CMSC417 Spring 2016 Lecture #12 3/21/2016

Agenda

- ⇒ grades posted for p2 & midterm
  - □ midterms handed back Wed
- ⇒ p3 assigned by tomorrow
- ⇒ office hours moved a bit this week

- ⇒ BGP (cont'd)
  - □ quick review
  - □ valley-free routing
  - □ policy examples
  - □ joining BGP & IGP
- ⇒ Reliable Transmission
  - □ sequence #s
  - □ stop and wait
  - □ sliding windows

CMSC 417 Spring 2016 Lecture #12 3/21/2016

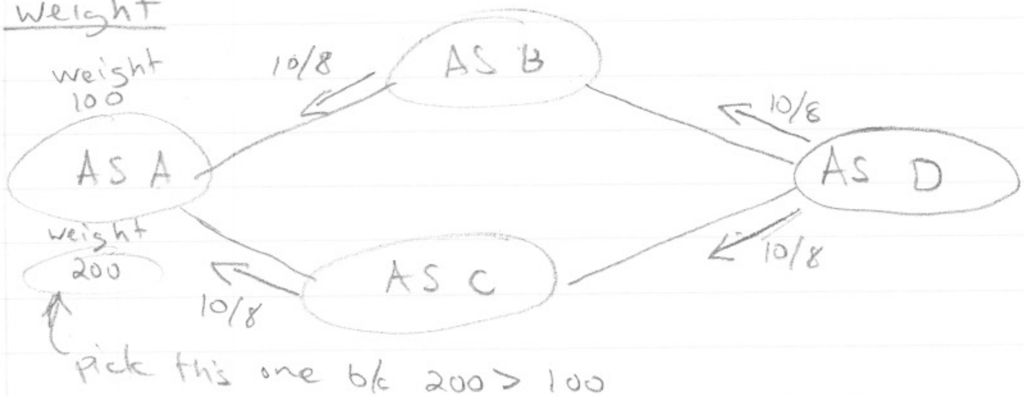
How BGP selects paths

each path has attributes

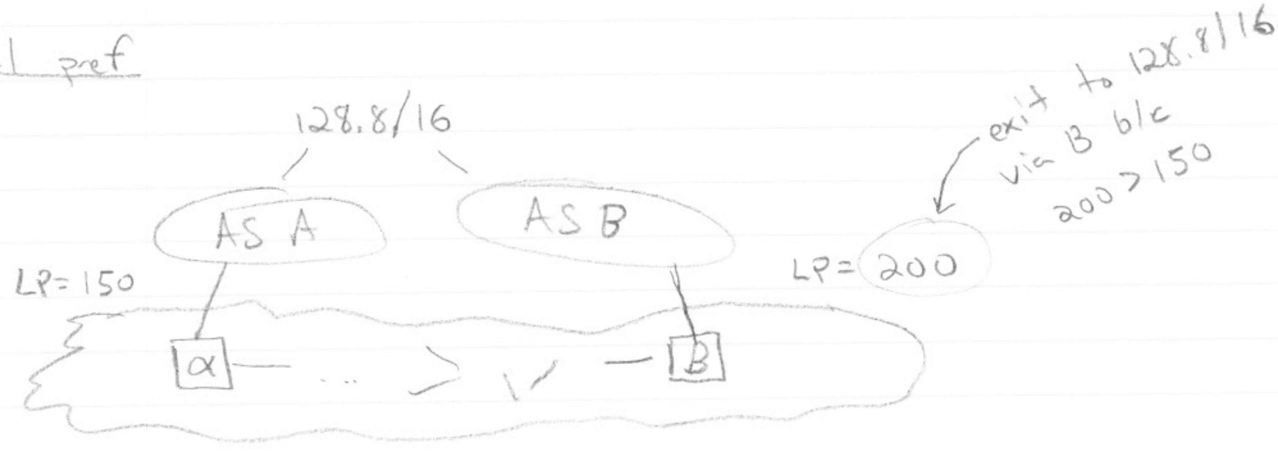
⇒ considered (loosely) in the following order

- □ weight (local to router, does not propagate)
- □ local preference / local pref (local to AS, allows you to prefer routes from certain ASes)
- □ shortest AS path
- □ route origin (EGP over IGP over incomplete)
- □ lowest MED (multi-exit discriminator, lets you tell other ASes how you'd prefer they enter you)
- □ prefer eBGP over iBGP
- □ lowest IGP metric to BGP nexthop
- □ address / ID-based tie-breakers

weight



local pref



CMSC 417 Spring 2016 Lecture #12 3/21/2016

Multi-Exit Discriminator (MED)

⇒ let's you tell others to prefer a way to get to you

⇒ e.g., reach me via terrestrial routes instead of a sattelite link

shortest AS path

⇒ ASes prepend their AS number to the path when they re-advertise

⇒ ASes can prepend themselves multiple times, why would they do this?

route origin

⇒ prefer IGP over EGP over INCOMPLETE

generated locally

learned from outside

gotten from some other source

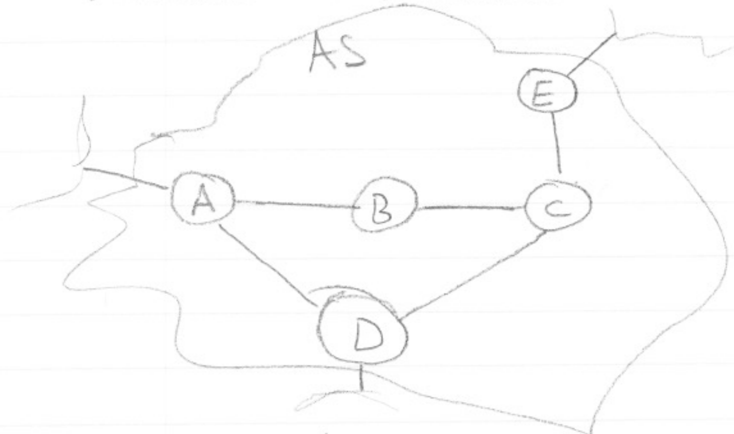
next hop

⇒ needs to be explicity stated b/c the next hop router may not be the BGP speaker

⇒ needs to be reachable using the IGP

CMSC 417 Spring 2016 Lecture #12 3/21/2016

Joining BGP with your IGP



⇒ assume A, D, E are BGP speakers

BGP table for AS

prefix	BGP NH
18.0/16	E
12.0/12	A
128.8/15	D
128.69/16	A

IGP table at B

dst	NH
A	A
C	C
D	C
E	C

⇒ join IGP & BGP to find paths

18.0/16 → C
12.0/12 → A
128.8/15 → C
128.69/16 → A

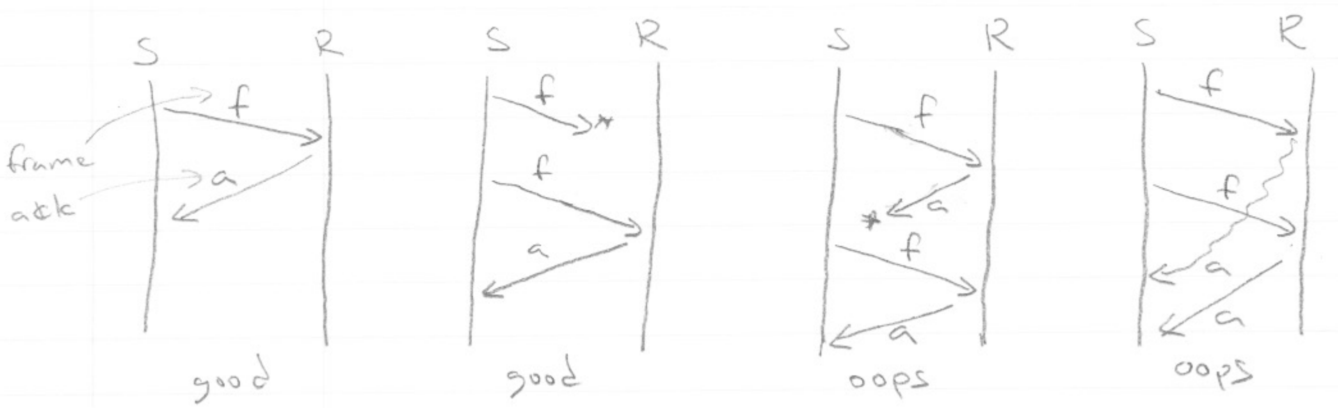
CMSC 417 Spring 2016 Lecture #12 3/21/2016

Reliable Transmission

Two approaches
- ⇒ ARQ (Automatic Repeat reQuest)
- ⇒ FEC (Forward Error Correction)

We'll focus on ARQ

Stop & Wait
- ⇒ send a message
- ⇒ wait for an acknowledgement (ack)
- ⇒ resend after a timeout



- ⇒ problem: you can't tell what frame/message an ack was talking about
  - □ you also might accidentally resend on a lost ack

- ⇒ soln: sequence #s
  - □ label frames/messages w/ sequence #s
  - □ increment by 1 on each
  - □ acks say what seq # they're from
  - □ rolling over seq #s? how many do you need?

CMSC 417 Spring 2016 Lecture #12 3/21/2016

Stop & Wait)

⇒ only one message / frame in flight at a time

⇒ pipe is not full

e.g., 1.5 Mbps link at 50 ms latency RTT

⇒ assuming 1500 byte frames

$$\frac{1500 \text{ bytes}}{50 \text{ ms}} \cdot \frac{1000 \text{ ms}}{s} \cdot \frac{\text{kilobit}}{128 \text{ bytes}} = \sim 235 \text{ Kbps}$$

much less than 1.5 Mbps

bandwidth-delay product matters

latency × BW

sliding windows)

⇒ have a lage space of seq #s for frames

⇒ each frame gets the next seq #

⇒ 3 variables at sender

SWS = sliding window size in seq #s

LAR = last ack received (a seq #)

LFS = last frame sent (a seq #)

invariant $LFS - LAR \leq SWS$

CMSC417 Spring 2016 Lecture #12 3/21/2016

Sliding Window Sizes

⇒ big
- □ allows for handling high latency and/or high bw situations

⇒ small
- □ gentler on the network
- □ gentler on recievers, esp. resource constrained devices
- □ if latency is low (or bw is low) it doesn't hurt you

⇒ no universal SWS
- □ to "fill the pipe" you need

$$SWS \cdot message\_len \geq \underbrace{bandwidth \cdot RTT}_{\text{called bandwidth-delay product}}$$

TCP manages the SWS to meet these needs and a few other things.