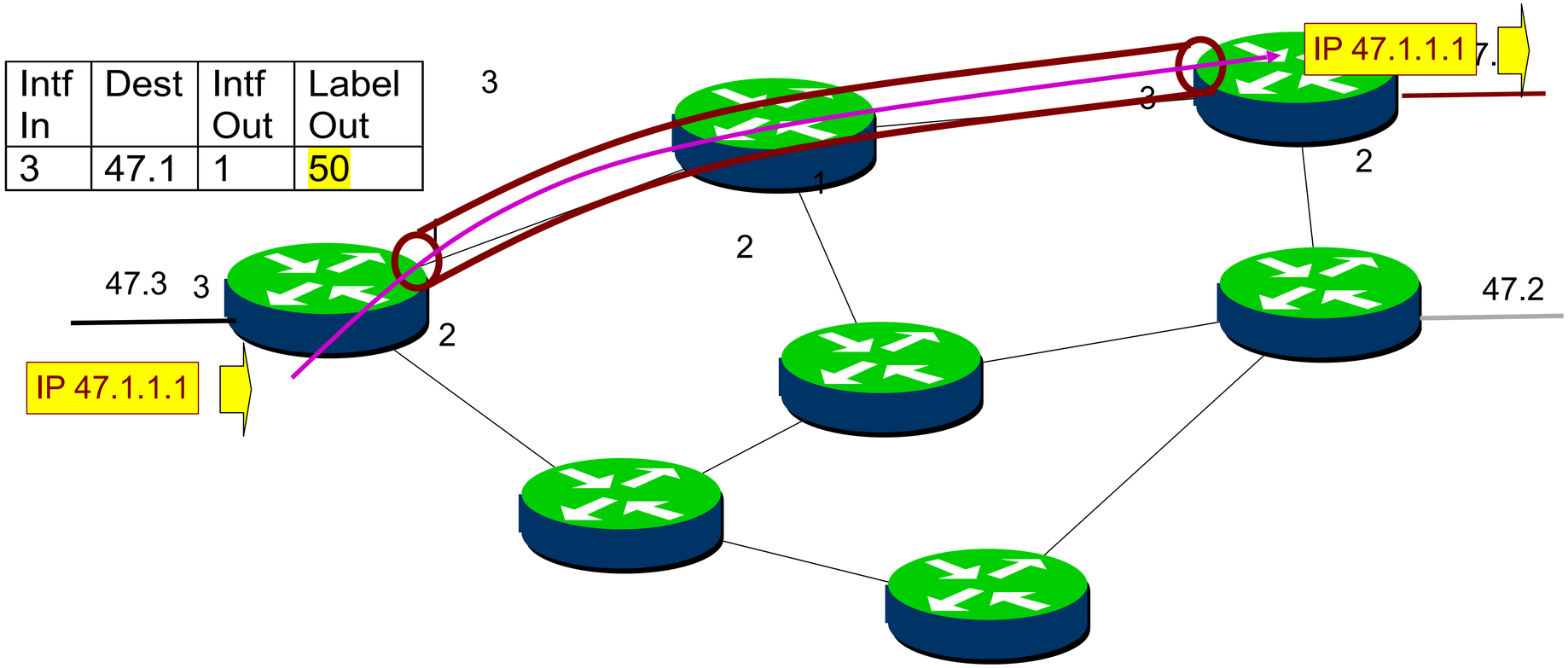
- IP packet is restored at end of LSP by egress router
  - TTL is adjusted, transit LSP routers count towards the TTL
  - MPLS is an optimization – does not affect IP semantics

Label Switched Path (LSP)

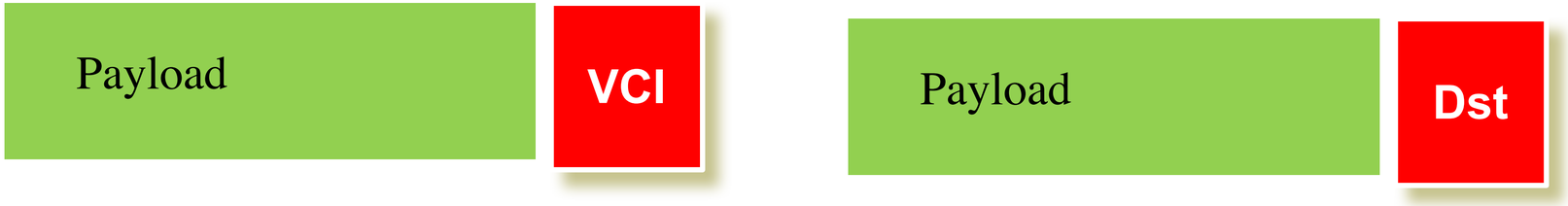
Intf In	Label In	Dest	Intf Out	Label Out
3	50	47.1	1	40

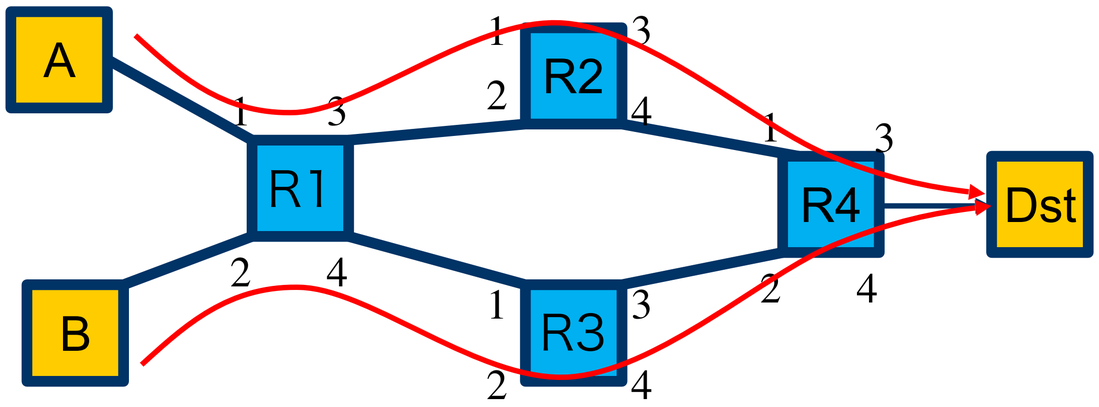
Intf In	Label In	Dest	Intf Out
3	40	47.1	1

Intf In	Dest	Intf Out	Label Out
3	47.1	1	50



VC versus Packets: Control over Path





R1 packet forwarding table:

Dst R2

R1 VC table:

VC 1 R2

VC 2 R3

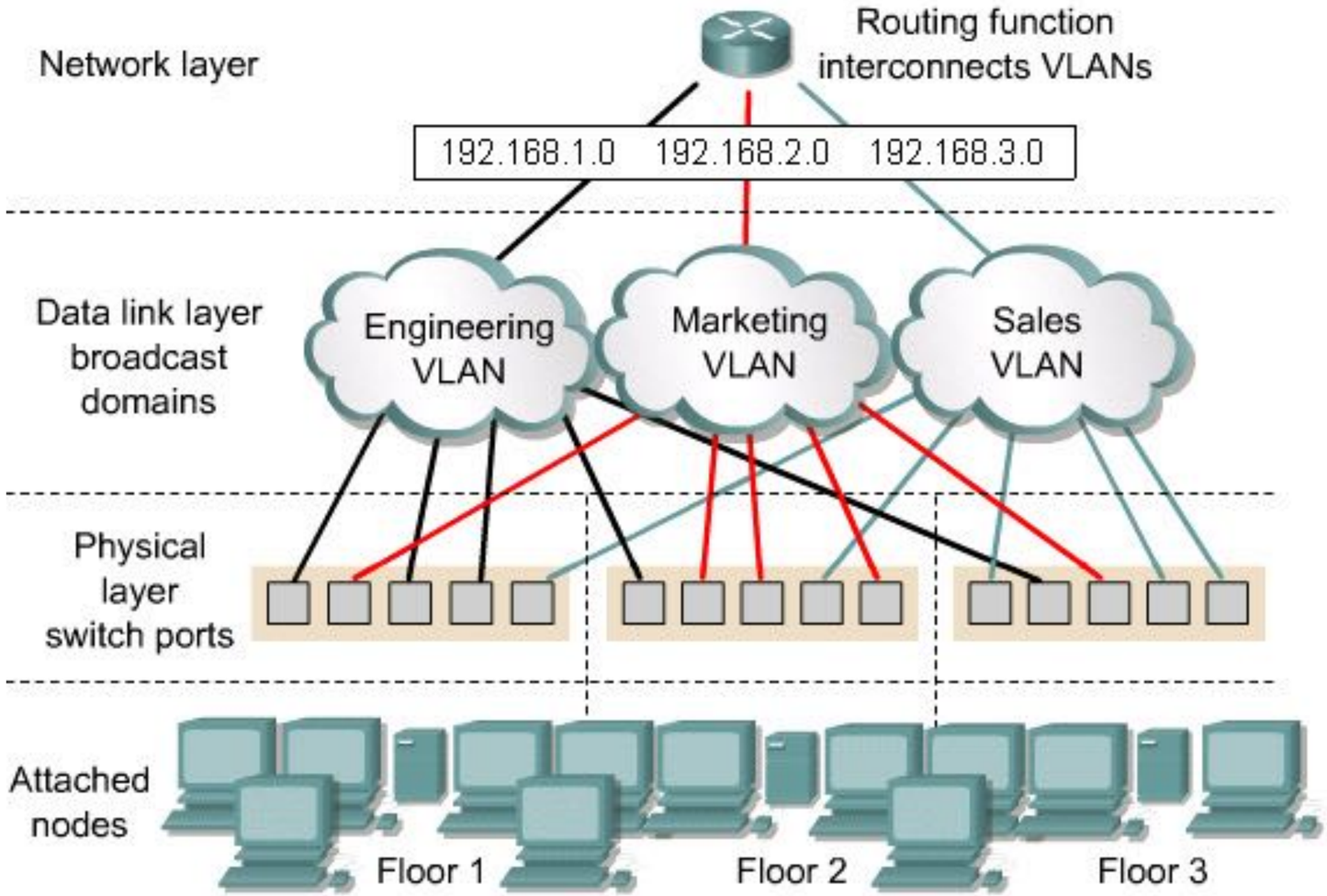
Different paths to same destination!

(useful for traffic engineering!)

Quick MPLS Recap

- Why does MPLS help with:
  - Faster router forwarding?
  - QoS?
  - Traffic Engineering?

VLAN Logical Topology



Example: 802.1Q Standard for VLANs over Ethernet

- A 32 bit VLAN header is inserted after the MAC addresses

- Header consists of
  - Tag Protocol Identifier (16b): single value that marks frame as a VLAN frame
  - Control bits (4b): mostly priority
  - VLAN Identifier (12b): identifies VLAN

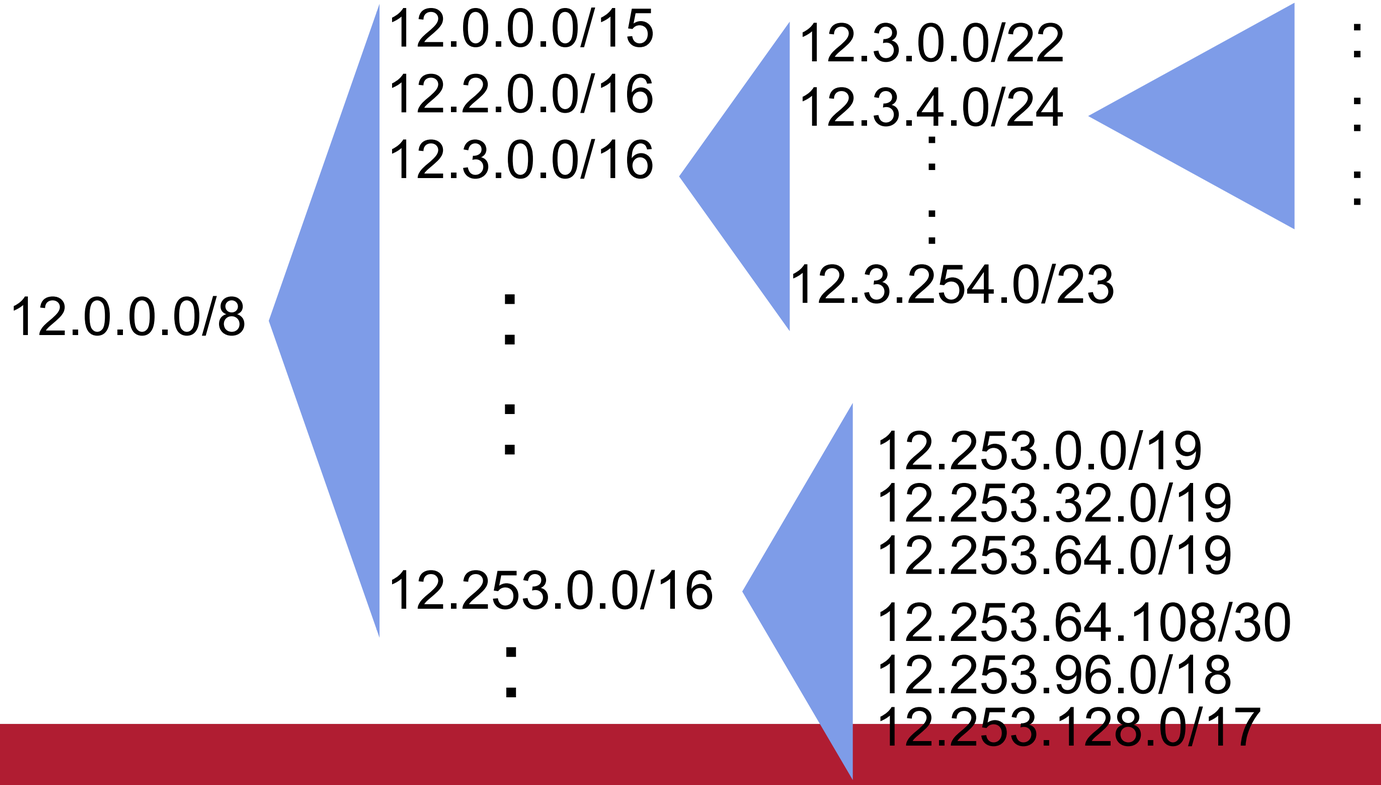
Quick VLAN Recap

- Why do VLANs help improve:
  - Performance? (Think about broadcasts)
  - Security?
  - Management?

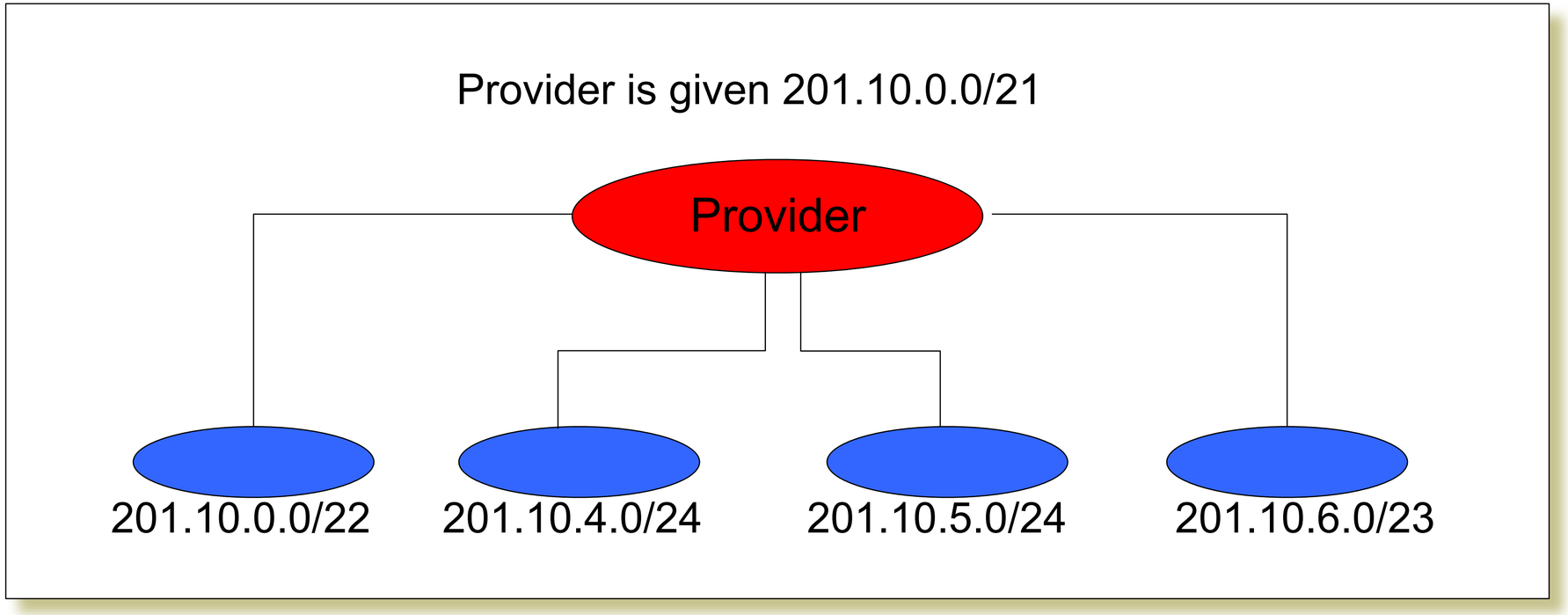
Why do we say that the prefix “grows” the closer you get to the packet destination?

CIDR: Addresses allocated in contiguous prefix chunks

Recursively break down chunks as get closer to host



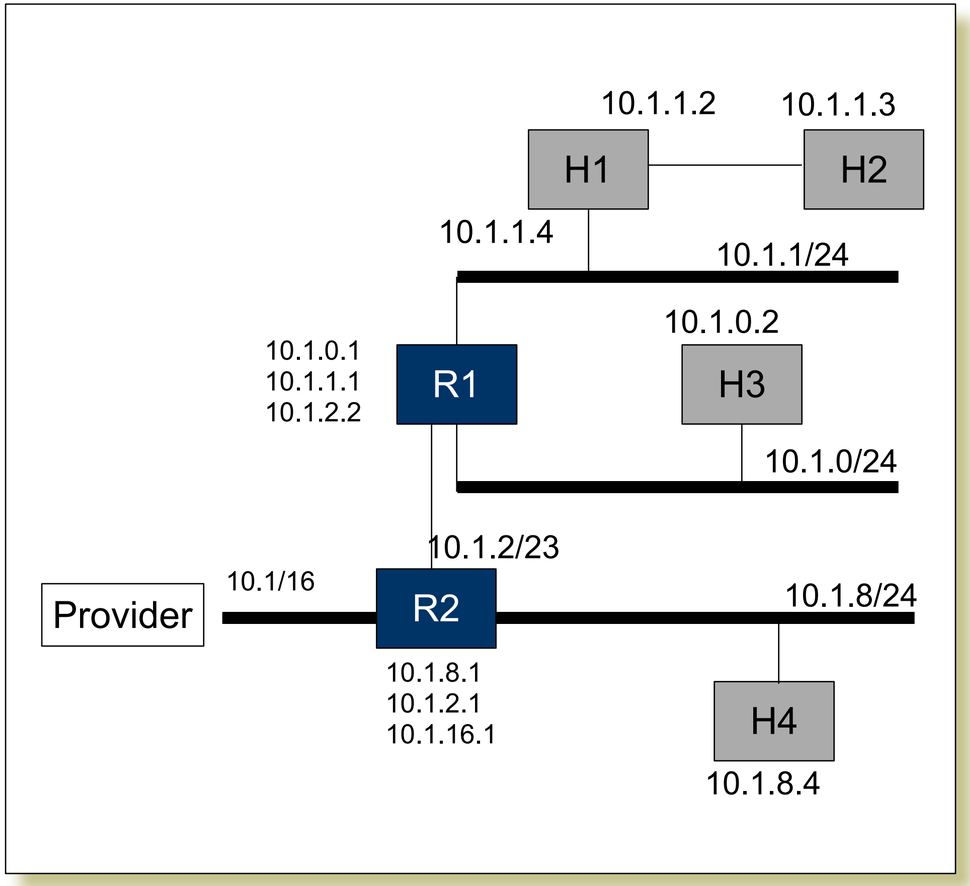
CIDR Illustration



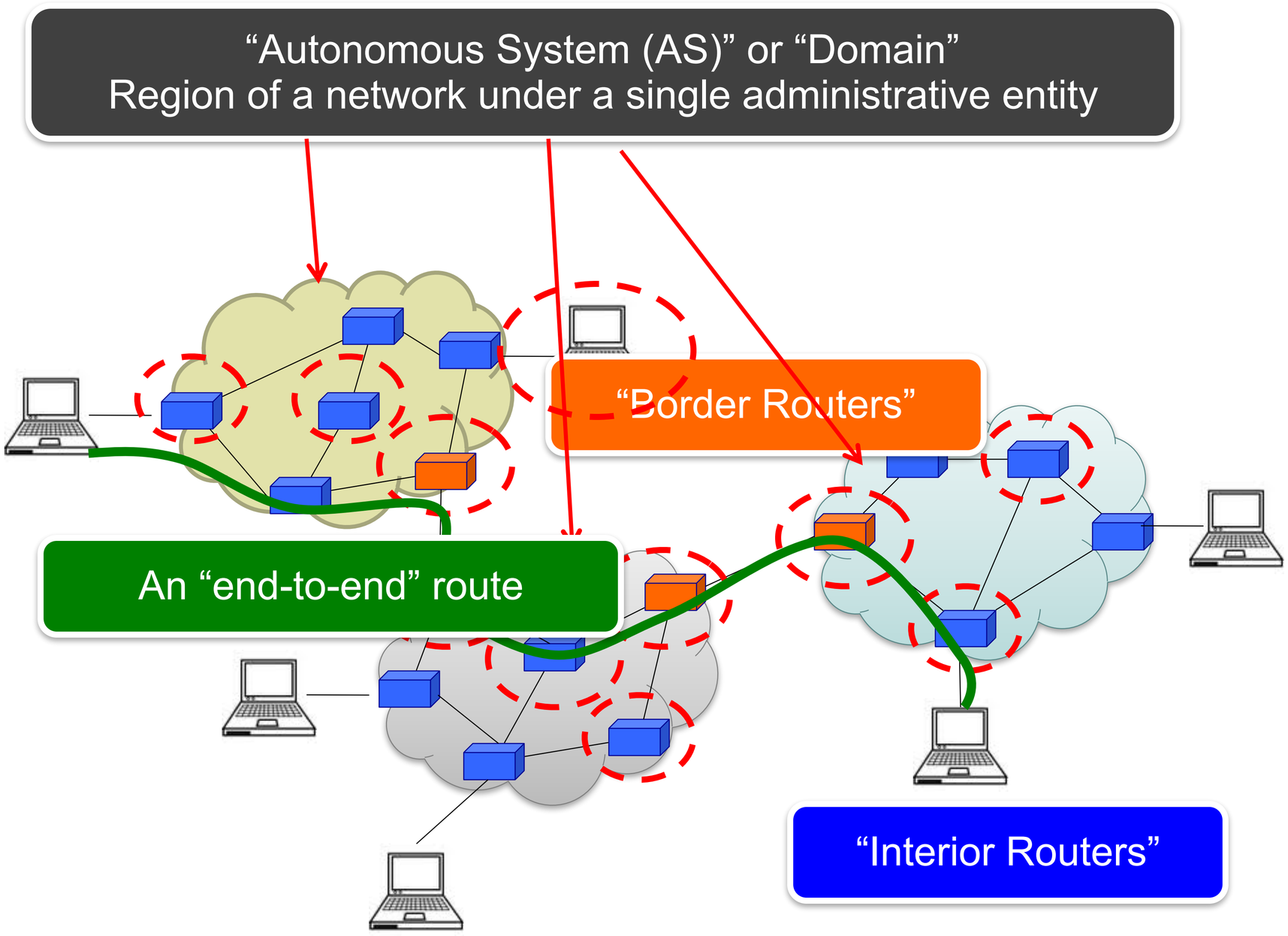
Term to know: “Route aggregation”

Routing to the Network

- Packet to 10.1.1.3 arrives
- Path is R2 – R1 – H1 – H2
- H1 serves as a router for the 10.1.1.2/31 network



“BGP is still a little confusing- can you recap how routing decisions are made?” (x3)


"Autonomous System (AS)" or "Domain"
Region of a network under a single administrative entity
"Border Routers"
An "end-to-end" route
"Interior Routers"

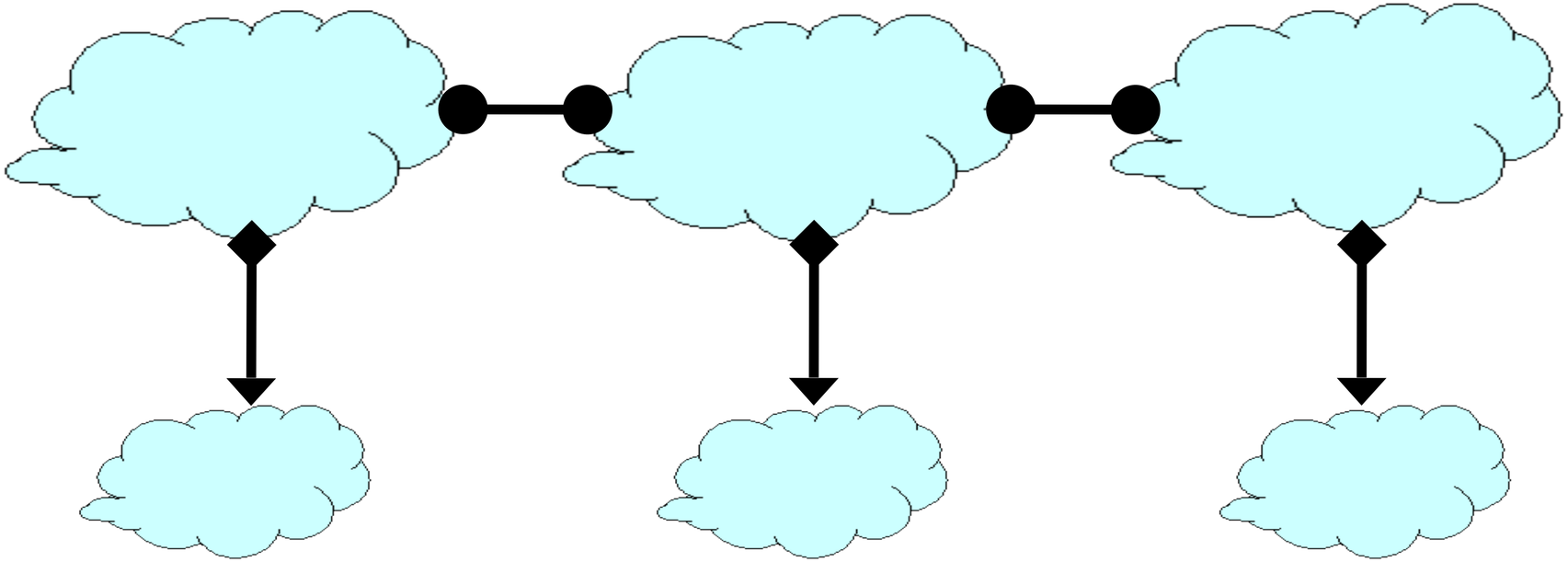
Autonomous Systems (AS)

- AS is a network under a single administrative control
  - currently over 30,000 ASes
  - Think AT&T, France Telecom, UCB, IBM, *etc.*
- ASes are sometimes called “domains”.
  - Hence, “interdomain routing”
- Each AS is assigned a unique identifier
  - 16 bit AS Number (ASN)

Topology and policy is shaped by the business relationships between ASes

- Three basic kinds of relationships between ASes
  - AS A can be AS B's *customer*
  - AS A can be AS B's *provider*
  - AS A can be AS B's *peer*
- Business implications
  - Customer pays provider
  - Peers don't pay each other
    - Exchange roughly equal traffic

Business Relationships



Relations between ASes

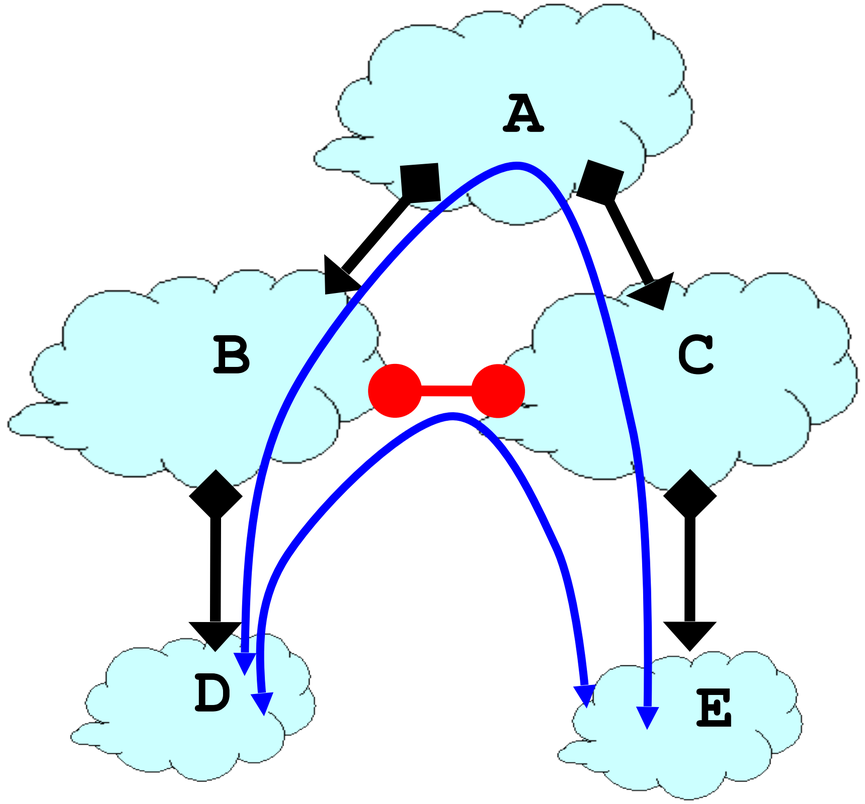
provider ◆——▶ **customer**

peer ●——● **peer**

Business Implications

- **Customers pay provider**
- **Peers don't pay each other**

Why peer?



E.g., D and E talk a lot

Peering saves B *and* C money

Relations between ASes

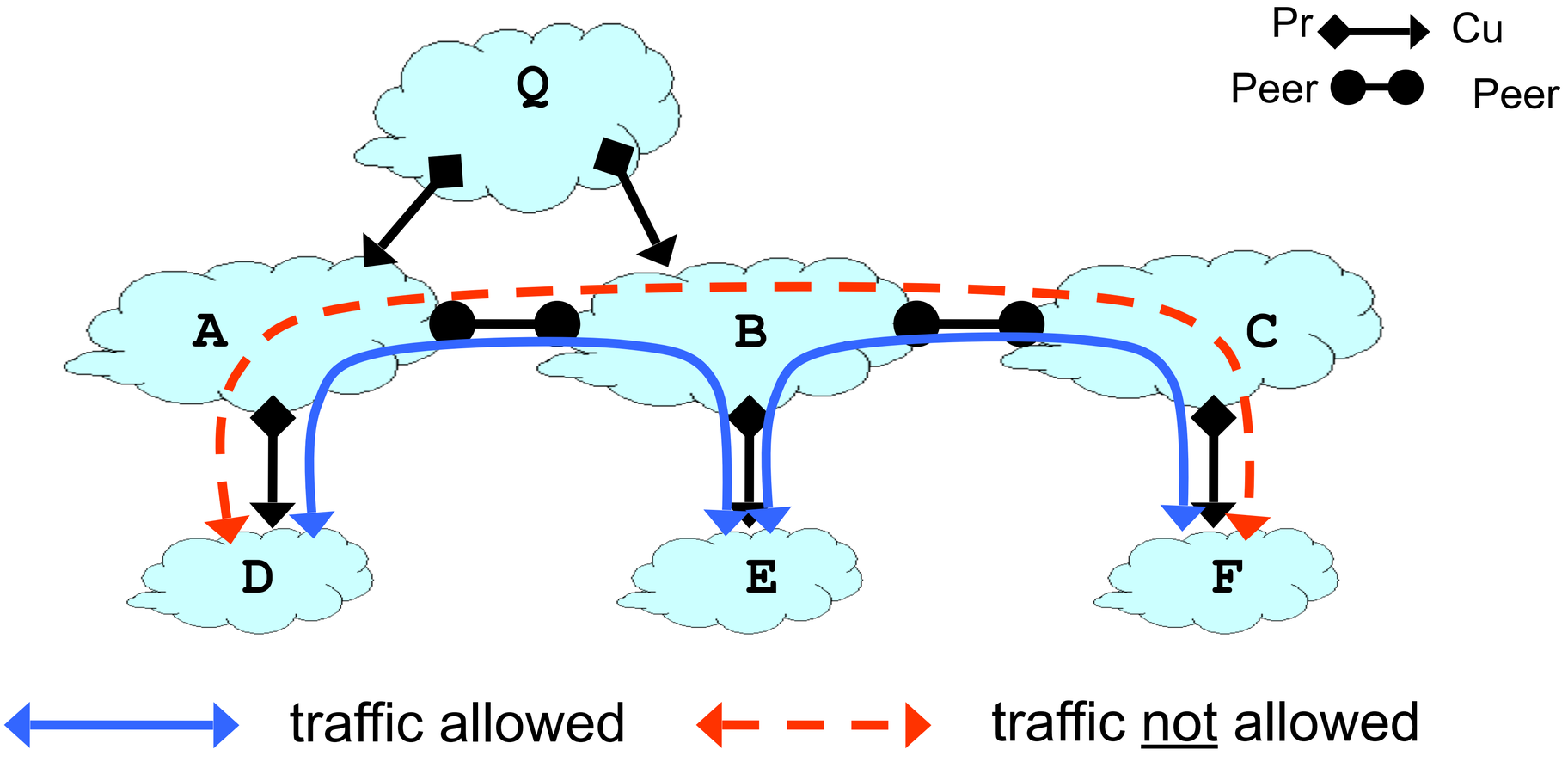
provider ◆——▶ **customer**

peer ●——● **peer**

Business Implications

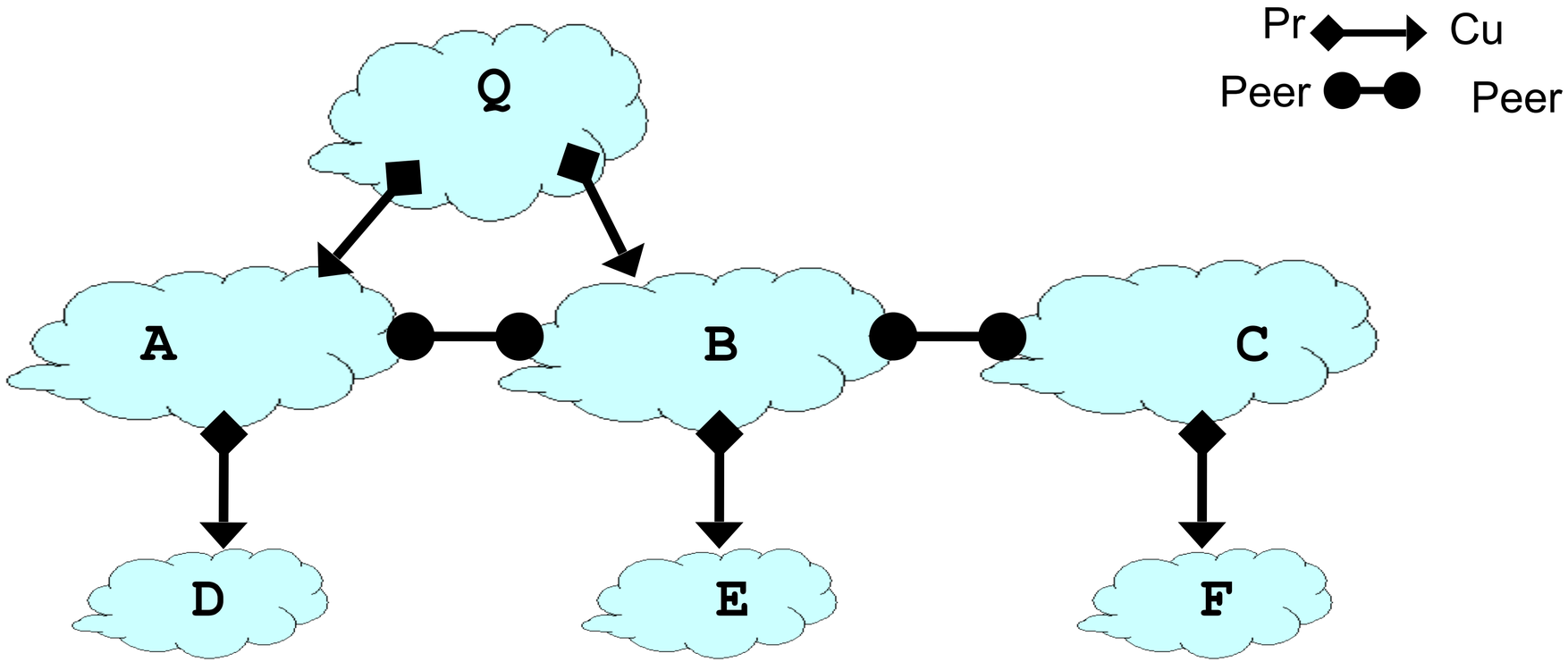
- **Customers pay provider**
- **Peers don't pay each other**

Routing Follows the Money!



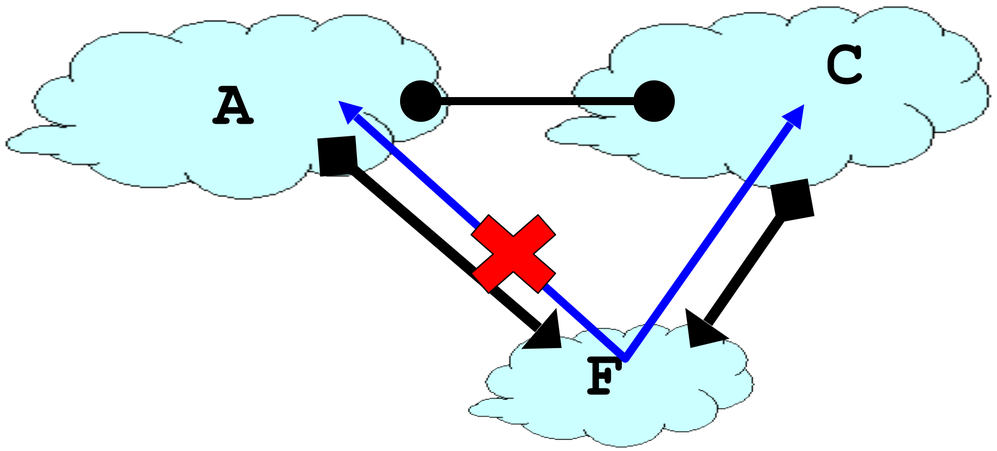
- ASes provide “transit” between their customers
- Peers do not provide transit between other peers

Routing Follows the Money!



- An AS only carries traffic to/from its own customers over a peering link

Routing Follows the Money!



Pr ◆⟶ Cu

Peer ●—● Peer

Interdomain Routing: Setup

- Destinations are IP prefixes (12.0.0.0/8)
- Nodes are Autonomous Systems (ASes)
  - Internals of each AS are hidden
- Links represent both physical links and business relationships
- BGP (Border Gateway Protocol) is the Interdomain routing protocol
  - Implemented by AS border routers

BGP: Basic Idea

An AS advertises ("exports") its best routes to one or more IP prefixes

Each AS selects the "best" route it hears advertised for a prefix

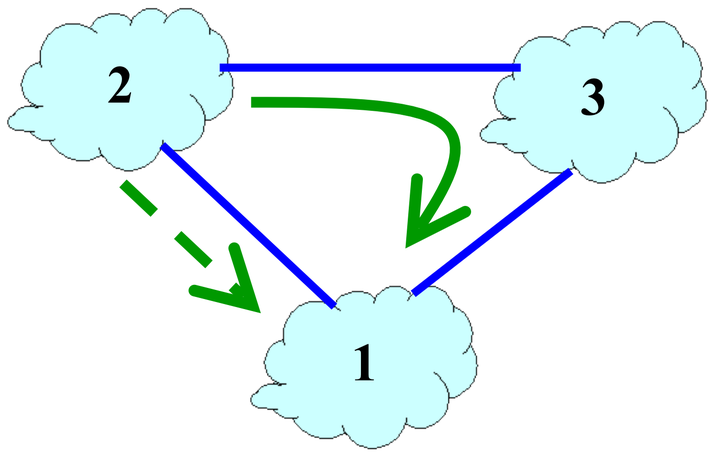
BGP inspired by Distance Vector

- Per-destination route advertisements
- No global sharing of network topology information
- Iterative and distributed convergence on paths
- With four crucial differences!

Differences between BGP and DV
(1) not picking shortest path routes

- BGP selects the best route based on policy, not shortest distance (least cost)

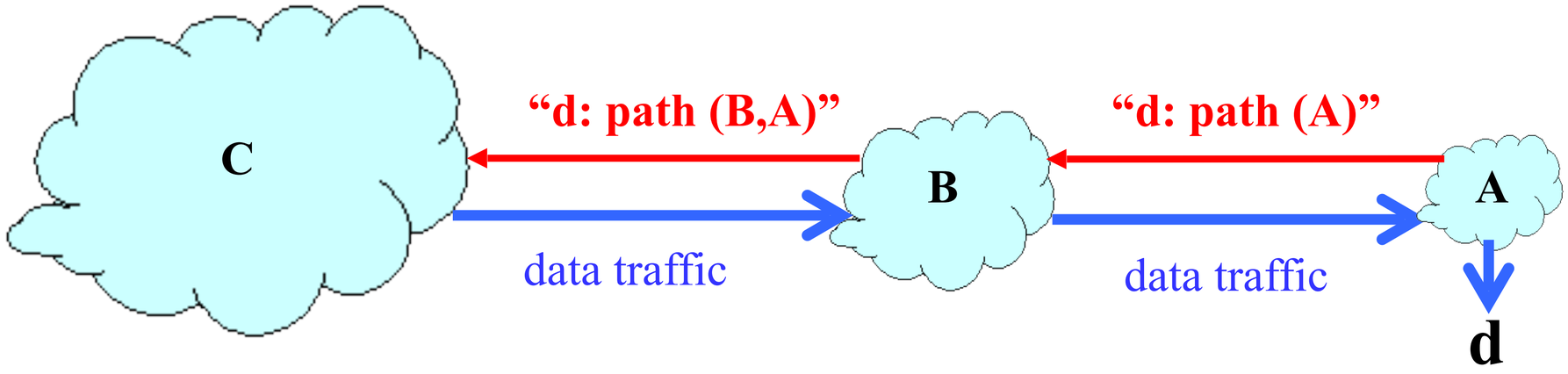
Node 2 may prefer "2, 3, 1" over "2, 1"



- How do we avoid loops?

Differences between BGP and DV
(2) path-vector routing

- Key idea: advertise the entire path
  - Distance vector: send *distance metric* per dest d
  - Path vector: send the *entire path* for each dest d



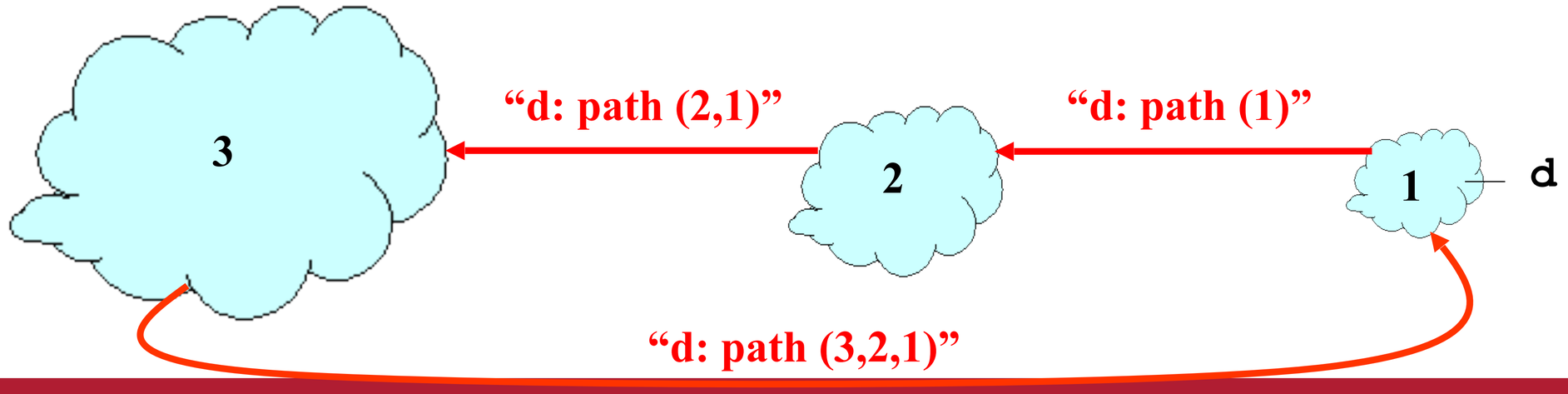
Differences between BGP and DV
(2) path-vector routing

- Key idea: advertise the entire path
  - Distance vector: send *distance metric* per dest d
  - Path vector: send the *entire path* for each dest d

- Benefits
  - loop avoidance is easy

Loop Detection w/ Path-Vector

- Node can easily detect a loop
  - Look for its own node identifier in the path
- Node can simply discard paths with loops
  - E.g., node 1 sees itself in the path “3, 2, 1”
  - E.g., node 1 simply discards the advertisement



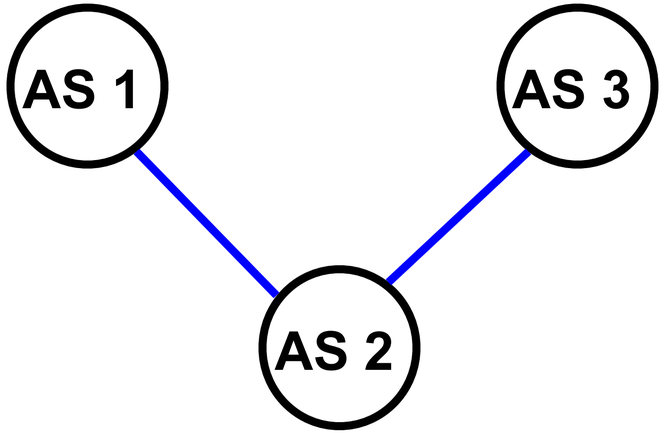
Differences between BGP and DV
(2) path-vector routing

- Key idea: advertise the entire path
  - Distance vector: send *distance metric* per dest d
  - Path vector: send the *entire path* for each dest d

- Benefits
  - loop avoidance is easy
  - flexible policies based on entire path

Differences between BGP and DV
(3) Selective route advertisement

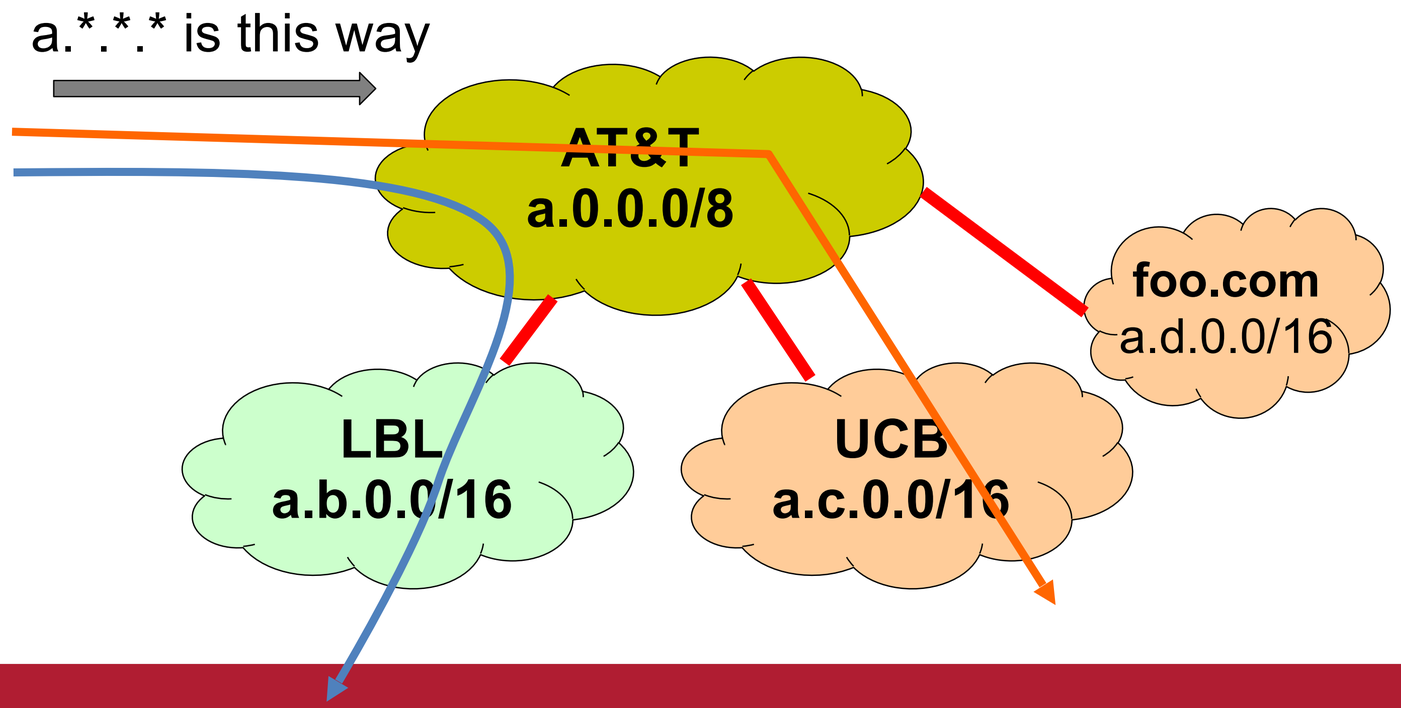
- For policy reasons, an AS may choose not to advertise a route to a destination
- Hence, reachability is not guaranteed even if graph is connected



Example: AS#2 does not want to carry traffic between AS#1 and AS#3

Differences between BGP and DV
(4) BGP may *aggregate* routes

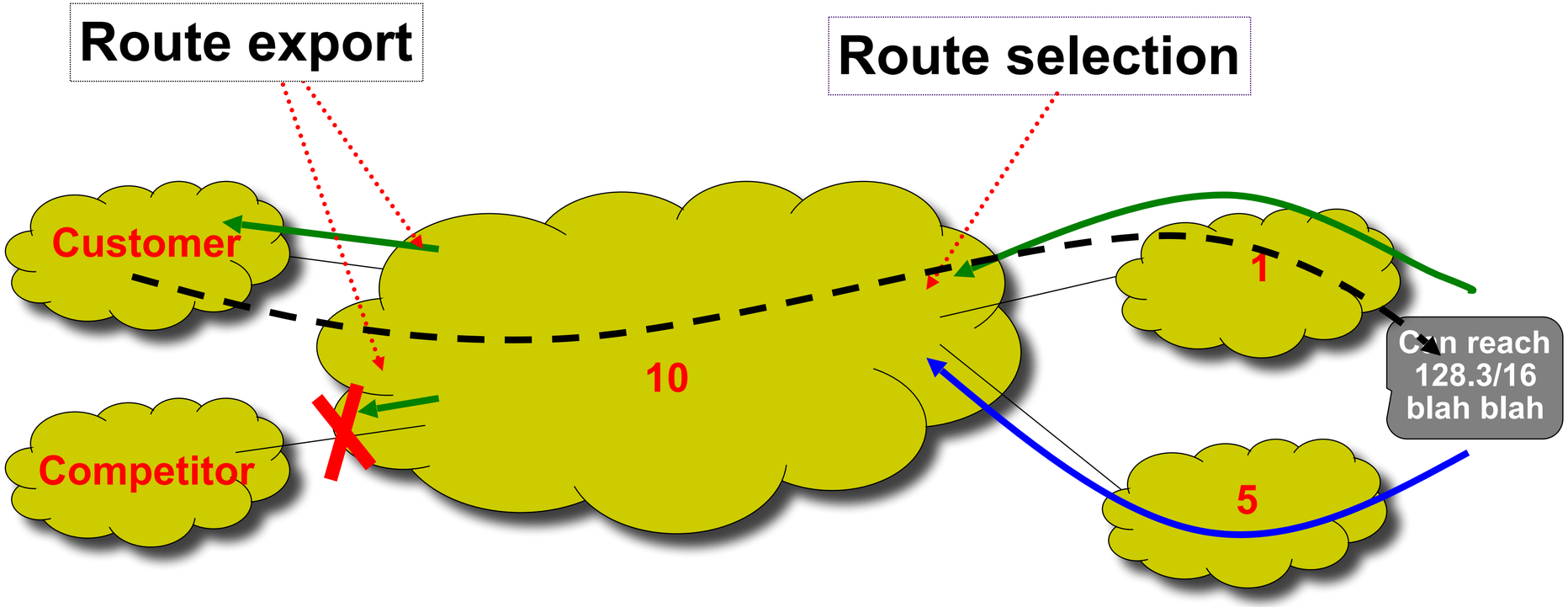
- For scalability, BGP may aggregate routes for different prefixes



BGP

- The role of policy
  - what we mean by it
  - why we need it
- Overall approach
  - four non-trivial changes to DV
  - how policy is implemented (detail-free version)

Policy imposed in how routes are selected and exported



- **Selection**: Which path to use?
  - controls whether/how traffic leaves the network
- **Export**: Which path to advertise?
  - controls whether/how traffic enters the network

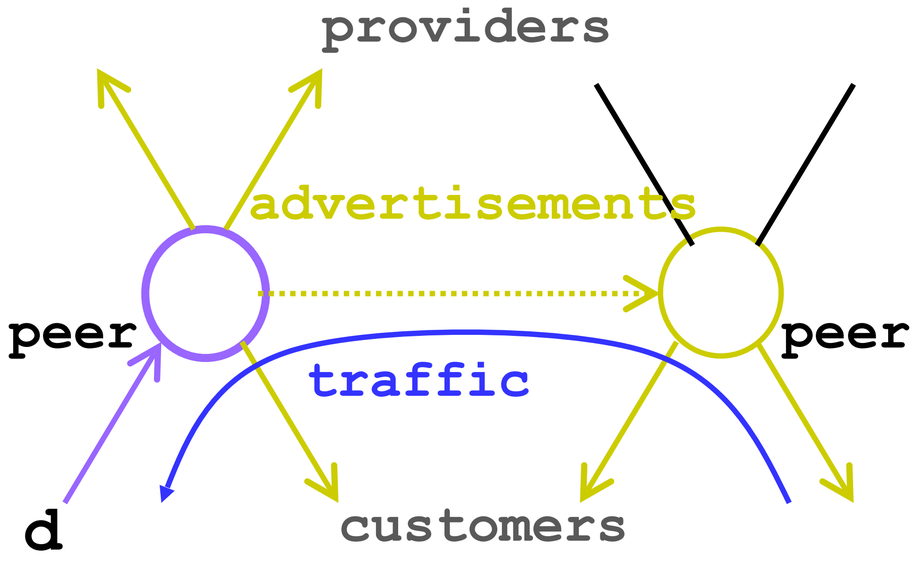
Typical Selection Policy

- In decreasing order of priority
  - make/save money (send to customer > peer > provider)
  - maximize performance (smallest AS path length)
  - minimize use of my network bandwidth (“hot potato”)
  - …
  - …

- BGP uses something called route “attributes” to implement the above

Typical Export: Peer-Peer Case

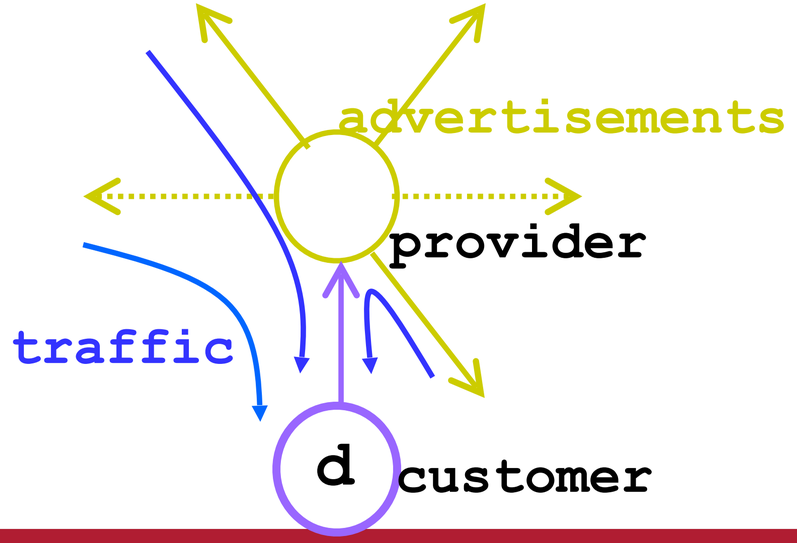
- Peers exchange traffic between their customers
  - AS exports only customer routes to a peer
  - AS exports a peer's routes only to its customers



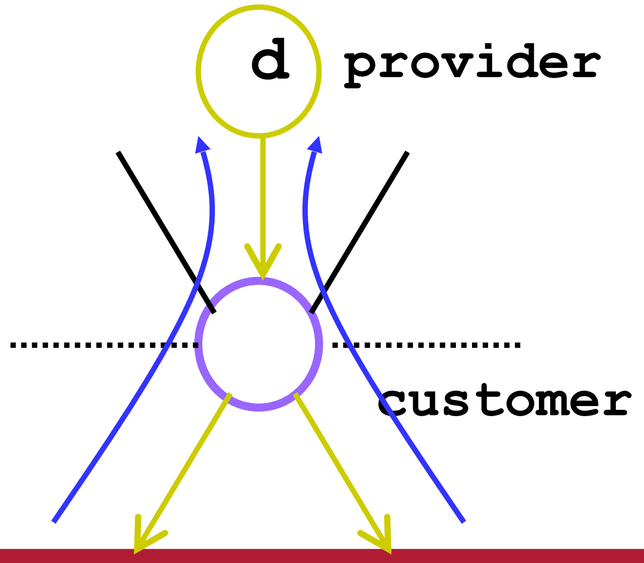
Typical Export: Customer-Provider

- Customer pays provider for access to Internet
  - Provider exports its customer routes to everybody
  - Customer exports provider routes only to its customers

Traffic to customer



Traffic from customer



Typical Export Policy

Destination prefix advertised by…	Export route to…
Customer	Everyone (providers, peers, other customers)
Peer	Customers
Provider	Customers

We'll refer to these as the "Gao-Rexford" rules (capture common -- but not required! -- practice!)

BGP Messages

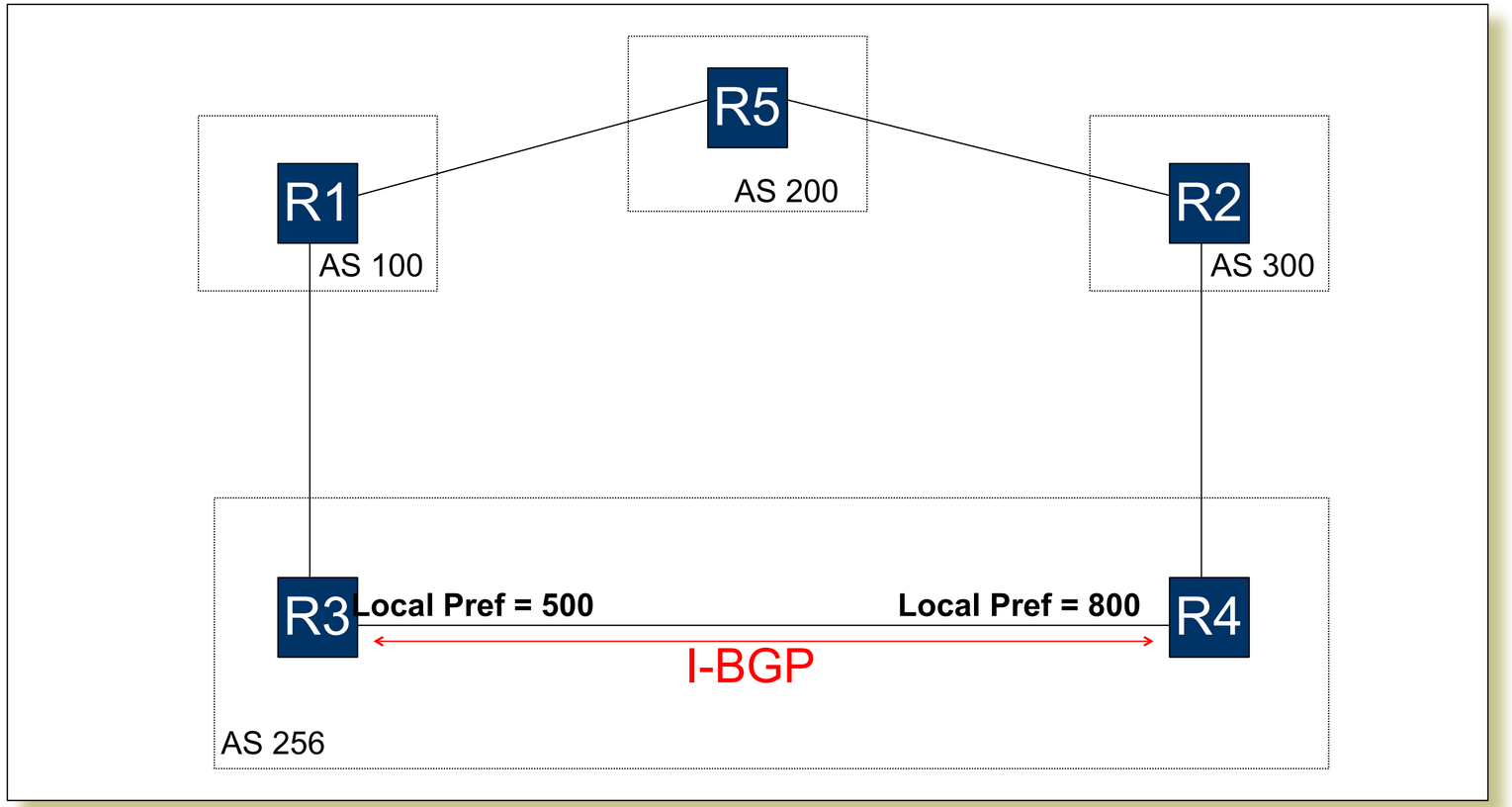
- Open
  - Announces AS ID
  - Determines hold timer – interval between keep_alive or update messages, zero interval implies no keep_alive
- Keep_alive
  - Sent periodically (but before hold timer expires) to peers to ensure connectivity.
  - Sent in place of an UPDATE message
- Notification
  - Used for error notification
  - TCP connection is closed *immediately* after notification

BGP UPDATE Message

- List of withdrawn routes
- Network layer reachability information
  - List of reachable prefixes
- Path attributes
  - Origin
  - Path
  - Metrics: used by policies for path selection
- All prefixes advertised in message have same path attributes

LOCAL PREF

- Local (within an AS) mechanism to provide relative priority among BGP routers (e.g. R3 over R4)

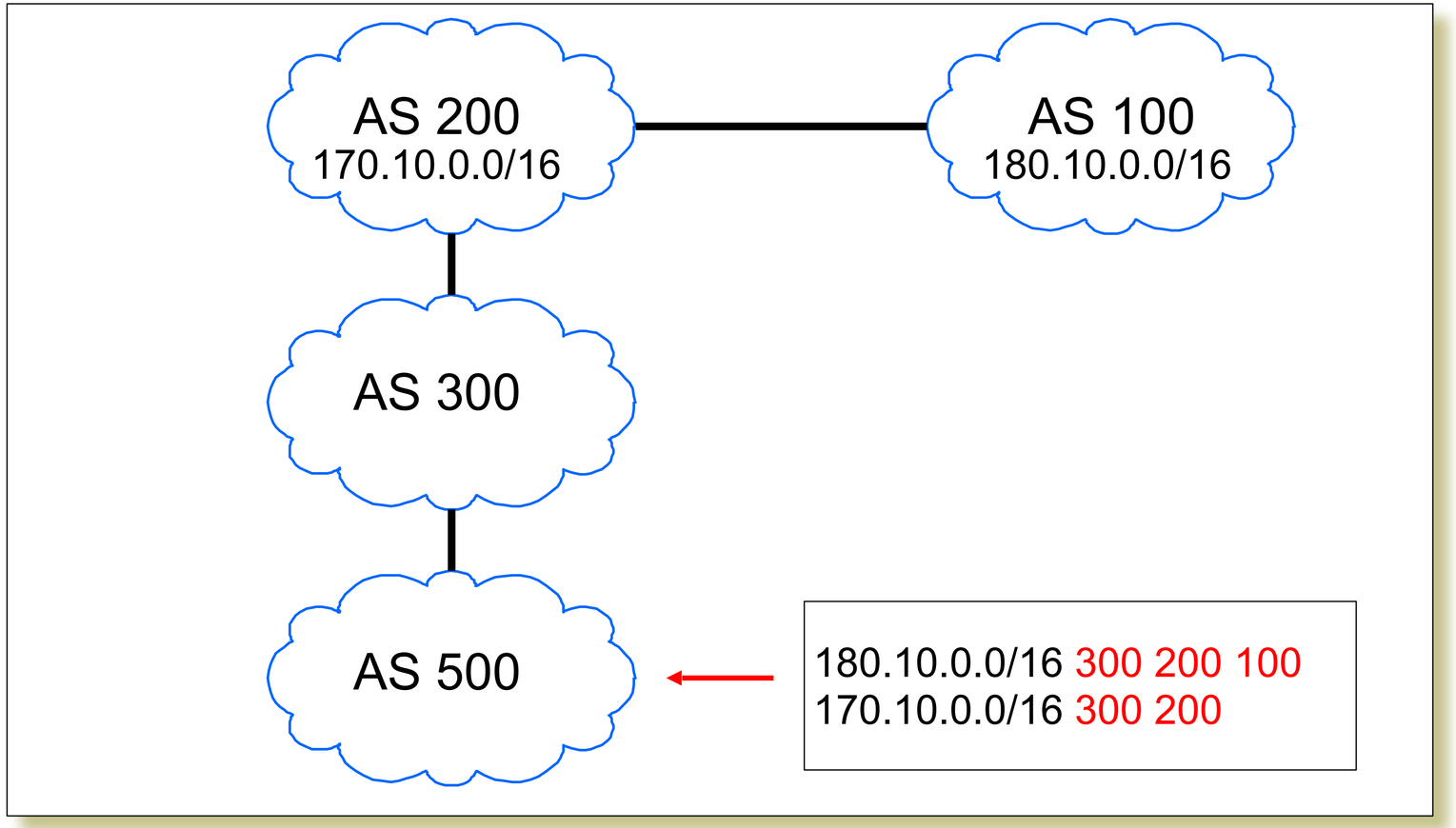


LOCAL PREF – Common Uses

- Routers have a default LOCAL PREF
  - Can be changed for specific ASes
- Peering vs. transit
  - Prefer to use peering connection, why?
- In general, customer > peer > provider
  - Use LOCAL PREF to ensure this

AS_PATH

- List of traversed AS's

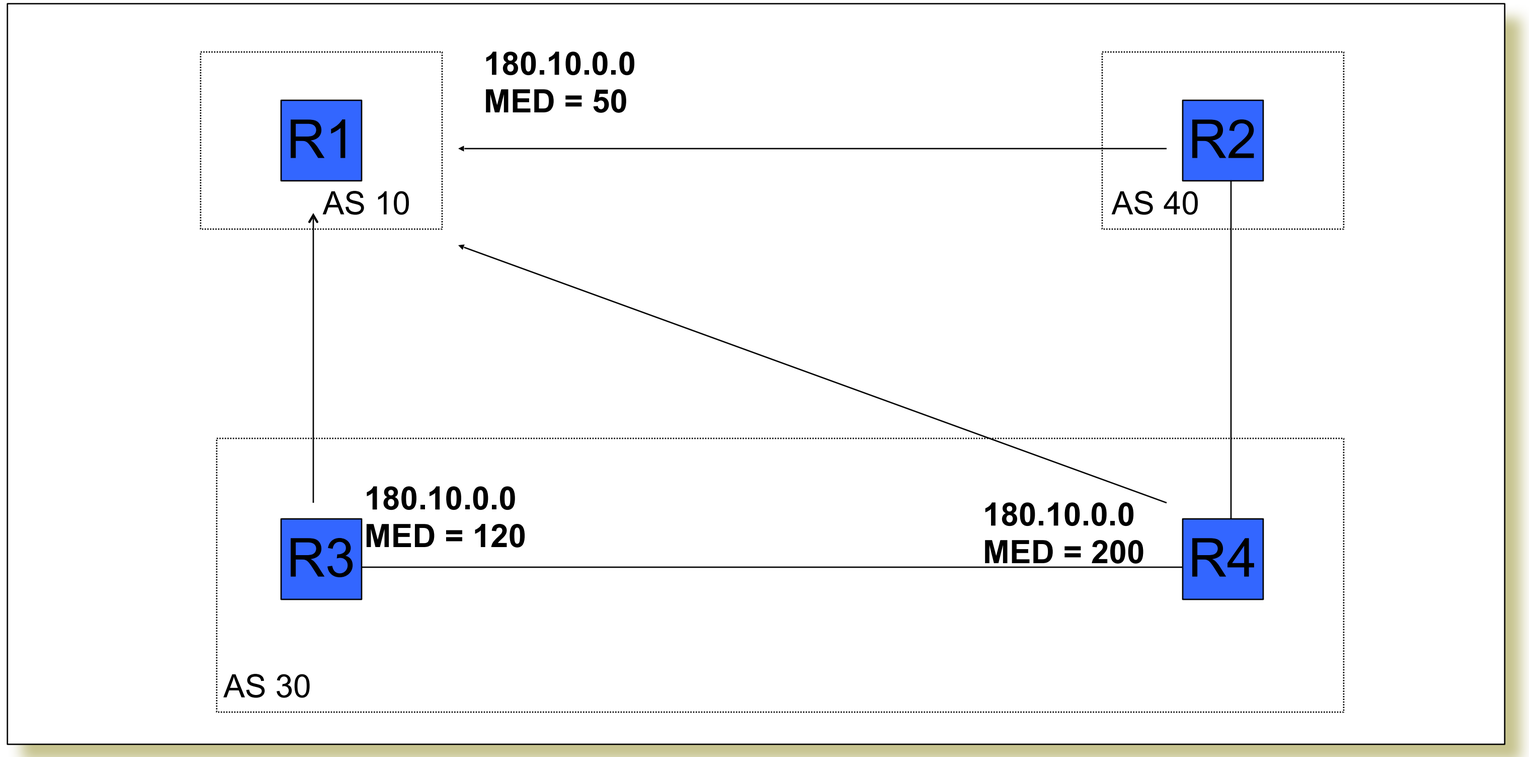


Multi-Exit Discriminator (MED)

- Hint to external neighbors about the preferred path into an AS
  - Non-transitive attribute
    - Different AS choose different scales
- Used when two AS's connect to each other in more than one place

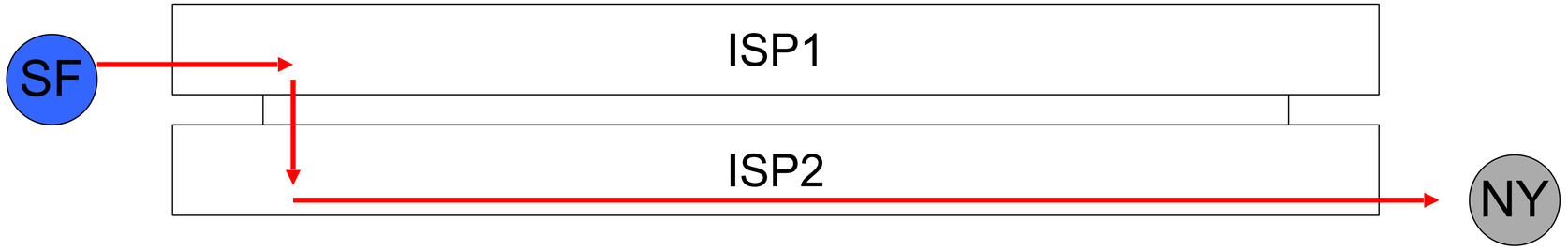
MED

- Hint to R1 to use R3 over R4 link
- Cannot compare AS40's values to AS30's



MED

- MED is typically used in provider/subscriber scenarios
- It can lead to unfairness if used between ISP because it may force one ISP to carry more traffic:



- ISP1 ignores MED from ISP2
- ISP2 obeys MED from ISP1
- ISP2 ends up carrying traffic most of the way

Path Selection Criteria

- Attributes + external (policy) information
- Rough ordering for path selection
  - Highest LOCAL-PREF
    - Captures business relationships and other factors
  - Shortest AS-PATH
  - Lowest origin type
  - Lowest MED (if routes learned from same neighbor)
  - eBGP over iBGP-learned
  - Lowest internal routing cost to border router
  - Tie breaker, e.g., lowest router ID

More BGP Questions?

Y’all (yinz?) had some blank faces when I recapped TCP on Tuesday…

So let’s do that too…

Quick Q: What are the goals of TCP?

All These Windows…

- Congestion Window: CWND
  - How many bytes can be sent without overflowing routers
  - Computed by the sender using congestion control algorithm
- Flow control window: AdvertisedWindow (RWND)
  - How many bytes can be sent without overflowing receiver's buffers
  - Determined by the receiver and reported to the sender
- Sender-side window = minimum{CWND, RWND}
  - Assume for this lecture that RWND >> CWND

Note

- This lecture will talk about CWND in units of MSS
  - (Recall MSS: Maximum Segment Size, the amount of payload data in a TCP packet)
  - This is only for pedagogical purposes

- Keep in mind that real implementations maintain CWND in bytes

Two Basic Questions

- How does the sender detect congestion?
- How does the sender adjust its sending rate?
  - To address three issues
    - Finding available bottleneck bandwidth
    - Adjusting to bandwidth variations
    - Sharing bandwidth

Detecting Congestion

- Packet delays
  - Tricky: noisy signal (delay often varies considerably)
- Routers tell endhosts when they're congested
- Packet loss
  - Fail-safe signal that TCP already has to detect
  - Complication: non-congestive loss (checksum errors)

Not All Losses the Same

- Duplicate ACKs: isolated loss
  - Still getting ACKs
- Timeout: much more serious
  - Not enough dupacks
  - Must have suffered several losses
- Will adjust rate differently for each case

Rate Adjustment

- Basic structure:
  - Upon receipt of ACK (of new data): increase rate
  - Upon detection of loss: decrease rate

- How we increase/decrease the rate depends on the phase of congestion control we're in:
  - Discovering available bottleneck bandwidth *vs.*
  - Adjusting to bandwidth variations

Bandwidth Discovery with Slow Start

- Goal: estimate available bandwidth
  - start slow (for safety)
  - but ramp up quickly (for efficiency)

- Consider
  - RTT = 100ms, MSS=1000bytes
  - Window size to fill 1Mbps of BW = 12.5 packets
  - Window size to fill 1Gbps = 12,500 packets
  - Either is possible!

“Slow Start” Phase

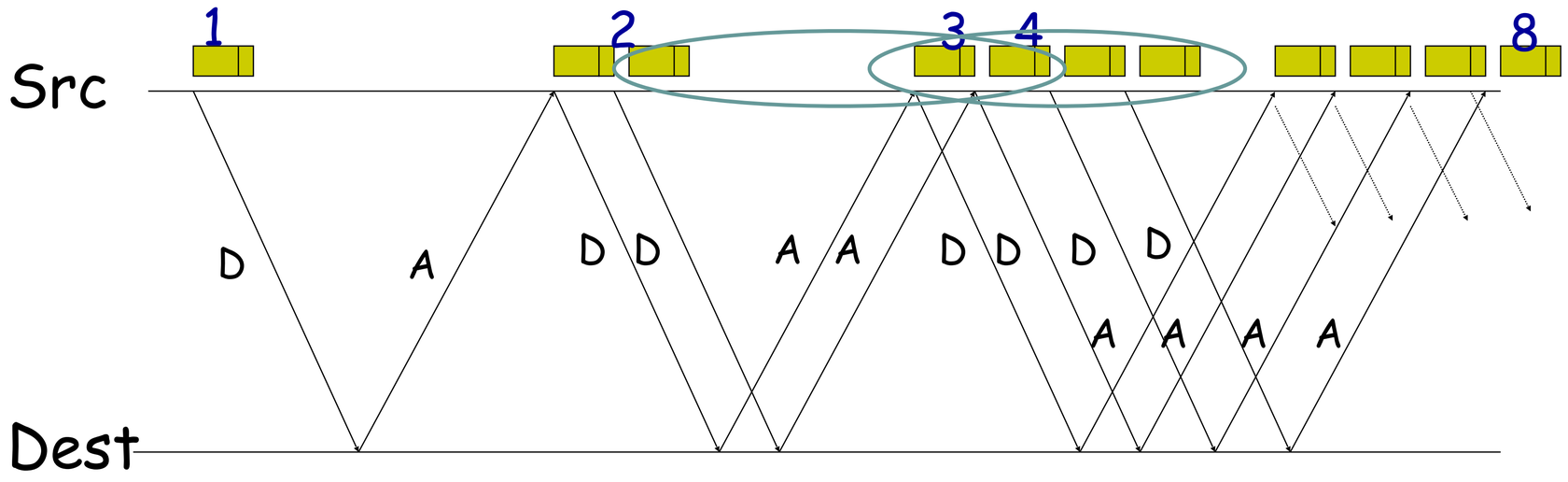
- Sender starts at a slow rate but increases **exponentially** until first loss

- Start with a small congestion window
  - Initially, CWND = 1
  - So, initial sending rate is MSS/RTT

- Double the CWND for each RTT with no loss

Slow Start in Action

- For each RTT: double CWND
- Simpler implementation: for each ACK, CWND += 1



Adjusting to Varying Bandwidth

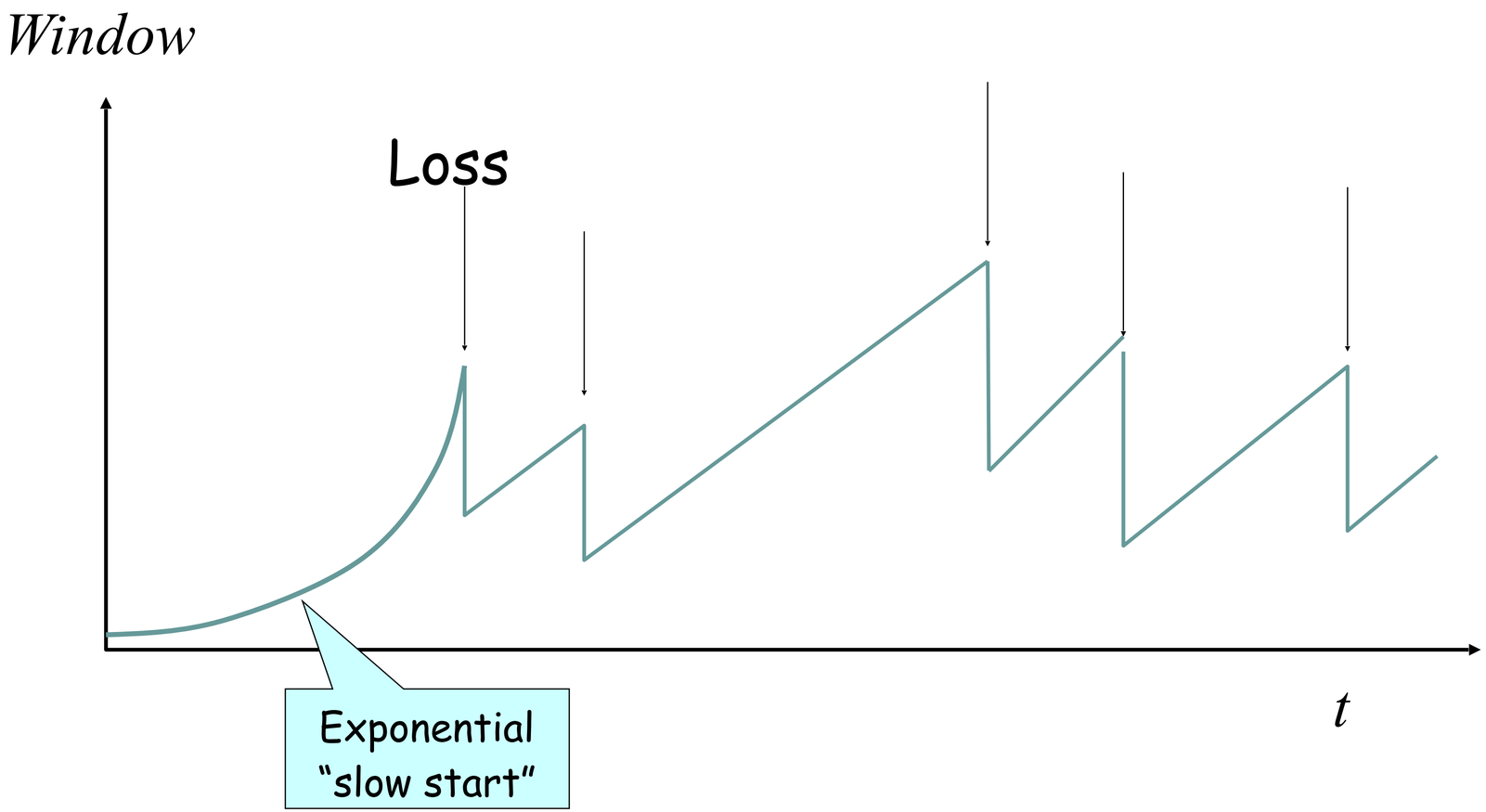
- Slow start gave an estimate of available bandwidth
- Now, want to track variations in this available bandwidth, oscillating around its current value
  - Repeated probing (rate increase) and backoff (rate decrease)
- TCP uses: “Additive Increase Multiplicative Decrease” (AIMD)
  - We’ll see why shortly…

AIMD

- Additive increase
  - Window grows by one MSS for every RTT with no loss
  - For each successful RTT, CWND = CWND + 1
  - Simple implementation:
    - for each ACK, CWND = CWND+ 1/CWND

- Multiplicative decrease
  - On loss of packet, divide congestion window in **half**
  - On loss, CWND = CWND/2

Leads to the TCP "Sawtooth"



Slow-Start vs. AIMD

- When does a sender stop Slow-Start and start Additive Increase?

- Introduce a “slow start threshold” (ssthresh)
  - Initialized to a large value
  - On timeout, ssthresh = CWND/2

- When CWND = ssthresh, sender switches from slow-start to AIMD-style increase

Why AIMD?

Recall: Three Issues

- Discovering the available (bottleneck) bandwidth
  - Slow Start
- Adjusting to variations in bandwidth
  - AIMD
- Sharing bandwidth between flows

Goals for bandwidth sharing

- Efficiency: High utilization of link bandwidth
- Fairness: Each flow gets equal share

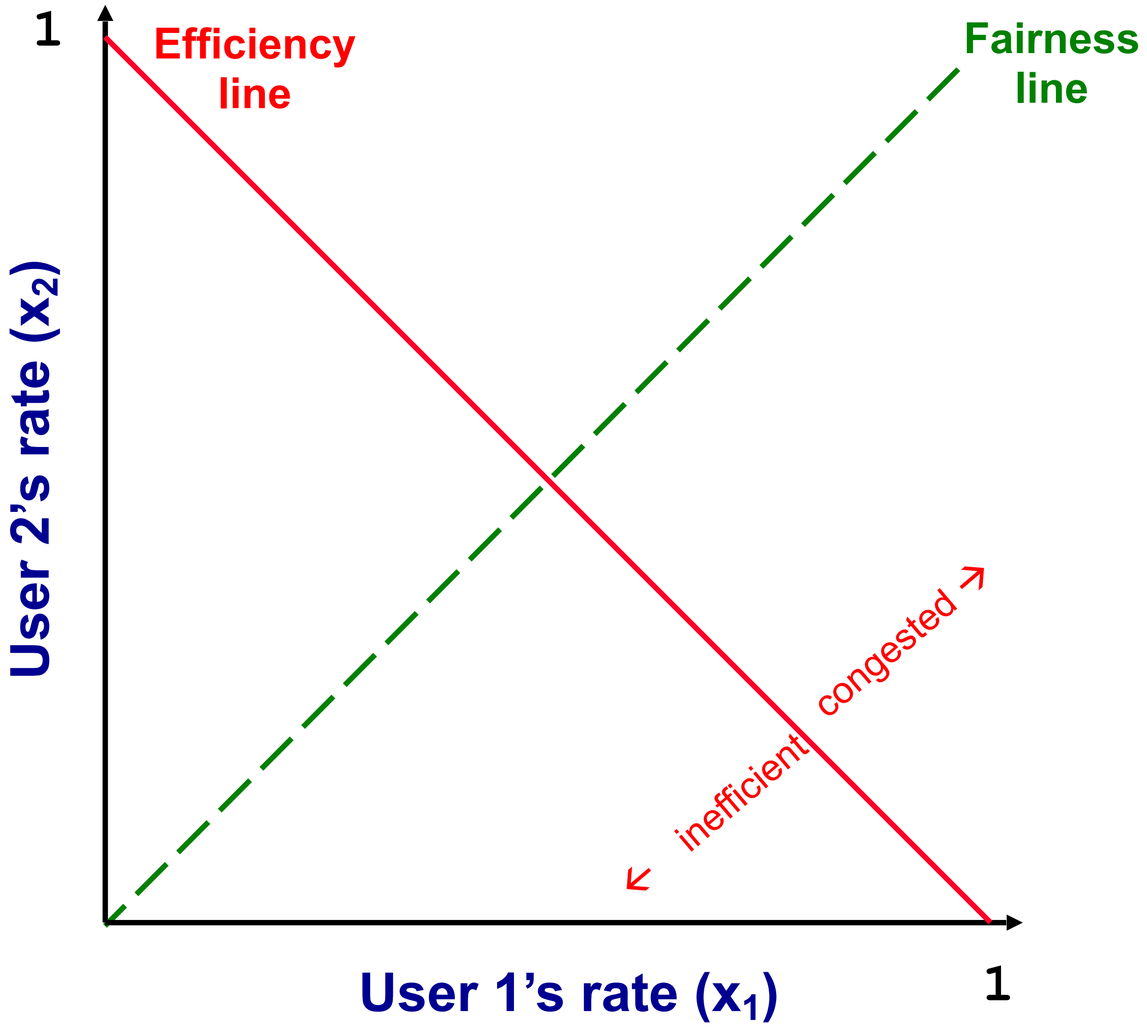
Why AIMD?

- Some rate adjustment options: Every RTT, we can
  - Multiplicative increase or decrease: CWND→ a*CWND
  - Additive increase or decrease: CWND→ CWND + b

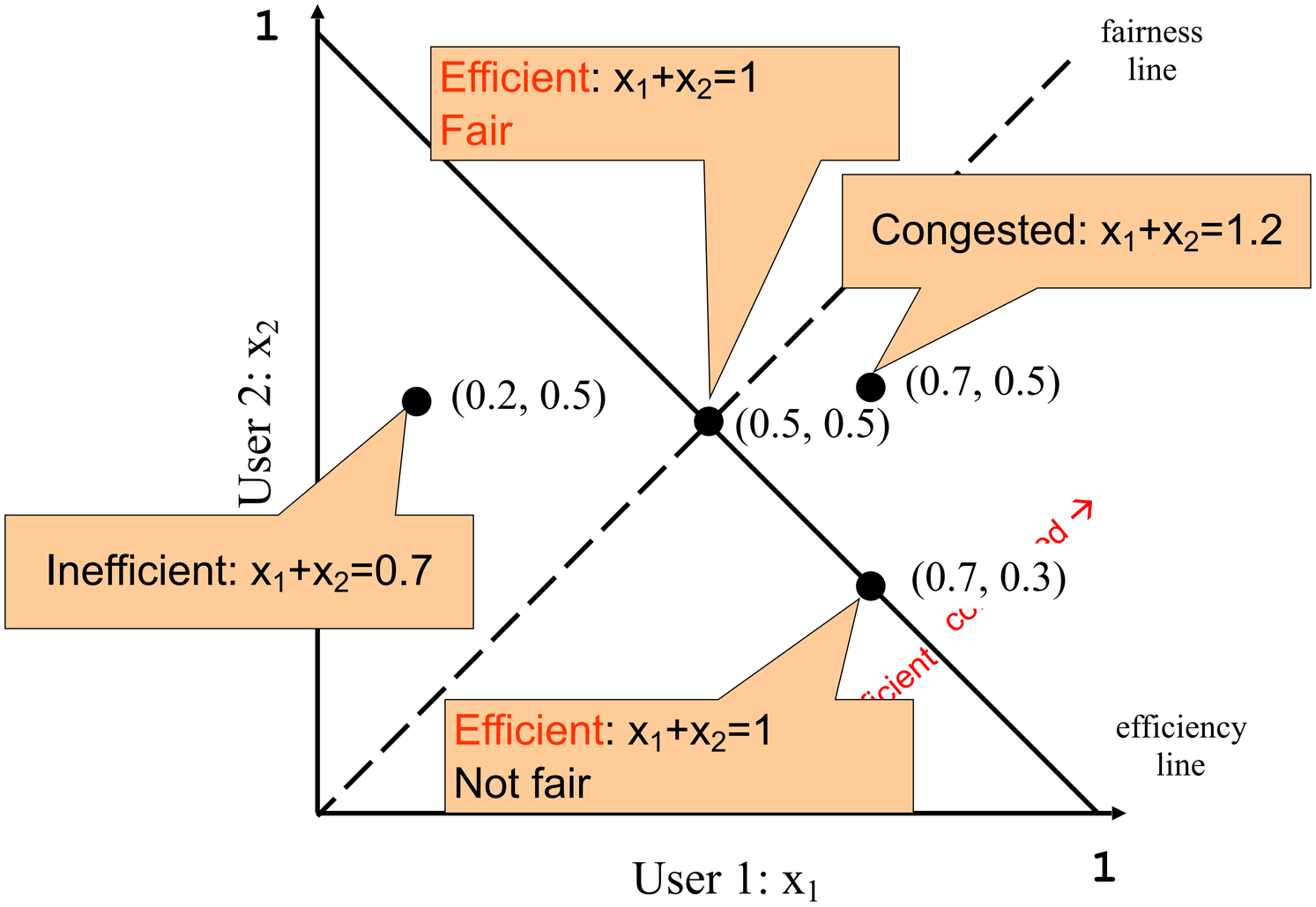
- Four alternatives:
  - AIAD: gentle increase, gentle decrease
  - AIMD: gentle increase, drastic decrease
  - MIAD: drastic increase, gentle decrease
  - MIMD: drastic increase and decrease

Simple Model of Congestion Control

- Two users
  - rates x_1 and x_2
- Congestion when $x_1+x_2 > 1$
- Unused capacity when $x_1+x_2 < 1$
- Fair when $x_1 = x_2$

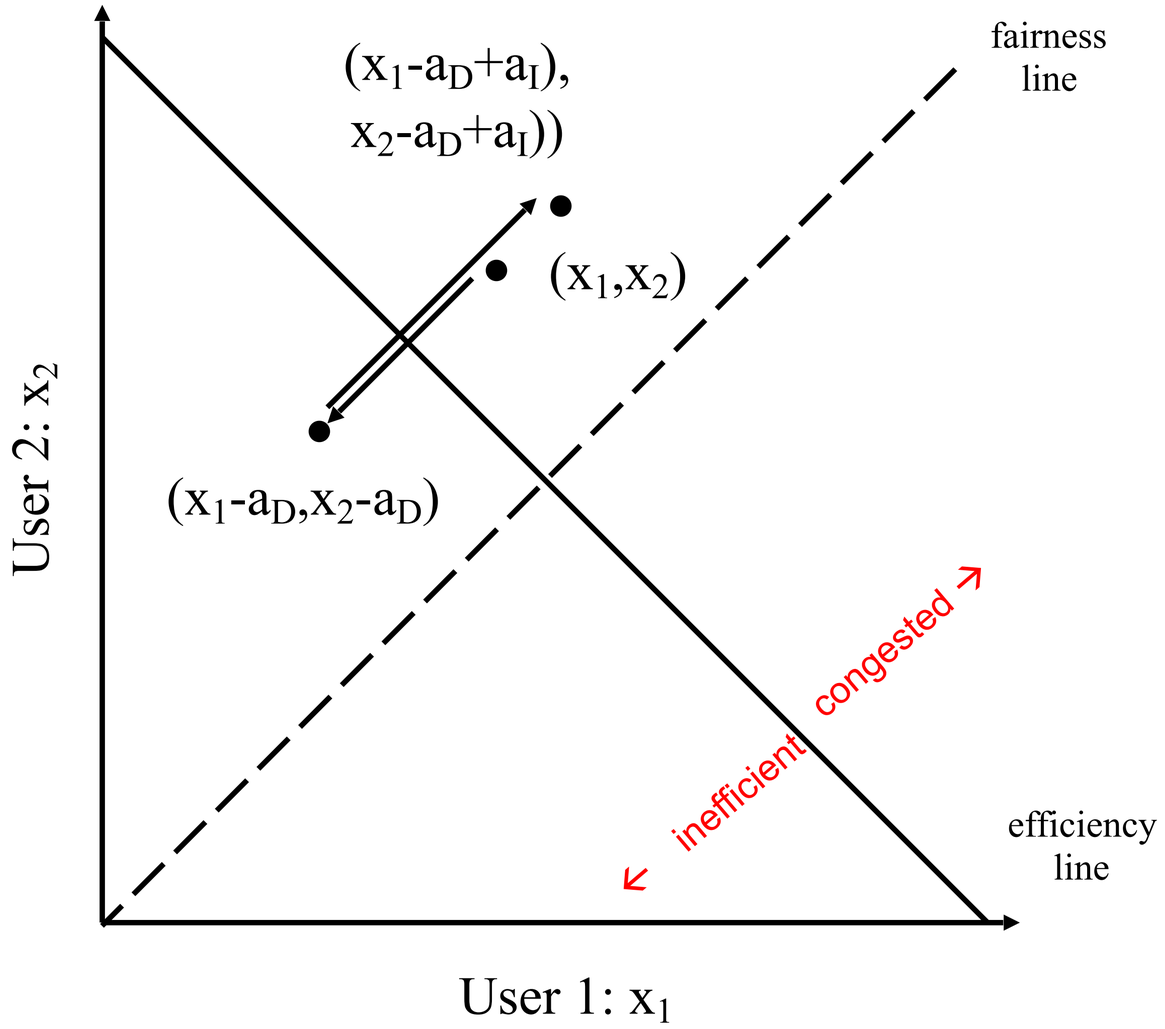


Example


1
fairness line
Efficient: $x_1+x_2=1$
Fair
Congested: $x_1+x_2=1.2$
User 2: x_2
(0.2, 0.5)
(0.5, 0.5)
(0.7, 0.5)
Inefficient: $x_1+x_2=0.7$
(0.7, 0.3)
Efficient: $x_1+x_2=1$
Not fair
efficiency line
User 1: x_1
1

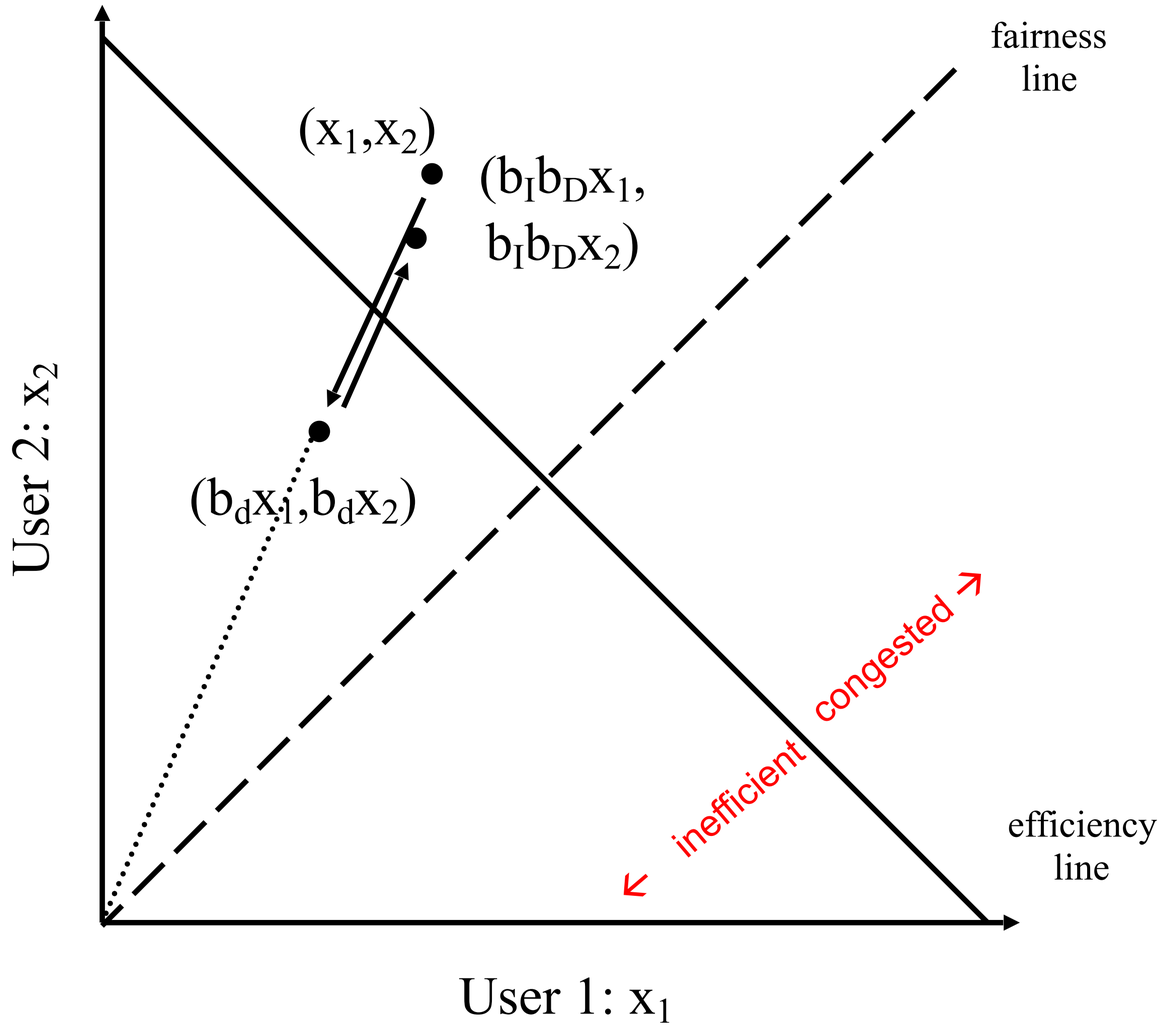
AIAD

- Increase: $x + a_I$
- Decrease: $x - a_D$
- Does not converge to fairness



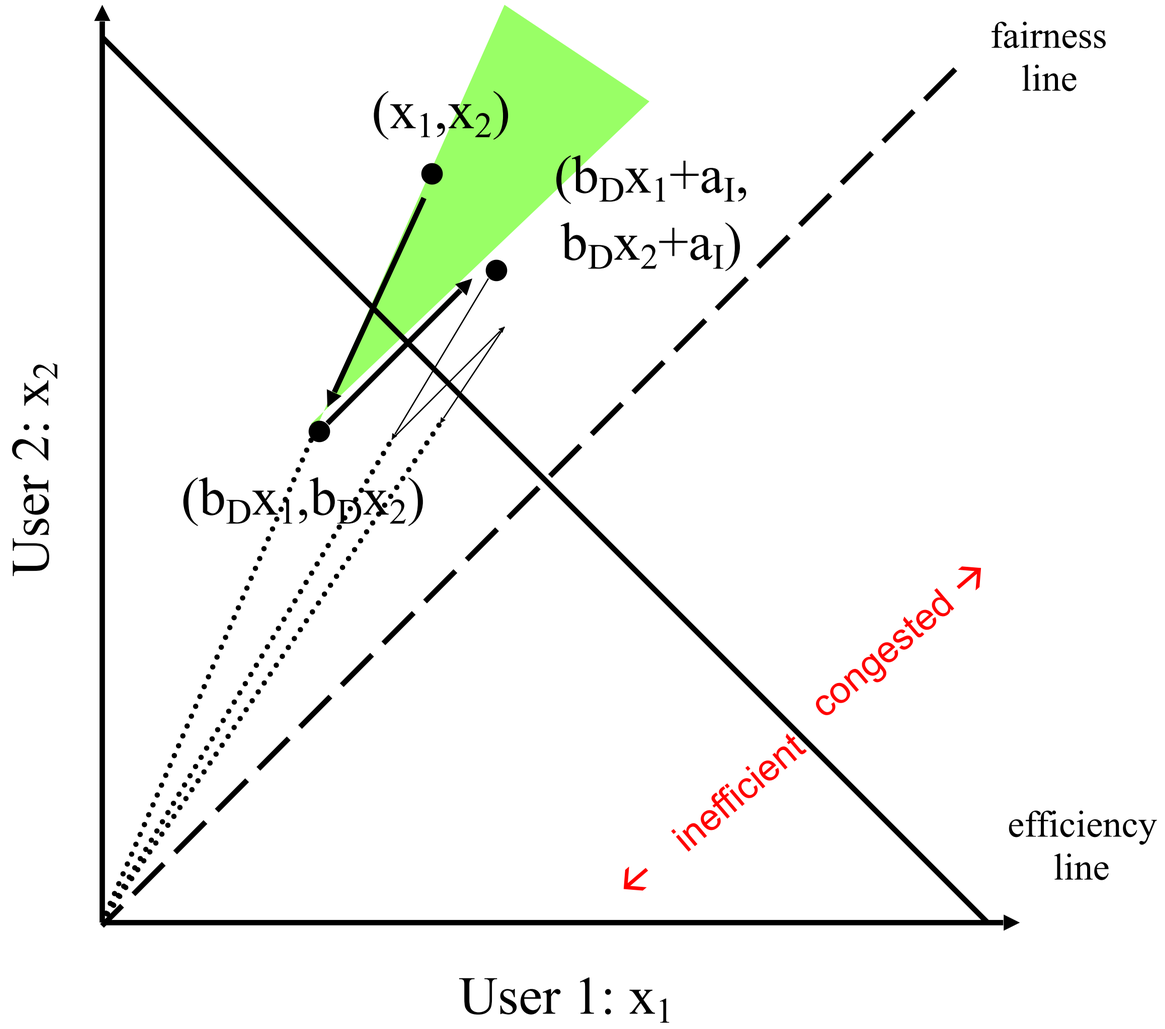
MIMD

- Increase: $x*b_I$
- Decrease: $x*b_D$
- Does not converge to fairness

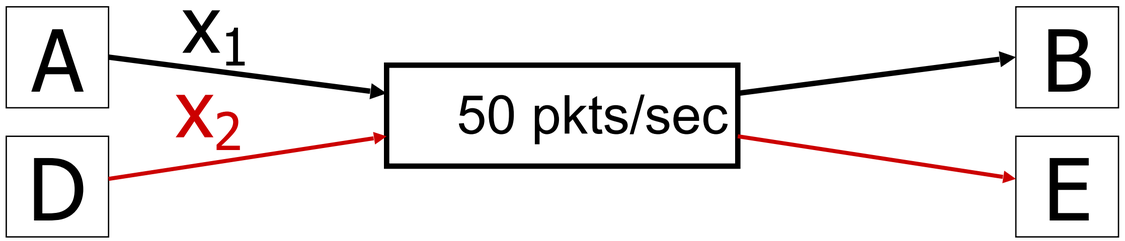


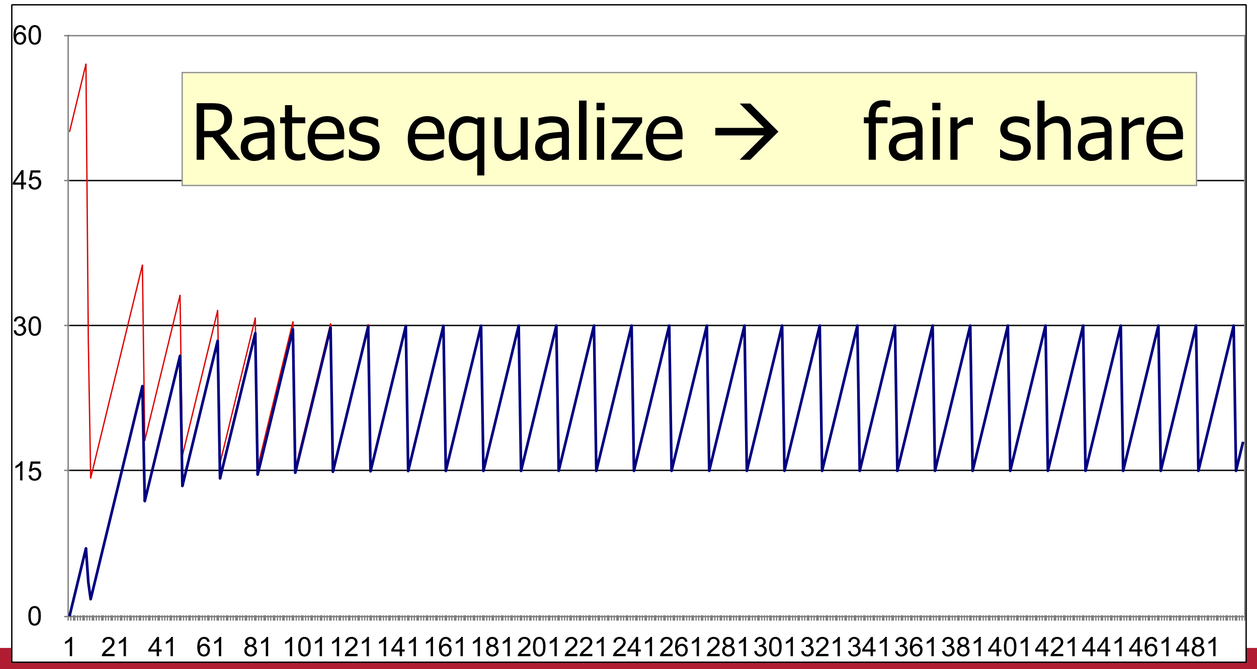
AIMD

- Increase: $x+a_I$
- Decrease: $x*b_D$
- Converges to fairness

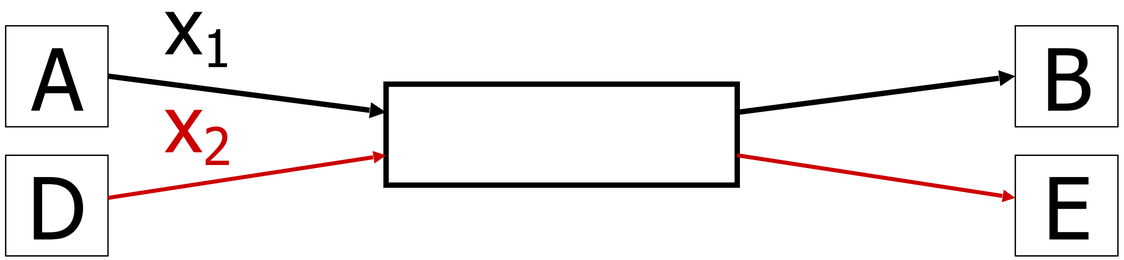


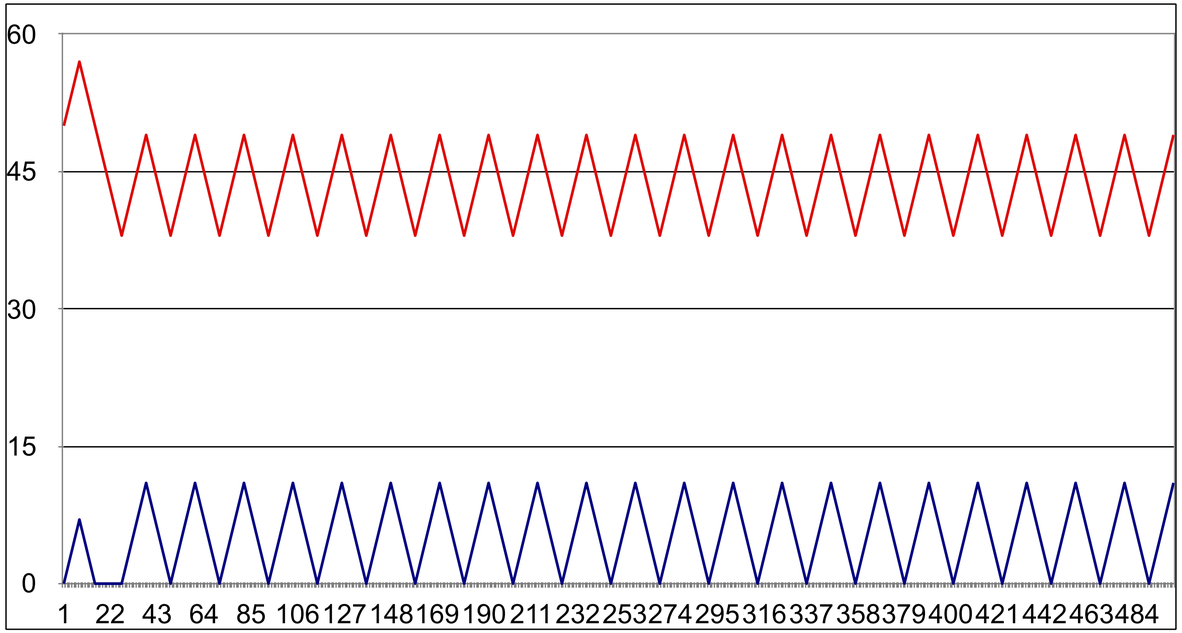
AIMD Sharing Dynamics





AIAD Sharing Dynamics





TCP Congestion Control Details

Implementation

- State at sender
  - CWND (initialized to a small constant)
  - ssthresh (initialized to a large constant)
  - [Also dupACKcount and timer, as before]

- Events
  - ACK (new data)
  - dupACK (duplicate ACK for old data)
  - Timeout

Event: ACK (new data)

- If CWND < ssthresh
  - CWND += 1

- *CWND packets per RTT*
- *Hence after one RTT with no drops:*
  CWND = 2xCWND

Event: ACK (new data)

- If CWND < ssthresh
  - CWND += 1

Slow start phase

- Else
  - CWND = CWND + 1/CWND

"Congestion Avoidance" phase (additive increase)

- *CWND packets per RTT*
- *Hence after one RTT with no drops:*
  CWND = CWND + 1

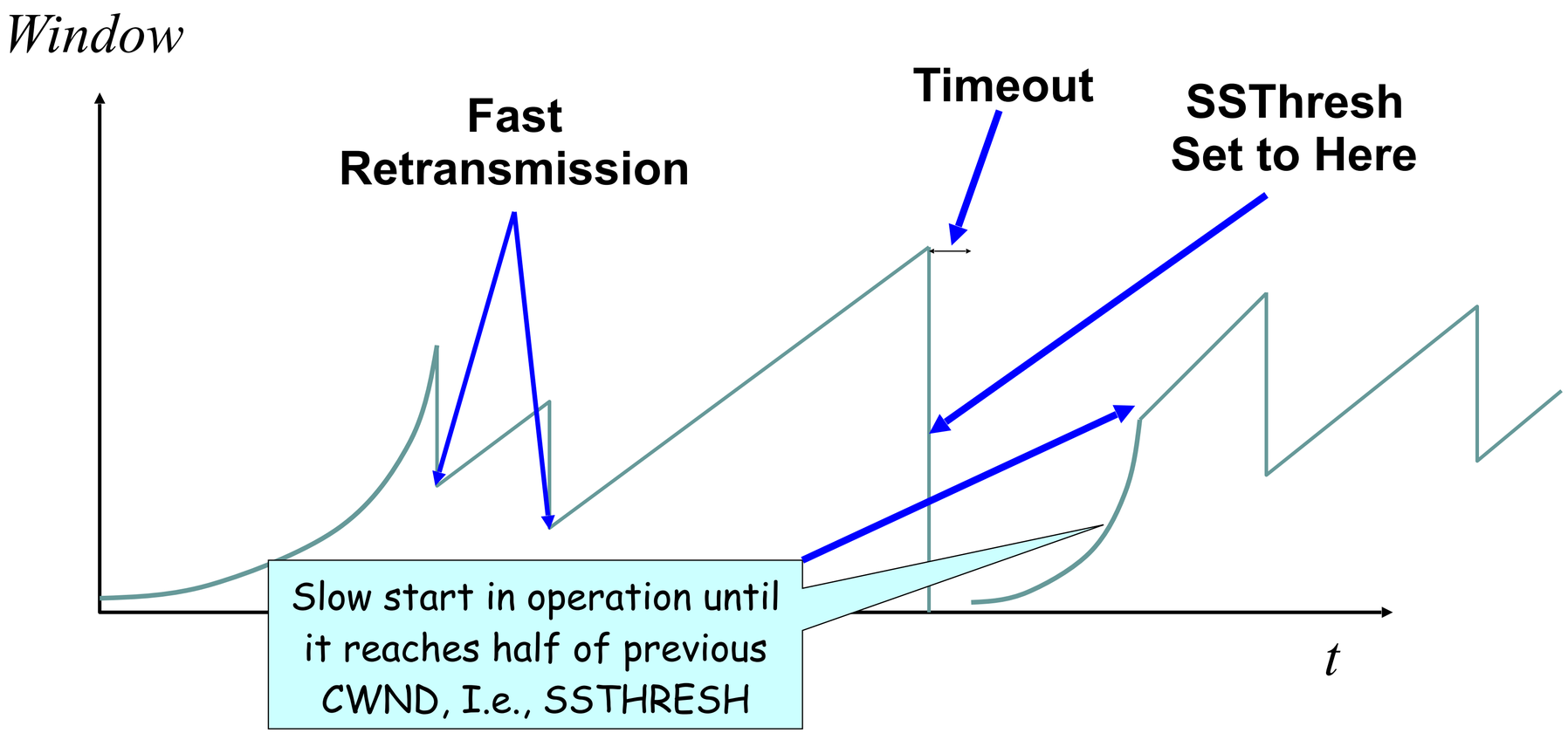
Event: TimeOut

- On Timeout
  - ssthresh ← CWND/2
  - CWND ← 1

Event: dupACK

- dupACKcount ++
- If dupACKcount = 3 /* fast retransmit */
  - ssthresh = CWND/2
  - CWND = CWND/2

Example



Slow-start restart: Go back to CWND = 1 MSS, but take advantage of knowing the previous value of CWND

One Final Phase: Fast Recovery

- The problem: congestion avoidance too slow in recovering from an isolated loss

Example (in units of MSS, not bytes)

- Consider a TCP connection with:
  - CWND=10 packets
  - Last ACK was for packet # 101
    - i.e., receiver expecting next packet to have seq. no. 101

- 10 packets [101, 102, 103,…, 110] are in flight
  - Packet 101 is dropped
  - What ACKs do they generate?
  - And how does the sender respond?

Timeline

- ACK 101 (due to 102) cwnd=10 dupACK#1 (no xmit)
- ACK 101 (due to 103) cwnd=10 dupACK#2 (no xmit)
- ACK 101 (due to 104) cwnd=10 dupACK#3 (no xmit)
- RETRANSMIT 101 ssthresh=5 cwnd= 5
- ACK 101 (due to 105) cwnd=5 + 1/5 (no xmit)
- ACK 101 (due to 106) cwnd=5 + 2/5 (no xmit)
- ACK 101 (due to 107) cwnd=5 + 3/5 (no xmit)
- ACK 101 (due to 108) cwnd=5 + 4/5 (no xmit)
- ACK 101 (due to 109) cwnd=5 + 5/5 (no xmit)
- ACK 101 (due to 110) cwnd=6 + 1/5 (no xmit)
- ACK 111 (due to 101) ← only now can we transmit new packets
- Plus no packets in flight so ACK “clocking” (to increase CWND) stalls for another RTT

Solution: Fast Recovery

Idea: Grant the sender temporary “credit” for each dupACK so as to keep packets in flight

- If dupACKcount = 3
  - ssthresh = cwnd/2
  - cwnd = ssthresh + 3
- While in fast recovery
  - cwnd = cwnd + 1 for each additional duplicate ACK
- Exit fast recovery after receiving new ACK
  - set cwnd = ssthresh

Example

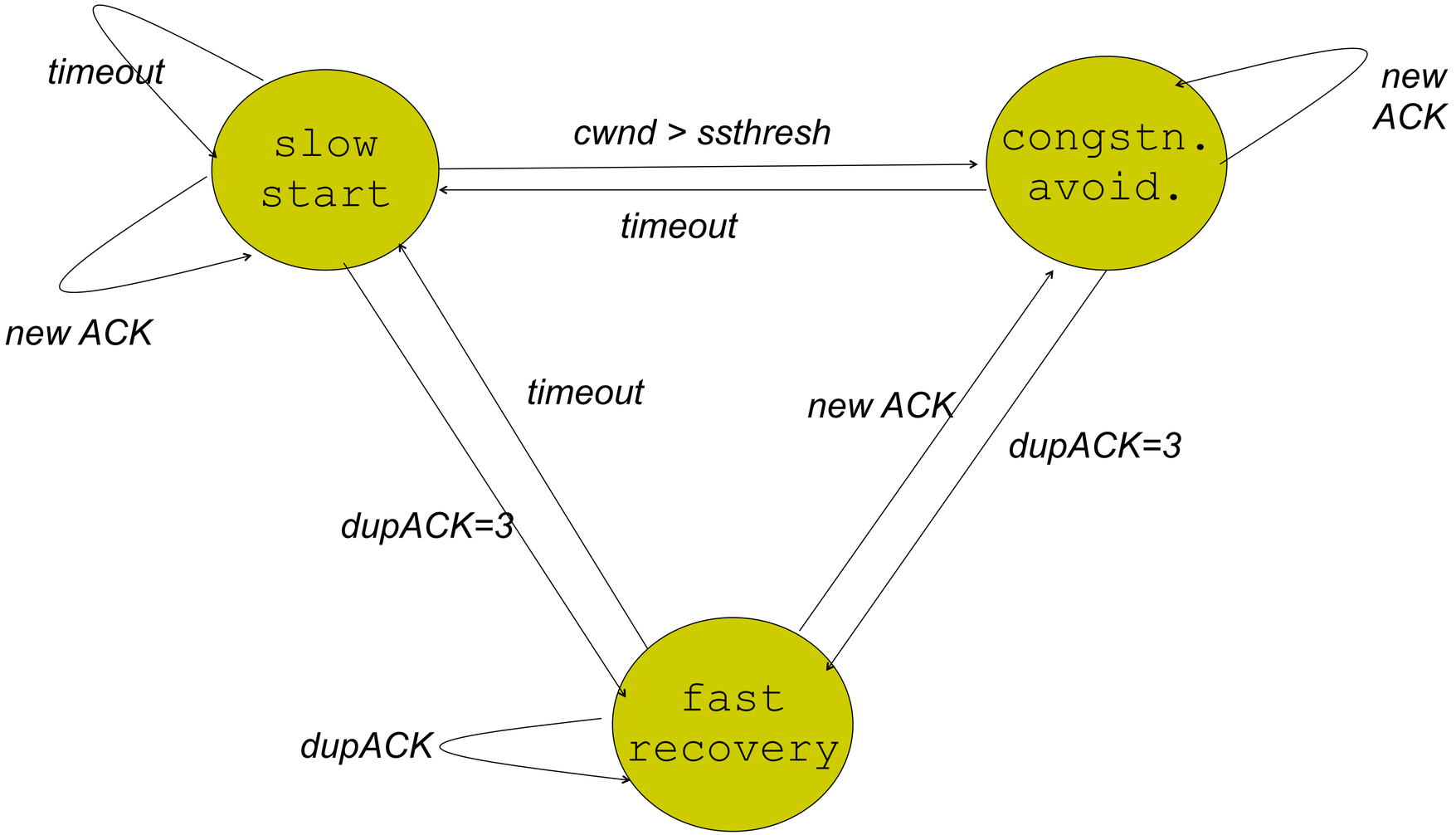
- Consider a TCP connection with:
  - CWND=10 packets
  - Last ACK was for packet # 101
    - i.e., receiver expecting next packet to have seq. no. 101

- 10 packets [101, 102, 103,…, 110] are in flight
  - Packet 101 is dropped

Timeline

- ACK 101 (due to 102) cwnd=10 dup#1
- ACK 101 (due to 103) cwnd=10 dup#2
- ACK 101 (due to 104) cwnd=10 dup#3
- REXMIT 101 ssthresh=5 cwnd= 8 (5+3)
- ACK 101 (due to 105) cwnd= 9 (no xmit)
- ACK 101 (due to 106) cwnd=10 (no xmit)
- ACK 101 (due to 107) cwnd=11 (xmit 111)
- ACK 101 (due to 108) cwnd=12 (xmit 112)
- ACK 101 (due to 109) cwnd=13 (xmit 113)
- ACK 101 (due to 110) cwnd=14 (xmit 114)
- ACK 111 (due to 101) cwnd = 5 (xmit 115) ← exiting fast recovery
- Packets 111-114 already in flight
- ACK 112 (due to 111) cwnd = 5 + 1/5 ← back in congestion avoidance

Putting it all together: The TCP State Machine (partial)



Questions? Ask now.