Midterm Review
DNS Security

CS144 Review Session 3

October 16, 2009

Saatvik Agarwal

Announcements

- Midterm: next Thursday in class
- Lab 2 due yesterday
- For those of you submitting late, contact us before your deadline if you need an additional extension. Tell us:
  - Where you are
  - How much more time you need

Question:

- Recall that IPv6 does not have a fragmentation header, and that IPv6 requires a 1280 byte MTU. If the link layer MTU is smaller than 1280 bytes, it must provide layer 2 fragmentation and assembly. Assume in a 4-hop network, each link has an MTU of 640 bytes and drops 50% of packets due to bit errors.
- Suppose node A sends a 1280-byte IPv4 datagram to node E. The datagram will become 2 fragments. What is the probability that the datagram will arrive successfully at E?

Answer:

- P(datagram=success) = (1/2)^4 * (1/2)^4 = 1/256

Question:

- In total, in the IPv4 case, what is the expected number of packets all of the nodes will transmit per IPv4 datagram (if the packet drop rate were 0%, then the network would send 2 fragments across 4 links, so 8 packets)?

Answer:

- packets = fragments · (P_{AB} +P_{BC} +P_{CD} + P_{DE}) =
  2 · (1 + ½ + ¼ + 1/8) = 15/4

Question:

- Given the end-to-end delivery rate above, how many fragments does the network expect to transmit to successfully deliver an IPv4 datagram from A to E?

Answer:

- $256 \cdot 7/4 = 448$ fragments

Question:

- Say the network is running IPv6, such that each link fragments and reassembles the packet. Furthermore, the link layer provides reliability, retransmitting lost fragments as needed. How many fragments does the network expect to transmit to successfully deliver an IPv6 datagram from A to E? Do not consider link-layer control packets that provide reliability.

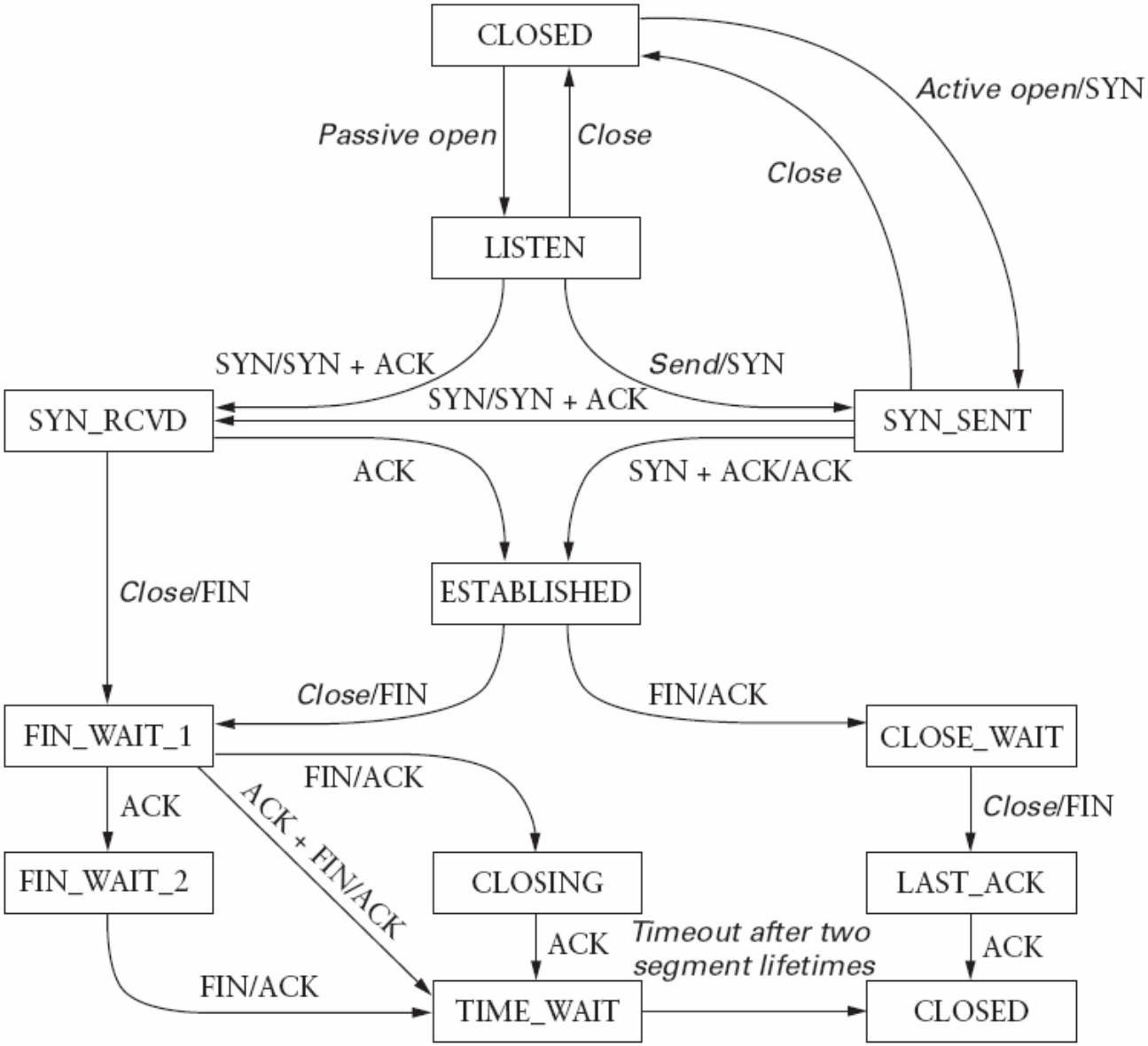
Answer:

- Each link expects to transmit a fragment twice to deliver it. Each datagram is two fagments, and there are 4 links. So 16 fragments.

Question:

- Is it possible for both sides of the TCP connection to end up in the TIME_WAIT state? Give an example when this would occur.

State summary…



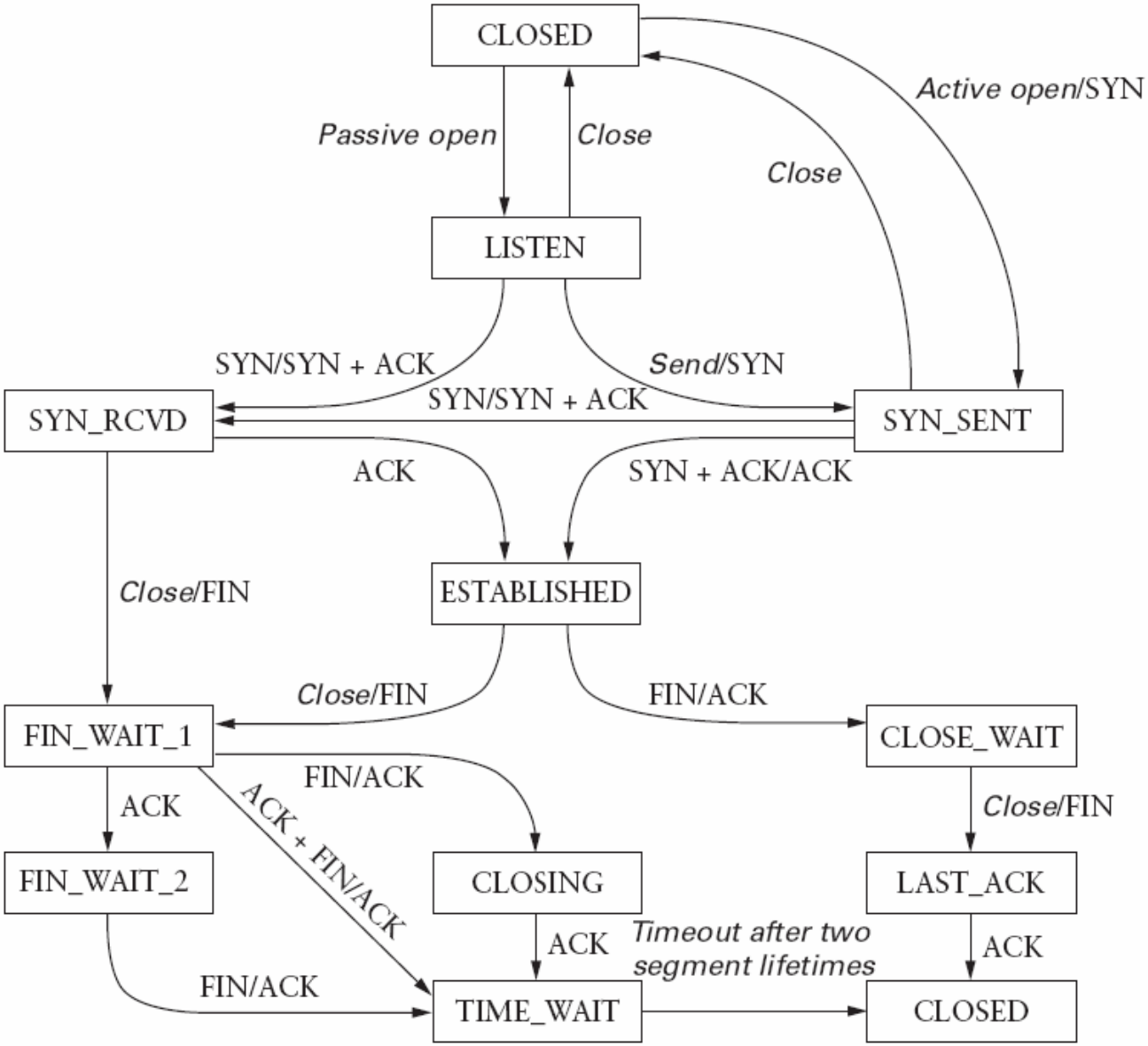
Answer:

- Yes, if both sides call close simultaneously, i.e. before either side receives the other's FIN packet.

Question:

- Suppose you eliminated the time wait state, and instead transitioned directly to the closed state wherever an arrow currently points to time wait. What kind of incorrect behavior might result? Give a concrete scenario in which this might occur.

State summary…



Answer:

- The client and server could establish a new TCP connection with the same IP addresses and port numbers of as the old connection. But packets from the old connection could still be floating around the network, and get interpreted as part of the current connection.

Extra Practice:

- Chapter 4
  - R16, R23, P10, P11, P22, P24, P31

DNS is distributed

- Three possible answers to any question
  - "Here's your answer"
  - "Go away"
  - "I don't know, ask that guy over there"
    - This is delegation. You start with a request, and then get bounced around all over the place.
    - 13 root servers: "www.foo.com? I don't know, go ask the com server, it's at 1.2.3.4"
    - Com server: "www.foo.com? I don't know, go ask the foo.com server, it's at 2.3.4.5"
    - Foo.com server: "www.foo.com? Yeah, that's at 3.4.5.6."
- Dealing with "ask that guy" ("Delegation") a lot of work, so DNS infrastructure divided into Servers (that run around) and Clients, or "Stub Resolvers", that either do or don't get an answer
  - BIND = Name Server
  - Your Desktop = Stub Resolver

Credit: Dan Kaminsky

What about bad guys?

- If everything depends on receiving the right number for the right name, wouldn't a bad guy want his number returned instead?
  - Yup
- So when the name server asks ns1.foo.com for www.foo.com, couldn't the bad guy reply first, with his own number?
  - Yup
- What's supposed to prevent this?
  - Transaction ID – "random" number between 0 and 65535. The real name server knows the number, because it was contained in the request. The bad guy doesn't know – at best, he can guess

Credit: Dan Kaminsky

The Guessing Game

- Good guy – the real name server – has a 65,536 to 1 advantage over the bad guy
  - Those are long odds for the bad guy
- When the good guy gets his reply in – “wins the race” – he can say how long until the next “race”, via something called the TTL, or “Time To Live”
  - 1 minute
  - 1 hour
  - 1 day
  - This is how long a given number is “valid” for a particular name.
- 1 day * 65,536 races / 2 = 84.5 years for 50% chance
  - Good luck on that.

Credit: Dan Kaminsky

First: If it's a race, between who can reply with the correct TXID first, the bad guy has the starter pistol

- Bad guy can force the name server to go run to the good guy and look something up
  - It takes time to get the real request (with random number) to the good guy
  - It takes more time to get the real response back from the good guy
  - It takes no time for the bad guy to immediately follow up a request with a fake response
    - Might have the wrong random number, but it'll definitely arrive first

Credit: Dan Kaminsky

Second, who said the bad guy can only reply once

- Winner of the race is the first person to show up with the correct random number
- Nowhere does it say the bad guy can't try lots of random numbers
  - He has time – he doesn't need to wait for anything to reach him, because nothing ever will
- If the bad guy can reply 100 times before the good guy returns, that 65536 to 1 advantage drops to 655 to 1.
  - Alas…still long odds. And when he loses, he has to wait the TTL. That could be 655 days – almost 2 years!
  - Or maybe not.

Credit: Dan Kaminsky

Finally, the bad guy doesn't actually need to wait to try again.

- If the bad guy asks the name server to look up www.foo.com ten times, there will only be one race with the good guy
  - The first race will be lost (most likely), and then the other nine will be suppressed by the TTL
    - No new races on this name for one more day! Here, use the answer from a while ago
    - So, can we race on other names?
- If the bad guy asks the name server to look up 1.foo.com, 2.foo.com, 3.foo.com, and so on, for ten names, there will be 10 races with the good guy
  - TTL only stops repeated races for the same name!
- Eventually, the bad guy will guess the right TXID before the good guy shows up with it
  - And now…the bad guy is the proud spoofer of … 83.foo.com
  - So? He didn't *want* to poison 83.foo.com. He wanted www.foo.com

Credit: Dan Kaminsky

Bait and Switch

- Is it possible for a bad guy, who has won the race for 83.foo.com, to end up stealing www.foo.com as well?
  - He has three possible replies that can be associated with correctly guessed TXID
  - 1) "Here's your answer for 83.foo.com – it's 6.6.6.6"
  - 2) "I don't know the answer for 83.foo.com."
  - 3) "83.foo.com? I don't know, go ask the www.foo.com server, it's at 6.6.6.6"
    - This has to work – it's just another delegation
      - 13 root servers: "83.foo.com? I don't know, go ask the com server, it's at 1.2.3.4"
      - Com server: "83.foo.com? I don't know, go ask the foo.com server, it's at 2.3.4.5"
      - Foo.com server: "83.foo.com? I don't know, go ask the www.foo.com server, it's at 6.6.6.6"

Credit: Dan Kaminsky

Defenses

a. Increase Query ID size. How?

b. Randomize src port, additional 11 bits

    Now attack takes several hours

c. Ask every DNS query twice:

   – Attacker has to guess QueryID correctly twice (32 bits)

   – Apparently DNS system cannot handle the load

d. DNSSec