Homework 3

Due: 11:59pm, 10 Dec 2018

1. NATs Consider how NATs work: they allow sessions from clients behind the NAT, with private IP addresses, to nodes outside of the NAT. The most common type of NAT maps an internal address/port pair to its external address and an arbitrary port: iIP:iPort -> eIP:ePort. We assume here that the NAT has an external, globally accessible IP address eIP. This mapping is done when the internal node initiates a connection to an external address, such as the globally accessible server S in the figure below. Before the mapping is established, no incoming packets are allowed. Once established, the NAT allows packets sent from S to the NAT's eIP:ePort to be translated back and sent to iIP:iPort. We will consider here a type of NAT that allows such packets from any port in S.

In this question we will try to solve the issue of two servers behind NATs, A and B, to talk to each other, using UDP. (Say you want to build a p2p chat program.)

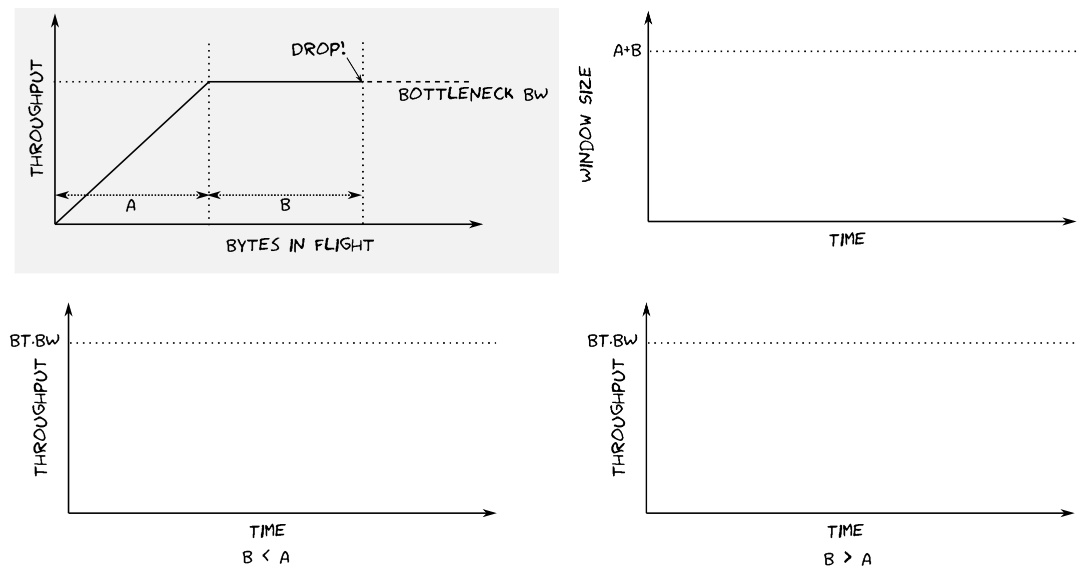
In more details, we make the following assumptions about the NAT: i. Consistent mapping: once a mapping is established (iIP:iPort <-> eIP:ePort), then any outgoing packet coming from iIP:iPort will be translated to eIP:ePort, regardless of the destination. ii. Host-based filtering: A host H sending a packet hIP:any to eIP:ePort will get through to iIP:iPort, *if A has sent a packet from iIP:iPort to H before, regardless of the destination port*. In other words, H can send a packet to eIP:ePort from any local port to H, if A communicated with H before, on any destination port, from iIP:iPort. Otherwise, the NAT drops the packets.[1]

a. Why can't A and B, both being behind NATs, talk to each other by default?

b. Assuming this type of NAT, and the help from a server S, can you sketch a protocol in which A, B, and S take part, such that A and B can eventually communicate using UDP? (Assume that nodes can send arbitrary UDP packets besides the actual communication, with information they have, and from the local ports that they choose. Assume A and B know their names – e.g., their chat ids. Ignore race conditions and timeouts. Lastly, assume you can't predict the port a NAT will use for a new mapping.)

[1]This type of NAT is called a "restricted-cone" NAT. There are both more restrictive and less restrictive NATs, and some in which such communication between A and B is not possible without a relaying server. There is no standard for how NATs should operate. It is also common for there to be multiple levels of NAT, and other cases in which A and B could be behind the same NAT, but we are ignoring these here.

2. TCP Congestion Consider a single TCP RENO, going through a bottleneck link, and the figure below. The shaded graph in the figure, similar to the one we saw in class, shows the throughput as we increase the window size (Bytes in Flight). Assume that a single drop occurs when the window size reaches (A + B) bytes in the figure, and that it is detected by a three duplicate acks.



a. What does A in the shaded graph represent?

b. What does B in the shaded graph represent?

c. Draw a graph of the window size *versus time*, for this connection. Annotate the y axis with any important values in your graph (no need to annotate the x axis).

d. Now draw a graph for the throughput *versus time*, for the case in which B is less than A. (Hint: use the shaded graph, and your knowledge of the evolution of the window, as guides).

e. Draw a similar graph of throughput *versus time*, for the case in which B is greater than A.

3. HTTP/2

a. In HTTP/2, multiple streams can share the same connection. Give two advantages of multiple streams in HTTP/2 over HTTP/1.1 pipelining.

b. With multiple streams, are there still reasons for a client to open several connections to the same HTTP/2 server?

c. HTTP/2 offers the option of server push, in which the server can pro-actively send objects to a client after an initial request. However, it is not trivial to decide when this is beneficial, and which objects to send. Give one potential advantage of using server push for web pages, and one potential disadvantage.

4. RPC Consider the snowcast project. The handout for snowcast defines two types of commands and three types of replies.

a. Choose one of grpc[2], thrift[3], or cap'n'proto[4], and define, using their IDLs, the 5 messages of snowcast. (Note that this is asking you to write the IDL description in text, not to download any of the frameworks and/or compile anything! Also, you may ignore fields in the snowcast messages that loose their functions in the RPC equivalents.)

b. How would using the above RPC system simplify your snowcast code? What would not change?

---

[2] `https://grpc.io/`

[3] `https://github.com/apache/thrift`. If you choose thrift, take a look at the tutorial/tutorial.thrift file in the repo.

[4] `https://capnproto.org/`