סקירה וניתוח של פרוטוקול QUIC

מטלת גמר ברשתות תקשורת

2024

יוכבד אופשטיין, מור ברגר, לינוי קאופמן, נועה שלום.

חלק יבש

שאלה 1:

נתאר 5 חסרונות ומגבלות של פרוטוקול TCP לפי מה שמתואר במאמר:

1. <u>זיווג בין בקרת העומס (Congestion Control) לבקרת הזרימה (Flow Control):</u>
מטרת בקרת העומס היא לשלוט על כמות החבילות אשר יכולות להיות ברשת. בקרת העומס משנה את גודל החלון לפי מצב הרשת, במצב של איבוד חבילות תהיה הגבלה על גודל החלון עד קבלת החבילות האבודות. TCP משתמש בחלון משותף עבור מנגנון זה וגם עבור מנגנוני בקרת הזרימה שנועדה להבטיח זרימה אמינה. מכיוון ששני המנגנונים משתמשים בחלון משותף יכול להיווצר מצב שבו יש המתנה ארוכה וניצול לא יעיל של הרשת.

2. <u>חסימת ראש התור (HLB):</u>
מכיוון שב-TCP יש מסירה מסודרת של נתונים ע"י העברת נתונים בסדר שבו הם נשלחו, אובדן של חבילה אחת תוביל לעיקוב של קבלת כל החבילות שבאות אחריה בסידור.

3. <u>עיכוב עקב התהליך יצירת קשר:</u>
כדי לייצר קשר בין שני נקודות קצה בעזרת TCP יש צורך בלחיצת יד משולשת. תהליך זה מאוד הכרחי לסדר ואמינות העברת נתונים אך גורמת לעיכוב משמעותי. בנוסף, אם יש הוספת אבטחה בעזרת TLS אז נדרש עוד סבב נוסף שלהחלפת אישורי אבטחה מה שמוסיף עוד RTT.

4. <u>מגבלות של כותרת (Header) קבוע:</u>
ה-Header של TCP מורכבת משדות בגודל קבוע, כולל שדה אפשרויות המוגבל ל-40 בתים.
השדות של ה-Sequence Number ושל ה-ACK בגודל 4 בתים, ושל חלון בקרת הזרימה בגודל 2 בתים, מגבילים את הביצועים של TCP במהירויות רשת גבוהות. ה-Sequence Number וה- ACK מתמלאים מהר מדי, וגודל החלון הקטן מגביל את קצב השידור המרבי של החיבור.

5. <u>מזהה ייחודי (ID) לחיבור וכתובת IP ייחודי:</u>
ה- TCP מזהה את החיבור לפי שילוב של כתובת IP ומספר ה- PORT, אבל כתובת ה-IP או המספר PORT של אחת נקודות הקצה יכול להשתנות במהלך החיבור TCP הקיים ממספר סיבות שונות. לכן כל שינוי של אחד מהנתונים האלו ישבור את החיבור הקיים, מה שיוביל לאיבוד של מידע. בנוסף, על מנת לחדש קשר עם IP חדש או PORT חדש נצטרך לחיצת יד משולשת.

2.2 TCP's Problems

In the following we enumerate the major issues that have been identified from the use of TCP over the years.

1. The coupling between CC and reliable delivery CC's goal is to control the number of packets *inside the network*, and function was latched onto the same window mechanism which was originally designed for receiver to perform flow-control on the sender, and to assure reliable delivery. Therefore any packet loss will constrain the window advancement until the loss is recovered; during the time period of loss detection and data retransmission, the majority, if not all, of the packets within the window may have been delivered to the receiver node, i.e. have exited the network. As shown in Figure 1, where the receiver sets the flow control window size to be equivalent to 8 packets; we can assume CC window is also 8 packets. As packet 1's arrival advances the window to enable the transmission of packets 2–9; the loss of packet 2 stops the window from moving forward, even when no packet inside the network. Therefore the number of unacknowledged packets does not reflect the number of packets inside the network. A few patches of "window inflation, deflation" haven been developed to mitigate this problem, with added complexity and limited effectiveness [3].

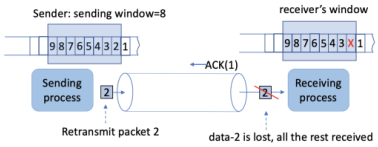

Figure 1: All packets are out of the network; the loss of packet-2 prevents flow window from moving forward.

2. Head of line blocking (HLB): because TCP assures sequential delivery of data byte streams, the loss of one single packet will block the delivery of all subsequent packets until the lost packet has been recovered. Fig. 2 shows an example where packet 2 has been delivered to the receiver but packet 1 is lost during transit. The data of packet 2 cannot be delivered to the application until packet 1 has been retransmitted and received, and passed to the application first, to assure sequential delivery of all data to the application process. Figure 1 and Figure 2 look similar, with the former reflecting an issue at the sender (which is blocked from sending more packets), and the latter an issue at the receiver (already received data gets blocked inside the kernel).

Figure 2: TCP Head of Line Blocking.

3. Delay due to connection setup: Every time an endpoint wants to communicate with another via TCP, a 3-way handshake is required to set up a TCP connection before application data can be sent. And if one wants to secure this connection by TLS, another round trip is needed for the two ends to exchange security credentials.

4. Limitations due to fixed protocol header : TCP header is a collection of *fixed-size* fields. It does have an option field, which is limited to 40 bytes max. In addition to carry application data between the two communicating ends, the TCP protocol design also packs all the control functions into the same 20-byte TCP header: connection setup, tear down, and reset; and data acknowledgment. When additional functions are identified, e.g., Selective ACK, they are added into the option field which has its own length limit of 40-byte maximum.
Three specific fields in the TCP header have been directly affected by the continued increase of network speed over time. The sequence number and ACK fields each are 4-byte long, and the flow control window size is 2-byte only. These small, fixed size fields limit TCP's performance at high network speed: the first two wrap around too quickly, making them no longer unique; and a small size of flow control window directly limits a TCP connection's throughput which is upper bound by ($window\ size \times RTT$). Again the TCP header option has been used to extend their lengths.

5. Unique connection identifier and IP address The IP address of either end host in a TCP connection may change during the connection's lifetime, due to a variety of reasons: host multihoming, mobility, or running behind an NAT. Because TCP uses the combination of two endpoints' IP addresses and port numbers as the connection identifier, any change in either IP address would break the existing connection, resulting in all the data exchanged up to that point being thrown away. To recover from the failed TCP connection, a new 3-way handshake is required to set up a new connection.

שאלה 2:

נתאר 5 תפקידים שפרוטוקול תעבורה צריך למלא:

1. הגדרת המזהה הייחודי של הקשר

2. ניהול החיבור של התעבורה:
   א. מזהה החיבור ייחודי גלובלי שמקשר בין שני קצוות החיבור, חייב למפות את הכתובות IP בצורה אמינה.
   ב. הקמת מצב חיבור (Connection State) ופירוקו בסוף הקשר
   ג. בקרת החילוף מידע בין שני הקצוות. נרצה שיהיה גמיש.
   ד. תמיכה בשינוי כתובת IP.

3. אמינות של העברת החבילות:
   א. מזהה נתונים ייחודי שמועבר בצורה אמינה לקצה השני.
   ב. בקרת זרימה בחלון. נרצה להימנע מחסימת ראש התור (HLB).

4. בקרת עומס על הרשת:
   א. נרצה לשלוט על כמות החבילות שנמצאים ברשת בכל רגע נתון.

5. אבטחת המידע:
   א. הצפנת ואבטחת המידע שעובר ברשת. (TLL מספקים רק סודיות נתונים)

2.5 Summary: Transport Protocol Functions over IP

Based on the observation of the transport protocol designs and their usages over the last few decades, below we make a summary of the basic functions that a unicast transport protocol running over IP needs to provide and the related design questions. Note that all the existing transport protocols use port numbers for data demultiplexing inside a host in a straight forward way, thus we omit it in the following enumeration.

1. defining connection identifier and data identifier

- **Globally unique connection ID** that binds the two ends of the connection, wishes to be IP address independent but must map to IP addresses reliably.
- connection state setup and tear down
- **Control information exchange** between the two ends; wish to have flexibility in defining new control messages.
- support of host IP address changes (can be due to host multihoming, physical mobility, or other causes)

2. transport connection management

3. reliable data delivery

- **Unique data identifier** that must be reliably exchanged with the other end.
- **Window flow control for reliable delivery**; wish to avoid the head-of-line blocking problem.

4. Congestion control: [4] needs to control the number of packets inside the network.

5. Security To be more specific, people currently view encrypted connections as *network security*, although TLL encrypted channels provide only data confidentiality; remote party authenticity and trust are managed through third-party certificate authorities (CAs).

שאלה 3:

בשאלה זו נתאר את אופן פתיחת הקשר ב QUIC והיכן הוא משפר את חלק מהחסרונות של TCP:

אופן פתיח הקשר בפרוטוקול QUIC:

פרוטוקול QUIC משלב את תהליך היצירת קשר הראשוני עם אימות של מפתחות הצפנה כדי להשיג את המידע הנדרש לשני התהליכים בRTT אחד.

QUIC משתמש בשליחת החבילה התחלתית כדי לתאם את המזהה של הקשר (CID) כל קצה תורם את החלק שלו בשדות של המזהה. בנוסף, ה TLS 1.3 גם כן נמצא בחבילה ההתחלתית, וכך אפשר לקבוע את הפרמטרים של ההצפנה .

כך הוא משפר את הלחיצת יד של TCP אשר שם נדרשים שני סבבי RTT, הראשון עבור יצירת הקשר הראשוני והעברת פרמטרים תחבורתיים, והשני לתהליך האבטחה.

בנוסף, המזהה קשר הייחודי (CID) מורכב משני מספרים שכל צד בוחר אחד מהם. מספר זה לא תלוי בPORT ולכן שינוי בפרמטרים בשכבה יותר נמוכה לא משפיע על הCID, ואפשר לשמור על החיבור במקרה כזה, בשונה מ TCP.

שיפור נוסף שך QUIC, הוא שאפשר לשלוח מידע מוצפן ב RTT-0 ואם לקצה השני כבר קיימים הנתונים של ההצפנה, הוא יוכל להמשיך את הקשר הקודם ללא צורך ביצירת קשר ראשוני.

3.1 Connection ID

QUIC uses the combination of two numbers, one selected by each end, to form a pair of connection IDs. The connection ID (CID) acts as a unique identifier for the connection, which is used to ensure that changes in addressing at lower protocol layers will not cause packets to be delivered to a wrong recipient. By using the connection ID, QUIC supports the demultiplexing ability similar to the functionality provided by TCP.

Related RFC: QUIC-TRANSPORT[15] §5 Connections.

3.2 QUIC Handshake

QUIC combines transport and cryptographic handshakes together, acquiring the information necessary for both in 1-RTT. More specifically, this entails doing an authenticated TLS 1.3 key exchange along with an authenticated transport parameters exchange at the same time. This minimizes the latency necessary to set up a secured connection. Whereas conventional TCP keeps security and transport parameter exchanges separate, requiring at least 2 RTTs to set up a secure connection.

QUIC uses the Initial packet to negotiate the connection IDs for a new connection. Each endpoint will populate the Source Connection ID field with its chosen value in its Initial packet and that ID will be used by the other endpoint to set the Destination Connection ID when sending future packets. Upon receiving an Initial packet, a server can optionally choose to verify a client's address by sending a Retry packet containing a random token, which should be repeated by the client in a new Initial packet to continue the handshake process. TLS 1.3 handshakes messages are also embedded in these Initial packets, which could establish a shared secret to protect the confidentiality and authenticity of future packets in 1-RTT. The chosen connection IDs will be included in the QUIC transport parameters, which will be authenticated during the TLS handshake process. With the negotiation of connection IDs, QUIC supports the reliable setup of a new connection similar to the functionality provided by TCP.

QUIC allows a client to send 0-RTT encrypted application data in its first packet to the server by reusing the negotiated parameters from a previous connection and a TLS 1.3 pre-shared key (PSK) identity issued by the server, though these 0-RTT data are not protected against replay attack. By supporting sending 0-RTT data, QUIC is also able to handle use cases where T/TCP is required. More detail about the cryptographic part of the handshake process is discussed in §6.1.

שאלה 4:

בשאלה זו נתאר את מבנה של חבילהת QUIC, וכיצד הוא משפר חסרונות של TCP.

בניגוד ל TCP שבו פורמט החבילה קבוע (ובפרט ה HEADER קבוע), ל- QUIC יש שני סוגים שונים של כותרות Headers: חבילות להקמת החיבור מכילים HEADER בפורמט הארוך, אך לאחר שהחיבור הוקם נשתמש בפורמט הקצר.

ה HEADER מורכב מ:

א. מזהה החיבור (מזהה החיבור המקור והיעד נבחרים בכל נקודת קצה)
ב. מספר חבילה
ג. דגלים (מידע על סוג החבילה ומצבה)
ד. תוכן מוצפן (Protected Payload)

איך הוא משפר חסרונות ב TCP:

- חסימת ראש תור (HLB): מבנה החבילה מאפשר לשלוח חביוות בסדר לא רציף, בשונה מ TCP.
- שינוי כתובת IP : הקישור בין נקודות הקצה הוא בעזרת Connection ID (CID) וכמו שפירטנו בסעיף הקודם, מאפשר שינוי של ה IP וה PORT בשונה מה TCP

4.1 Packet Format

Unlike TCP where the packet header format is fixed, QUIC has two types of packet headers. QUIC packets for connection establishment need to contain several pieces of information, it uses the long header format. Once a connection is established, only certain header fields are necessary, the subsequent packets use the short header format for efficiency [13]. The short header format that is used after the handshake is completed is demonstrated in Fig. 4. In each packet, one or more frames can be embedded in it and each frame does not need to be of the same type as long as it is within the MTU limit.

Each packet in a QUIC connection is assigned a unique packet number. This number increases monotonically, indicating the transmission order of packets and is decoupled from loss recovery [5]. Therefore, it can be used to tell easily and accurately about how many packets may be inside the network, as compared to TCP congestion control which shares the same flow control window used for reliability.

QUIC receiver ACKs the largest packet number ever received, together with selective ACK (ACKing all received packets below it, coded in continuous packet number ranges) as shown in Figure 5. The use of purposely defined ACK frames can support up to 256 ACK blocks in one ACK frame, as compared to TCP's 3 SACK ranges due to TCP option field size limit. This allows QUIC to ACK received packets repeatedly in multiple ACK frames, leading to higher resiliency against packet reordering and losses. When a QUIC packet is ACKed, it indicates all the frames carried in that packet have been received.

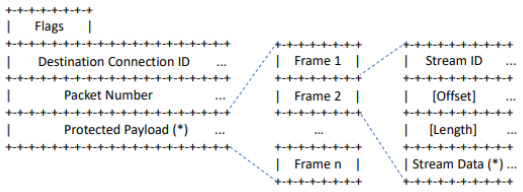

Figure 4: QUIC Short Header Packet Format. Frame 2 is a Stream frame containing application data.

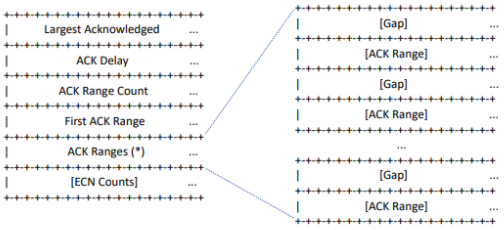

Figure 5: QUIC ACK Frame Format. The Largest Acknowledged field indicates the largest packet number the sender is acknowledging. An ACK Range indicates the number of continuously acknowledged packets before the largest acknowledged packet number. The gap indicates the number of continuously unacknowledged packets before each ACK Range

שאלה 5:

מה QUIC עושה כאשר חבילות מגיעות באיחור או לא מגיעות כלל?

QUIC משתמש במספר מנגנונים כאשר חבילות מגיעות באיחור או לא מגיעות בכלל.
המנגנון הראשון הוא בקרת עומס. הוא עוזר לפרוטוקול להתאים את קצב השליחה בהתאם לתנאים של הרשת. רק אם יש חבילות שמגיעות באיחור באופן קבוע, אז זה מעיד על עומס קיים ויגרום לQUIC להוריד את הקצב כדי למנוע איבוד של חבילות נוספות.

> To avoid unnecessary congestion window reduction, QUIC does not collapse the congestion window unless it detects *persistent congestion*. When two packets requiring acknowledgment are declared to be lost, persistent congestion will be established if none of the packets sent between them is acknowledged, an RTT sample existed before they were sent and the difference between their sent time exceeds the *persistent congestion duration* calculated based on the average RTT (`smoothed_rtt`), the deviation of the RTT samples (`rttvar`) and the maximum time the receiver might delay sending the acknowledgment.

מנגנון נוסף הוא שימוש בשידורים עצמיים, וזה מאפשר לו להימנע מחסימות שנגרמות כתוצאה מהמתנה לשיחזור חבילות שנאבדו. רק השידורים של החבילות שנאבדו יצטרכו לחכות לשידור חוזר וזה לא יחסום את שאר החבילות מלהתקדם.

> Since QUIC uses multiple independent streams, it avoids the Head-of-Line Blocking problem caused by waiting for recovering lost packets in TCP. When a packet is lost, only the streams with data frames contained in the packet will need to wait for the retransmission of the lost frames. It will not block other streams from moving forward. For instance, in Fig. 6 there are three QUIC streams denoted in red, green, and blue for a single connection. In case there is a packet loss for the red stream, it will not block the delivery of packets for the green stream and the blue stream.

איתור ושיקום אובדן של חבילות. לכל חבילה QUIC משתמש בשתי שיטות על מנת לקבוע אם החבילה נאבדה. הראשונה היא מבוססת על מספר חבילה: מספר הרצף קטן מהמספר של החבילה המוכרת. השיטה השנייה מבוססת על זמן: החבילה נשלחה לפחות מספר פעמים בזמנים מסויימים של המקסימום לרשת המשוערת הנוכחית RTT וה-RTT שנדגם אחרון לפני החבילה שנשלחה.

לאחר שQUIC איתר את החבילה שנאבדה, הוא ינסה לשחזר אותה. המסגרות שנאבדו מוכנסות לחבילות חדשות, שיהיו להם מספרי חבילות חדשים - שונים מהמספרים של החבילות שנאבדו.

> **Loss Recovery:** After a loss has been detected, the lost frames are then put into new outgoing packets (which will be assigned new packet numbers, unrelated to the lost packets).
> With loss detection and recovery, QUIC supports the reliable ordered byte-stream delivery similar to the functionality provided by TCP.
> Related RFC: QUIC-RECOVERY[14] §6 Loss Detection.

edged, and when a certain threshold is met. QUIC uses two types of thresholds for determining whether an earlier sent packet is lost, (i) packet number based: the in-flight packet's sequence number is smaller than the acknowledged packet by a certain number. For instance, assuming the largest acknowledged packet number is x and the packet reordering threshold is t, then all in-flight packets with a packet number smaller than $x - t$ will be declared lost. (ii) time-based: the in-flight packet was sent at least certain times of the maximum of the current estimated network RTT and the latest sampled RTT before the acknowledged packet. For

שאלה 6:

נתאר מהו בקרת העומס (CC) של QUIC:

QUIC משתמש במספרי החבילות לבקרת העומס וב OFFSET לבקרה על אמינות.

בדומה ל TCP, QUIC משתמש במנגנון בקרת עומס מבוסס על גודל חלון המגביל את מספר הבייטים שהשולח יכול להעביר. חשוב לציין, QUIC לא מפתח אלגוריתם העברה או משתמש באלגוריתם ספציפי, אלא שולח אותות לבקרה על העומס והשולח יכול ליישם את מנגנון בקרת העומס שלו.

על מנת להימנע מהקטנת חלון מיותרת, QUIC לא מצמצם את החלון עבור חבילה אחת שנאבדה, אלא מחכה לקלוט עומס מתמשך ורק אז מצמצם את החלון. עומס מתמשך ייקבע אם לפחות שתי חבילות שדורשות ACK נאבדו ולא קיבלנו ACK על החבילות ביניהם.

בנוסף, QUIC שולח חבילות באופן מדורג על מנת להפחית את הסיכוי לעומס על ידי שליחת חבילות במרווח זמן לפי החישוב שלו.

5.2 Congestion Control

Decoupling of congestion control from reliability control: QUIC uses packet numbers for congestion control, and stream frame offset for reliability control.

Incorporating existing algorithms: Similar to TCP congestion control, QUIC utilizes a window-based congestion control scheme that limits the maximum number of bytes the sender might have in transit at any time. QUIC does not aim to develop its own new congestion control *algorithms*, nor use any specific one (e.g., Cubic). QUIC provides generic signals for congestion control, and the sender is free to implement its own congestion control mechanisms. A congestion control algorithm documented in the QUIC standard is described in appendix A.

To avoid unnecessary congestion window reduction, QUIC does not collapse the congestion window unless it detects *persistent congestion*. When two packets requiring acknowledgment are declared to be lost, persistent congestion will be established if none of the packets sent between them is acknowledged, an RTT sample existed before they were sent and the difference between their sent time exceeds the *persistent congestion duration* calculated based on the average RTT (`smoothed_rtt`), the deviation of the RTT samples (`rttvar`) and the maximum time the receiver might delay sending the acknowledgment.

A QUIC sender will pace its sending to reduce the chances of causing short-term congestion by ensuring its inter-packet sending interval exceeds a limit calculated based on the average RTT (`smoothed_rtt`), the congestion window size, and the packet size.

Related RFC: QUIC-RECOVERY[14] §7 Congestion Control.