Manik Jain
2022MCS2832

# ACN ASSIGNMENT 1

1. Why do you see a difference in webpage fetch times with short and large router buffers?

   The disparity in webpage fetch times between routers with short and large buffers stems from the intricate dynamics of a TCP congestion control. A substantial router buffer size leads to a constant doubling of the congestion window by TCP, prompting a steady influx of packets into the buffer. However, the delay incurred by each packet, typically around 4 millisecond, accumulates in larger buffer scenario, elongating the fetch time. In contrast a router the smaller buffer necessitates TCP to halve its congestion window upon reaching specific limits. Consequently this mechanism accelerates the transmission of webpage fetch request. Essentially the interplay between buffer size congestion window adjustment and packet delay intricately shapes the efficiency of data delivery. In sum, larger router buffers inadvertently amplify delay through sustained packet accumulation, while smaller buffer prompt faster fatty time by forcing quicker congestion window adaptations.

2. Bufferbloat can occur in other places such as your network interface card (NIC). Check the output of ifconfig eth0 on your VirtualBox VM. What is the (maximum) transmit queue length on the network interface reported by ifconfig? For this queue size, if you assume the queue drains at 100Mb/s, what is the maximum time a packet might wait in the queue before it leaves the NIC?

   Bufferbloat, a phenomenon that disrupts network performance, can manifest not only in routers but also in other components such as network interface cards (NICs). Assessing the output of "ifconfig eth0" in a VirtualBox VM offers insights into this. The "txqueuelen" parameter denotes the maximum transmission queue size of the NIC, indicating the upper limit of queued packets awaiting dispatch. For instance, a "txqueuelen" value of 1000 signifies the NIC can queue up to 1000 packets for eventual transmission. Given the standard Ethernet packet size of around 1500 bytes and a queue length of 1000 packets, the queue can accommodate up to 1,500,000 bytes of data.

   Assuming a queue drainage rate of 100 Mb/s, the time a packet might linger in the queue before leaving the NIC is calculated by dividing the total queue capacity in bits (12,000,000 bits) by the transmission rate (100 × 10^6 bits/s), yielding a delay of 0.12 seconds. This computation underscores the potential delay packets face within the NIC's transmit queue, emphasizing the critical role of buffer management in maintaining efficient network performance and averting bufferbloat-related issues.

3. How does the RTT reported by ping vary with the queue size? Describe the relation between the two

   The Round-Trip Time (RTT) measured through a ping command experiences fluctuations based on queue size alterations. RTT encapsulates propagation and queuing delays. Given a fixed propagation delay of 6ms, RTT correlates directly with queuing delay, which is contingent on queue size. Essentially, RTT equates to the sum of propagation delay and queuing delay (queue size * per packet transmission delay). Consequently, enlarging the queue size leads to a commensurate augmentation in RTT. This phenomenon occurs due to the direct proportional relationship between queue size and queuing delay within the RTT equation. By comprehending this connection, network administrators can better manage queue sizes to optimize RTT and overall network performance.

4. Identify and describe two ways to mitigate the bufferbloat problem.

   Bufferbloat, a predicament plaguing computer networks, arises when excessive packet buffering induces elevated latency and subpar network functionality. This problem emerges from the commendable yet misguided effort to avert packet loss during network congestion by deploying capacious buffers in networking tools like routers and switches. Addressing bufferbloat entails the implementation of strategies aimed at forestalling or curtailing the adverse consequences of superfluous buffering in these devices.

   One approach involves the management of queue sizes. This encompasses meticulous configuration and oversight of buffer capacities within networking tools to avert undue buffering and mitigate latency. Prudent adjustment of queue sizes strikes a balance between the imperative of preventing packet loss in congestion scenarios and the need to circumvent the deleterious outcomes of bufferbloat.

   A second method employs Active Queue Management (AQM) algorithms. These algorithms actively oversee and regulate queue lengths within network devices to preclude overfilling during congestion instances. By doing so, they foster diminished and more consistent latency by controlling the delay introduced through buffering. For instance, the Random Early Discard (RED) algorithm employs a probabilistic approach to preemptively drop packets prior to the queue reaching a saturation point. This preemptive packet dropping serves to signal congestion to transmitting devices and averts the accumulation of excessive congestion within the buffer.

   These mitigation strategies offer a means to grapple with the bufferbloat issue, enhancing network efficiency and diminishing latency by rationalizing buffering practices in the face of varying levels of data traffic.

5. Describe how and why your results change when you re-run the emulation.
    a. Dynamic Traffic Patterns: The mutability inherent in traffic patterns is a pivotal factor driving shifts in emulation results. Network traffic rarely maintains a constant state; even subtle changes in the dispersion and timing of incoming data packets can bring about varied queuing dynamics and latency consequences.

    b. Packet Dimensions: Divergence in packet sizes has the potential to influence queuing behavior. Smaller packets often undergo swifter processing, encountering reduced queuing delays. Conversely, larger packets might amplify buffer congestion, culminating in lengthier delays and an escalated propensity for packet drops.

    c. Concurrent Activities: Concurrent operation of diverse applications and devices partaking in network resources can generate undulating levels of load and congestion. These parallel undertakings can introduce unpredictability to network performance, thereby influencing the precision and uniformity of emulation outcomes.

    In essence, the dynamic interplay of these factors underscores the inherent variability in network behavior and its substantial impact on the results of repeated emulations. The intricate orchestration of traffic patterns, packet dimensions, and background activities underscores the intricate nature of network dynamics and the need for comprehensive evaluation to glean accurate insights.