Bayero Journal of Pure and Applied Sciences, 16(1): 36 - 45
Received: April, 2023
Accepted: June, 2023
ISSN 2006 – 6996

TRANSPORT PROTOCOLS: A SURVEY OF DIFFERENT SCHEMES FOR INTERNET CONGESTION CONTROL

Bisu, A. A.
Department of Physics Bayero University, Kano-Nigeria
Email: aabisu.elt@buk.edu.ng

ABSTRACT
Transmission Control Protocol and User Datagram Protocol (UDP) are the key network standards and technologies used over Internet Protocols (IP) for the global Internet communication. The idea of Internet ubiquitous assumes that an Internet access should be available even in remote rural locations lacking network infrastructure. In this case satellite access represents an attractive solution. Nevertheless, experience shows that over satellite and wireless links, Transmission control Protocol (TCP) is limited in terms of capacity underutilisation due to high bandwidth-delay product (BDP). Several enhanced versions were proposed as solutions. For instance, tuning TCP parameters have been proposed to avoid the underutilization of high-capacity and wireless links with high BDP. These topics have been often addressed, but considering recent high speed TCP variants, the evolution of end users' habits, and recently proposed satellite link access scheme, a new study is necessary to reconsider some preconceptions and previous recommendations in such a context. This paper review some of the TCP variants and a survey commonly proposed enhanced solutions for standard TCP and User datagram Protocol (UDP) over the Internet Protocol (IP).
Keywords: Protocol, Algorithm, Transmission, Congestion, Data, TCP, IP, Transport, ACK, packet, UDP

INTRODUCTION

Transmission Control Protocol and Internet Protocol (TCP/IP) suites are the network standards and technologies for the global Internet developed and founded by U.S Department of Defence (DoD) under the Defence Advanced Research Projects Agency (DARPA) (Postel, 1981a; Postel, 1981b; Comer, 2006). TCP is an end-to-end (E2E) connection-oriented protocol which accepts data from a data stream submitted by the application layer, segments it into chunks, and adds a TCP header creating a protocol data unit named TCP segment. This helped to specify how computers communicate, set conventions for interconnecting networks and forwarding traffic (Postel, 1981a; Postel, 1981b; Comer, 2006). As the Internet evolve and grow with new unique technologies, services and applications, the protocols also require enhancements to accommodate new changes for efficient and effective performance of data communications, especially over high-capacity, and long-distance networks. Protocols such as TCP/IP provide the syntactic and semantic rules for communication and form the basis for the Internet data communications by over 80−90% of billions connected machines and things nowadays (Bisu, 2021). Standard TCP carries the details of the message formats, how machines respond to message reception in the form of acknowledgement (ACK) to ensure reliability, and specifies how errors, link congestions and other abnormalities can be handled. TCP allows computer communication to be discussed independently of a vendor's network hardware, while the IP was designed for interconnected systems of packet-switched communication networks that transmit blocks of data known as datagrams or packets from sources (Tx) to destinations (Rx) hosts of fixed length IP addresses, and if necessary, fragment and reassemble long datagrams for transmission via small packet networks (Postel, 1981a; Postel, 1981b; Comer, 2006; Cerf, & Kahn, 1974). However, User Datagram Protocol (UDP) was designed to make available a datagram mode of packet-switched communication within the Internet environment. Like TCP, UDP assumes that the Internet Protocol (IP) is used as the underlying protocol. This provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism.

UDP is transaction oriented in which delivery and duplicate protection is not guaranteed (Postel, 1980). Applications requiring ordered reliable delivery of streams of data should employ the TCP.

The Internet communication utilise these transport protocols to deliver and control data transmission. As mentioned, TCP and UDP are the most popular transport protocols for data communications over the Internet Protocol (IP) nowadays. TCP aims at enforcing E2E reliability using a flow control (sliding window) and loss detection algorithm based on the expiration of a timer used by the sender. Sliding window size is fixed according to two processes in most TCP versions. At the beginning, the receiver specifies each TCP segment sent to ACK data received, the amount of additional data it is willing to buffer for the connection. This value is known as the receive window (*rwnd*). To establish the state of the different links and intermediate systems between the sender and the receiver, TCP maintains another variable on the sender side known as congestion window (*cwnd*). The value of *cwnd* increases when a new ACK indicates new data received, and it decreases when data loss is detected, for instance when a timer expires. Finally, the amount of data that can be sent before a new ACK, known generally as *flight size*, is the minimum between these two variables, *rwnd* and *cwnd*.

Since the first version of TCP in the 1980s (Postel, 1981a), a lot of improvements have been proposed and implemented as shown in figure 1.0. These proposed enhancements are mainly motivated by the necessity to ensure the protocols match with the properties of new and emerging networks links, particularly with respect to capacity and delay that led to the definition of several schemes (Pirovano & Garcia, 2013). Subsequent sections and subsection of this paper discussed the different schemes and mechanisms of the TCP and UDP transport protocols. These enhancements and the timeline showing a chronological view of main variants are shown in Figure 1.0 (Pirovano & Garcia, 2013).

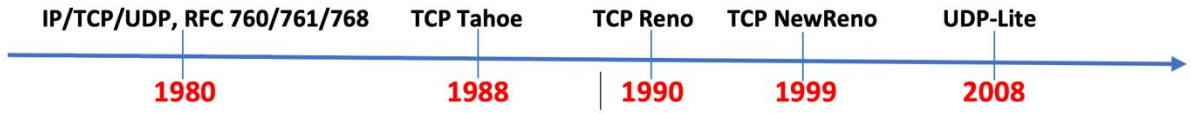

Figure 1.0: TCP and UDP schemes Timeline

METHODOLOGY

This survey presents a standard TCP and UDP schemes and mechanisms for data communications over the Internet Protocol (IP) as the major and widely used congestion control implemented by the global Internet (Fairhurst, Trammell, & Kuehlewind, 2017). The focus on TCP and UDP is due to their wider application and relevance in the Internet for data packets flow and congestion controls. This paper reviewed both the standard TCP and UDP schemes as well as their enhancements through other proposed variants. Moreover, modified, and enhanced schemes such as Tahoe, Reno, NewReno, and UDP-Lite that aimed to address performance issues in different network environments, particularly a more realistic heterogeneous communications network environment characterised with high Bandwidth-Delay Products (BDP) and transaction-oriented Internet applications and services were also discussed.

Standard Transmission Control Protocol

The original TCP was designed based on (Cerf & Kahn, 1974), as a highly reliable E2E protocol to support information and resource sharing in different packet-switched networks. This provides packet sizes, transmission failures, sequencing, flow control, E2E error checking, and the creation/destruction of logical process-to-process connections (Postel, 1981a; Cerf & Kahn, 1974). The high E2E reliability of TCP enhances robustness in the presence of unreliable data communication and improves the availability in the presence of congestion in packet-switched communication networks and interconnected systems of these networks (Postel, 1981a). This connection-oriented protocol is intended to fit into layered hierarchical protocols that support host-to-host (E2E) packet switching network applications reliably as an acknowledgement (ACK) based protocol. This (ACK) determines the success or failure of data transmission, and strongly impacted by the time delay (latency) in accepting, delivering, and transporting the data. Therefore, there is a need for careful development of timing procedures to achieve successful data delivery with improved performance in different network environments (Caini, Firrincieli & Lacamera, 2009; Finch, Sullivan & Ivancic, 2010).

Ideally, TCP should be able to operate over a wide spectrum of communication systems from hard-wired connections to packet-switched or circuit-switched networks and use basic IP to send and receive variable-length segments of data encapsulated in envelopes of Internet datagrams (Postel, 1981a). The IP datagram provides a way for addressing the TCP connection's source and destination, and the frag- mentation/reassembly of segments needed to accomplish transport and delivery via different multiple networks and interconnected gateway systems. Security, precedence, and TCP segment compartmentalisation is encapsulated within the IP packet (datagram) for E2E communication of information over multiple networks (Postel, 1981a; Postel, 1981b). In the layering of the Open System Interconnection (OSI) model of the TCP/IP suite, TCP interfaces with high-level user application (layer) processes and lower-level IP layer. Establishing a TCP connection starts with the Three-Way Handshake (3WH) procedure as shown in Figure 2.0. This is initiated by the TCP source and replied to by the destination node or initiated simultaneously by two TCP connections where each of the two nodes receives a Synchronise (SYN) segment that carries no ACK. The arrival of an old duplicate SYN segment helps the receiver note that a simultaneous connection initiation is in progress. This is by sending and receiving three segments carrying SYN, ACKs and Sequence Number (*Seqn*) data (Postel, 1981a; Postel, 1981b).

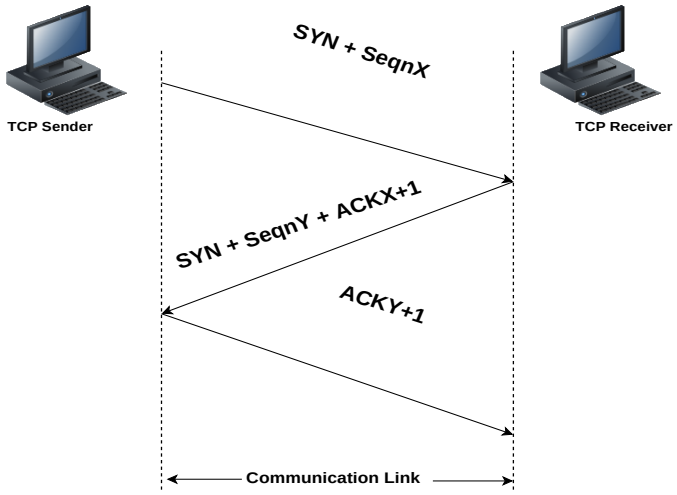

Figure 2.0: TCP Three-Way Handshake (3WH) Procedure

The high reliability of TCP allows it to recover from lost, duplicated, damaged, and out of order data delivered over the Internet. This is accomplished by using a sequence number (used to correct segment ordering, to eliminate duplicates segments and for flow control) and ACK (use for data retransmission within the timeout interval) from the receiver while damaged data are found using a checksum on each transmitted segment. However, standard TCP assumed wired networks like Local Area Networks (LAN) or large ARPANET networks based on packet switching technology (Postel, 1981a). This assumption limits the performance of standard TCP in other network environments such as wireless and SatCom networks (Kim *et al.*, 2015; Border *et al.*, 2001; Park *et al.*, 2011) as will be discussed in the following sections.

TCP Schemes

The first and most famous enhanced versions of TCP are Tahoe, NewReno and Vegas (Jacobson, Braden, & Borman, 1992; Jacobson & Braden, 1988). These versions and the most recent ones mainly differ in the way they manage the congestion window size and the events or indicators that trigger its updates. The first one has been proposed by (Jacobson, 1988) and introduced the 'fast retransmit' algorithm. Prior to this enhancement, TCP only used a time out based segment loss detection mechanism (i.e. the expiration of a pending timer managed by the sender), Tahoe version enhanced TCP with another way to detect losses based on duplicate acknowledgements, dupAcks. A TCP sender numbers the flow of bytes it receives from the application layer and the receiver periodically acknowledges the data it receives by sending back the number of the expected byte, the last byte number of the first continuous flow of data received incremented by one. Tahoe assumes that in the event of duplicate acknowledgement, it must retransmit the corresponding data even if the timer has not yet expired for these data.

In cases of timer expiration or duplicate acknowledgements, Tahoe assumes that the network is congested, and hence decreases its congestion window size to one Maximum Segment Size (MSS), a sender network property defining the maximum size of a TCP segment. In TCP Reno version, Jacobson proposed to differentiate these two events in the sense that if timer expiration is symptomatic of a serious problem that justify a drastic decrease of *cwnd* and hence of the *flight size*, dupAcks revealed a loss followed by the reception of new segments, that is an isolated segment loss. That is why in the second case TCP Reno applies a 'fast recovery' algorithm that consists in dividing the *cwnd* size by two (Stevens, 1997; Ho *et al.*, 2008).

During a nominal behaviour without losses, these TCP versions increment the *cwnd* size by two mechanisms; Slow Start (SS) and Congestion Control (CC) (Bisu, 2021). If the *flight size* is less than a threshold value (*ssthresh*), they use SS and increase *cwnd* by *cwnd = cwnd + 1 MSS* for each acknowledged segment. If the current *flight size* is greater than *ssthresh,* during a new phase called Congestion Avoidance (CA) the increment is given by *cwnd = cwnd + (1 MSS/cwnd).* The value of *ssthresh* is half the maximum *flight size* reached before the last loss detection or equals to *rwnd* for new connections. NewReno enhance Reno in the sense that after a fast retransmit, the sender stays in fast recovery state until the original window has been entirely acknowledged. Doing so, improves performances in the case of multiple losses in a same window. In the old version, standards TCP end-to-end congestion control mechanisms is known as additive increase multiplicative decrease (AIMD) because the TCP sending rate is controlled by a *cwnd* that is halved for every loss, and increased by one packet per window of data acknowledged (Postel, 1981a; Postel, 1981b).

In the middle of the 1990s, researchers at the University of Arizona introduced TCP Vegas offering a new congestion avoidance algorithm (Brakmo et al., 1994). Rather than *losses*, this algorithm uses the observed *packet delay* to determine the rate at which to send its segments. This method depends heavily on an accurate calculation of the Round-Trip Time (*RTT*) value which represents the time elapsed between the sending of a segment and the reception of the acknowledgement by the sender. TCP Vegas assumes that under some conditions, an increase in RTT reveals network congestion (Pirovano & Garcia, 2013).

TCP/IP and Internet technologies are evolving and growing exponentially. With new proposals emerged and old are being revised to cope with the demands and dynamics of communications technologies (Pirovano & Garcia, 2013). The key demand is connectivity that brings additional traffic; new Internet uses bring new applications and dynamics in traffic patterns. Internet data traffic is expected to grow dramatically with the deployment of 5G networks; new applications and requirements are also expected which requires the use of different network and communications technologies. Therefore, TCP implementations need to be revisited and redesigned for better performance and to accommodate future heterogeneous networks and communications technologies (Dubois *et al.*, 2010; Kim *et al.*, 2015; Border et al., 2001).

TCP is the most widely used transport protocol on the Internet nowadays, and accounts for 80−90% of the Internet data traffic (Bisu, 2021). This has caught the attention of many researchers investigating the performance of TCP over different communications channels and heterogeneous network environments (Trivedi *et al*, 2010; Caini, Firrincieli & Lacamera, 2009). This has resulted in different TCP variants being proposed; here we review three key standard TCP variants schemes, which include Tahoe, Reno, and New Reno because of their relevance to how the standard TCP evolved.

Tahoe

Tahoe is the first standard TCP version to implement the congestion control algorithm proposed by (Jacobson, 1988), which is based on the *cwnd* regulating the number of segments sent (transmission rate) over the network, and the TCP sender estimation of losses as a mechanism for controlling congestion (Braun *et al*, 2010) . The *cwnd* increment in *Tahoe* follows the SS exponential increment phase and CA linear increment phases.

TCP *Tahoe* uses *Go-back-N* error-recovery and timeout loss detection and recovery mechanisms. These mechanisms are inefficient and suffer from quite a few drawbacks including the limitation of *Go-back-N* error-recovery that it can only retransmit packets already received by the receiver because of cumulated *ACKs* usage. Another drawback is the loss detection and recovery mechanism using a timer triggered for every packet and remains active until the reception of a corresponding *ACK* or *FRet*. Therefore, at every packet loss, it waits for timeout and the pipeline to be emptied; this is costly in high Bandwidth-Delay Product (*BDP)* links. After loss detection, *Tahoe* goes back to SS phase with a value *ssthresh (γ) = cwnd/2* to recover from the loss.

Mechanisms were proposed to overcome these drawbacks; selective repeat mechanism solved the limitations of error-recovery, while *FRec.* and *FRet.* mechanism improved loss detection and recovery issues in *Tahoe* (Braun *et al*, 2010). These and many other enhancements were implemented in different TCP versions such as *Reno* (Pirovano & Garcia, 2013).

Reno

TCP *Reno* is an improved version of *Tahoe*, which includes a modification of the *FRet.* algorithm to integrate *FRec.* algorithm and employs selective repeat mechanism for the packet loss recovery (Braun *et al*, 2010). These enhancements bring some level of intelligence in *Reno* that helps detect packet loss earlier while the pipeline (buffer/link) is not emptied. The modification of *FRet*. to incorporate *FRec.* halves the *cwnd* (set *cwnd* and *ssthresh* to $cwnd_{loss}/2$) without going back to SS phase. During the *FRet.* period, *cwnd* is incremented by the number of *dupACKs* not the usual *ACKs*. The TCP sender using *Reno* needs an immediate *ACK* whenever a segment is received, and the sender then assumes that receiving a *dupACK* indicates the next segment in the sequence has been delayed in the network and some segments arrived at the receiver *out-of-order* or lost due to network congestion. Thus, when the sender receives three *dupACKs*, it assumes the segment was lost and activates the *FRet.* mechanism so that the segment is re-transmitted before the *timeout timer* expires and the pipe is still almost full. After the packet loss, returning to the SS algorithm phase like in *Tahoe*, it empties the pipe and that is the drawback solved by integrating *FRet., FRec.* and *selective repeat* mechanisms instead of returning to the SS algorithm.

Nonetheless, the *Reno* version also has some limitations that required improvements for better performance. These include successive *FRet.* or the false *FRet.* followed by a *false-recovery* problems. Another drawback is performance degradation in the presence of multiple packets drops within the same transmission window or single RTT. This makes *Reno* performance similar to *Tahoe* under high packet losses since only one packet loss can be detected. TCP versions such as NewReno modifications were proposed and implemented to solve these limitations.

NewReno

TCP *NewReno* scheme is a proposed enhancement to the behaviour of *Reno* that limits performance at the event of *multiple packets* loss in one window. *NewReno* modification was made in *FRec.* algorithm using partial *ACKs* (*pACKs*) to indicate multiple losses in one window (Braun *et al*, 2010). This modification and changes that followed have been described in (Floyd, Henderson & Gurtov, 1999; Henderson *et al*., 2012) and found to be adequately efficient for wired networks with low to moderate BDP, but inefficient with poor performance over high BDP and high link error wireless network environments. *NewRen*o performs poorly in wireless network environment because it assumes and interprets packet loss because of network congestion (Braun *et al*, 2010; Henderson *et al*., 2012). These losses could be due to wireless link error or bad transmission due to corruption. In the absence of *SACK*, the sender has little implicit information available to make retransmission decisions during *FRec.*, thus *NewReno* algorithm responds to *pACKs,* which are *ACKs* that cover new data, but not all the data outstanding when the loss was detected (Henderson *et al*., 2012). In the *NewReno FRet/FRec* modification, the sender can infer, from the received *dupACKs,* whether multiple losses in one window of data likely occurred and avoid retransmission *time-out* or making multiple *cwnd* reductions due to such event. The *NewReno* applies to the *FRec.* mechanism, starting when three *dupACKs* arrive at the TCP sender and finishes when either retransmission *time-out* occurs or *ACK* that acknowledges all the data including the outstanding data when the *FRec.* Started (Floyd, Henderson & Gurtov, 1999; Henderson *et al*., 2012).

NewReno attempted to solve the problem of multiple packet losses in one window by responding to *pACKs* in the absence of *SACK* due to either the *SACK* option is not locally supported or TCP connection at the other end is unwilling to use the explicit *SACK* option (Floyd, Henderson & Gurtov, 1999; Henderson *et al*., 2012). Moreover, *NewReno* performance becomes degraded over high BDP and wireless network environments such as SatCom and hybrid ISTNs environments. This poor performance over network environments motivated new and improved TCP schemes.

User Datagram Protocol Schemes

User Datagram Protocol (UDP) is another important data transport protocol used by the Internet (IP) nowadays. This makes available a datagram mode of packet-switched communication in interconnected communication environments. UDP provides a procedure for applications to exchange messages with other programs using minimum protocol mechanisms. Unlike TCP, this protocol is transaction oriented; reliable delivery and duplicate protection for congestion control and loss recovery are not guaranteed.

Thus, UDP is not best choice for applications and services that require ordered reliable delivery and congestion/loss control of data streams (Postel, 1980).

Standard UDP is the pioneer *connectionless* protocol designed to maintained message boundaries, with no connection setup or feature negotiation (Postel, 1980; Fairhurst, Trammell & Kuehlewind, 2017). This protocol employs independent messages referred to as *datagrams* and offers minimum transport service using *non-guaranteed datagram* delivery (Braden, 1989; Postel, 1980; Fairhurst, Trammell & Kuehlewind, 2017). Also allows direct access to the *datagram* service of the IP layer by the applications that do not require the level of guaranteed service of TCP. UDP is the best choice for applications that require the use of communications service such as multicast or broadcast delivery that may not be offered by TCP (Braden, 1989; Postel, 1980; Fairhurst, Trammell & Kuehlewind, 2017). Standard UDP is nearly a null protocol, which provides *checksumming* of data and *multiplexing* using port number over the IP. Therefore, an application program running over UDP must deal directly with *E2E* communications issues such as *retransmission* for reliable delivery, *flow control*, *congestion avoidance*, *packetization* and *reassembly* that a connection-oriented protocol like TCP would have offered (Braden, 1989). However, complex coupling between IP and TCP will be reflected in the coupling between UDP and most applications utilising it. Well-known ports are used by UDP to exchange datagram and follow the same rule as TCP well-known ports, For instance, when datagram with UDP port address arrived and there is no pending *LISTEN* call, UDP sends an *Internet Control Message Protocol (ICMP)* port unreachable message. UDP must pass any IP option received from the IP layer transparently to the application layer, an application must be able to specify IP options to be sent in its UDP datagrams, which must be pass to the IP layer (Braden, 1989; Postel, 1980). The transmission of data in UDP is carried out by encapsulating each datagram into a single IP packet or several IP packets fragments. This allows a datagram to be larger than the effective path Maximum Transmission Unit (MTU), the fragments are reassembled before delivery to the UDP receiver to make it transparent to the user of the transport service (Fairhurst, Trammell & Kuehlewind, 2017). Larger messages may be sent without fragmentation when *jumbo grams* are supported. The standard UDP header format consists of fields of 16-bit each that include, *source port*, *destination port, message length, checksum* and *data octets* or *payload* as shown in figure. 3.0 (Postel, 1980).

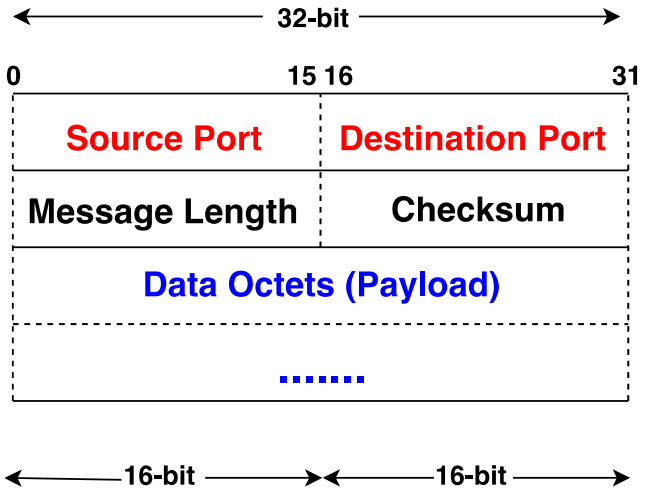

Figure 3.0: Standard UDP Header Format (Postel, 1980; Fairhurst, Trammell & Kuehlewind, 2017).

Standard UDP does not have the capabilities for the providing congestion control, flow control or error correction, but has capabilities for *datagrams broadcast, multicast, unicast* and *any-cast* (Fairhurst, Trammell & Kuehlewind, 2017). The protocol is widely used by applications that do not require guaranteed data delivery and retransmissions of lost data such as Domain Name Services (DNS) and streaming services. UDP detects *datagrams/payload errors* as well as delivery of packets to an unintended destination. This leads to discard of received datagrams without notifying the sender (source) of the service (Fairhurst, Trammell & Kuehlewind, 2017).

Lack of flow control in UDP results in missing messages by a receiving application that is not able to run sufficiently fast, or frequently. This lack of congestion control handling may cause UDP traffic to experience loss when utilising an overloaded path and result in loss messages from other protocols like TCP when utilising the same network path (Fairhurst, Trammell & Kuehlewind, 2017) .

Lightweight User Datagram Protocol

The Lightweight User Datagram Protocol (UDP-Lite) is a new protocol *semantically* identical to the standard UDP (Postel, 1980; Fairhurst, Trammell & Kuehlewind, 2017). UDP-Lite is useful for the applications and programs in error-prone network environments where *partially damaged payloads* are preferably delivered rather than discarded by the network. UDP-Lite is *semantically* like UDP when partially damaged data payloads preferred to be delivered instead of discarded as in standard UDP (Larzon *et al.*, 2004). UDP-Lite is based on *three observations* about the behaviour of standard UDP algorithm as highlighted below (Larzon *et al.*, 2004):

1. Certain classes of applications like *audio* and *video codecs* benefit from having damaged datagrams delivered rather than discarded by the network. These *codec* applications include speech codec, the Internet Low Bit Rate Codec (ILBRC), error resilient codecs (H.263+ and H.264) and MPEG-4 video codecs. These codecs application programs were designed to better handle errors in the data payload than loose the entire packets (Larzon, Degermark & Pink, 1999a; Larzon, Degermark & Pink, 1999b; Larzon *et al.*, 2004).
2. IP data transportation links employs a strong link layer integrity (error checking) like Cyclic Redundancy Check 32 (CRC-32) that must be used by default by any IP traffic (Larzon *et al.*, 2004). Traffic using UDP-Lite may benefit from a unique link behaviour that allows *partially damaged IP packets* to be *forwarded* when requested and the under-lying support through link layer *error checking* (Larzon *et al.*, 2004). Several radio technologies operating at a point where cost and delay are sufficiently low, and error-prone links are aware of error sensitive part of the packet support this unique behaviour of UDP-Lite. The physical link has a chance of providing high protection that reduce the likelihood of corruption of the error sensitive bytes by using unequal *Forward Error Correction (FEC)* (Larzon *et al.*, 2004).
3. The intermediate layers like *transport* and *IP layers* should not impede *error-tolerant* applications from flowing well in the presence of *error-prone* links. IP layer header has no *checksum* that blankets the IP data payload, thus IP layer protocol is not the major concern, and normally available transport layer protocol best fit for these kinds of applications is the UDP that does not require overhead for *retransmissions* of *erroneous/loss packets*, in-order delivery, or error correction (Larzon *et al.*, 2004).

The key difference between standard UDP and UDP-Lite, is the checksum with optional partial coverage (Checksum Coverage filed) shown in Figure 4.0. This replaced the UDP's message length field as in Figure 3.0, the message *source* divides the *packet* into a *sensitive part* covered by the *checksum*, and *insensitive part* that is *not covered* by the *checksum* (Larzon, Degermark & Pink, 1999b; Larzon *et al.*, 2004; Stanislaus, Fairhurst & Radzik, 2004).

.

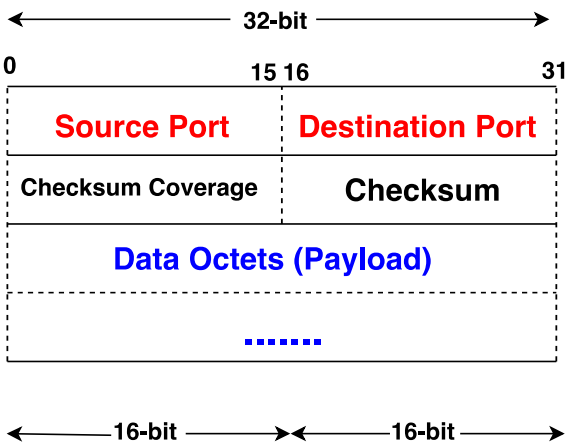

Figure 4.0: UDP-Lite Header Format (Larzon, Degermark & Pink, 1999a; Larzon *et al.*, 2004).

When UDP-Lite option is used, *errors* in the *insensitive part* of the *packet* will not result in the packet being discarded by the transport layer at the receiver side. While UDP-Lite behave semantically like standard UDP when *errors* in the *sensitive part* (*packet* covered by the *checksum*) are normally set as default (Larzon *et al*., 2004). Therefore, UDP-Lite sender's *partial checksum* provides additional flexibility compared to standard UDP (Larzon, Degermark & Pink, 1999a; Larzon, Degermark & Pink, 1999b; Larzon *et al*., 2004), particularly for applications that classify *payload* as *partially insensitive* to *bit errors* (Larzon *et al*., 2004).

However, both UDP-Lite and standard UDP are unreliable transport (*connectionless*) protocols without *congestion* and *flow control* mechanisms available, they utilise identical set of port numbers assigned by the Internet Assigned Numbers Association (IANA) for use by the UDP. Thus, any application that requires *reliable E2E* connection-oriented transport protocol should employ TCP schemes and mechanisms.

DISCUSSIONS AND ANALYSIS

The standard TCP schemes discussed earlier are heavily *RTT* (delay) dependent and designed to assume *packet losses* are due to the congestion on the link. These *RTT* dependence and attributed that *loss events* are due link congestion. This led to performance degradation and disparity when using standard TCP over heterogeneous networks. Heterogeneous networks such as satellites and terrestrial wireless links are characterised with high latency (*RTT*), high capacity, high bandwidth and high link errors that can affect the performance of standard TCP (International Telecommunication Union, 2009; International Telecommunication Union, 2010). Capacity utilisation, fairness, scalability, and stability are some of the performances that can be degraded using standard TCP scheme (Bisu, 2021).

TCP schemes such as BIC, CUBIC, and HYBLA were proposed to mitigate the performance disparity due to the high RTT and/or high bandwidth network links such as satellite and wireless links (Allman, Glover & Sanchez, 1999; Allman *et al.,* 2000; Ghani & Dixit, 1999). However, each of these schemes considered the performance improvement of large BDP networks based on either high RTT or high-capacity networks. Pure satellite and ISTN links have characteristics of high RTT, high capacity, and high link errors.

Looking at the different transport protocol schemes and mechanisms discussed, a more efficient transport protocol is required (Larzon, Degermark & Pink, 1999b). This should match the properties of link layers and applications as mentioned, and the error-control mechanisms of the transport layer must provide protection to significant information like headers, but optionally ignore errors that are best handled by the applications.

CONCLUSION

The two most famous and widely used transport protocols over the global Internet are the TCP and UDP schemes. TCP is an E2E and connection-oriented that guaranteed delivery of data packets while UDP is transaction-oriented (connectionless) in which *congestion* and *flow control* mechanisms are not available. The key difference between these two protocols is E2E reliability (TCP) and unreliability (UDP) of delivery data packets. However, the standard versions of these protocols have their shortcomings in terms of how congestion and packet losses/drops are controlled in different network environments. Several proposals were made to improve the standard versions and performance, particularly for high BDP and wireless links such as satellites and heterogeneous network settings. This paper conducted a survey of the most famous enhancements of the standard TCP and UDP schemes. The enhanced versions discussed in this paper were effective for high-capacity terrestrial and wireless links. However, these enhanced schemes do not solve the performance problems over satellite link with high BDP. There are proposals that are made specifically for high BDP links such as Geostationary Earth Orbit (GEO) satellite links, which is beyond the scope of this article. Standard transport protocols and their enhanced versions are vital to the effective functioning of the global Internet. Therefore, there is need to continue to research in this area and make the protocols more robust to accommodate the dynamics and growth of the Internet.

REFERENCE:

Allman, M. *et al.* (2000). Ongoing TCP research related to satellites. *Internet Society Network Working Group*, BCP. 28, RFC, 2760.

Allman, M., Glover, D., & Sanchez, L. (1999). Enhancing TCP over satellite channels using standard mechanisms. *The Internet Society; IETF,* RFC 2488 ,.

Bem, D. J., Wieckowski, T. W., & Zielinski, R. J. (2000). Broadband satellite systems. *IEEE Communications Surveys & Tutorials*, 3(1), 2-15.

Bisu, A. A. (2021) A Review of Standard Transmission Control Protocol for Data Communications. *Bayero Journal of Physics and Mathematical Sciences,* 12(1),129 – 142.

Border, J. *et al*. (2001). Performance enhancing proxies intended to mitigate link related degradation. *The Internet Society Network Working Group*, RFC 3135.

Braden, R. (Editor). (1989). Requirements for Internet hosts–communication layers. *The Internet Engineering Task Force,* RFC 1122.

Brakmo, L. S., O'Malley, S. W., & Peterson, L. L. (1994, August 31 - September 02). TCP Vegas: New techniques for congestion detection and avoidance. *Proceedings of the Conference on Communications Architectures, Protocols and Applications,* 24-35, London, United Kingdom.

Braun, T., Diaz, M., Enriquez-Gabeiras, J., & Staub, T. (2010). End-to-End quality of service over heterogeneous networks. In Enhanced Tansport Protocols (pp. 111-129). *Springer*.

Caini, C., & Firrincieli, R. (2004). TCP HYBLA: A TCP enhancement for heterogeneous networks. *John Wiley International Journal of Satellite Communications and Networking*, 22, 547-566.

Caini, C., & Firrincieli, R. (2006). "End-to-End TCP Enhancements Performance on Satellite Links. *IEEE Symposium on Computers and Communications,* 1031-1036.

Caini, C., Firrincieli, R., Lacamera, D. (2009). Comparative performance evaluation of TCP variants on satellite environments. *IEEE International Conference on Communications*.

Caini, C., Firrincieli, R., Lacamera, D., de Cola, T., Marchese, M., Marcondes, C., Sanadidi, M. Y., & Gerla, M. (2009). Analysis of TCP live experiments on a real GEO satellite testbed. *Elsevier Performance Evaluation*, 66(6), 287-300.

Cerf, V., & Kahn, R. A. (1974). Protocol for packet network intercommunication. *IEEE Transactions on Communications,* 22(5), 637-648.

Comer, D. E. (2006). Internetworking with TCP/IP: Principles, protocol, and architecture, (5th ed., Vol. 1). *Pearson Prentice Hall*.

Dubois, E., Fasson, J., Donny, C., & Chaput, E. (2010) Enhancing TCP based communications in mobile satellite scenarios: TCP PEPs issues and solutions. *IEEE Advanced Satellite Multimedia Systems Conference and Signal Processing for Space Communications Workshop*.

Fairhurst, G., Trammell, B., & Kuehlewind, M. (2017). Services Provided by IETF Transport Protocols and Congestion Control Mechanisms. *Internet Engineering Task Force*, RFC 8095.

Farserotu, J., & Prasad, R. (2000). A Survey of future broadband multimedia satellite systems, issues and trends. *IEEE Communications Magazine*, *Broadband Direct to Home/User Wireless Systems*, 128-133.

Finch, P. E., Sullivan, D. V., & Ivancic, W. D. (2010). An evaluation of protocol enhancing proxies and modern file transport protocols for geostationary satellite communication. *IEEE Aerospace Conference*.

Floyd, S., & Henderson, T. (1999). The NewReno modification to TCP's fast recovery algorithm. *Network Working Group, Experimental,* RFC 2582.

Floyd, S., Henderson, T., & Gurtov, A. (2004). The NewReno modification to TCP's fast recovery algorithm. *Network Working Group, Standards Track,* RFC 3782.

Ghani, N., & Dixit S. (1999). TCP/IP enhancement for satellite networks. *IEEE Communications Magazine*, 64-72.

Ha, S., Rhee, I., & Xu, L. (2008). CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review - Research and Developments in the Linux Kernel*, 42(5), 64-74.

Henderson, T., Floyd, S., Gurtov, A., & Nishida, Y. (2012).The NewReno modification to TCP's fast recovery algorithm. *Internet Engineering Task Force, Standards Track,* RFC 6582.

Ho, C-Y., Chen, Y., Chan, Y., & Ho, C. (2008). Fast retransmit and fast recovery schemes of transport protocols: A survey and taxonomy. *Elsevier Computer Networks,* 52, 1308-1327.

International Telecommunication Union. (2009). Transmission control protocol over satellite networks. ITU-R S.2148,1-24.

International Telecommunication Union. (2010). Performance enhancements of transmission control protocol over satellite networks. ITU-R S.1711-1, 1-50.

Jacobson, V. (1988). Congestion avoidance and control. *ACM SIG-COMM and Computer Communication Review*, 18(4), 314-329.

Jacobson, V., & Braden, R. (1988). TCP extensions for long-delay paths. *Network Working Group,* RFC 1072.

Jacobson, V., Braden, R., & Borman, D. (1992). TCP extension for high performance. *Internet Society Network Working Group*, RFC 1323.

Kim, Y., Park, H., Kim, J., & Choi, Y. (2015). Fairness PEP solution for satellite TCP. *IEEE International Conference on Ubiquitous and Future Networks,* 83-85.

Kota, S., & Marchese, M. (2003). Quality of service for satellite IP networks: A Survey. *John Wiley International Journal of Satellite Communications and Networking*, 21, 303-349.

Larzon, L-A., Degermark, M., & Pink, S. (1999). Efficient use of wireless bandwidth for multimedia applications. *IEEE International Workshop on Mobile Multimedia Communications,*187-193.

Larzon, L-A., Degermark, M., & Pink, S. (1999). UDP Lite for real time multimedia applications. *Hewlett Parkard*.

Larzon, L-A., Degermark, M., Pink, S., Jonsson, L., & Fairhurst, G. (2004). The lightweight user datagram protocol. *Internet Society Network Working Group,* RFC 3828.

Marchese, M. (2007). QoS over heterogeneous networks. *John Wiley & Sons*.

Mathis, M., Mahdavi, J., S. Floyd, & Romanow, A.(1996).TCP selective acknowledgement options. *Network Working Group, Standard Track,* RFC 2018.

Park, M., Shin, M., Oh, D., Kim, B., & Lee, J. (2011, Jun 20-23). TCP HYBLA+: Making TCP more robust against packet loss in satellite networks. *Springer International Conference on Computational Science and its Applications,* Spain, 424-435.

Paxson, V., & Allman, M. (2000). Computing TCP's retransmission timer. *The Internet Society Network Working Group, Standards Track,* RFC 2988.

Paxson, V., Allman, Chu, M. J. & Sargent, M. (2011). Computing TCP's re-transmission Timer. *The Internet Engineering Task Force, Standards Track,* RFC 6298.

Pirovano, A., & Garcia, F. (2013). A new survey on improving TCP performances over geostationary satellite Link. *Network and Communication Technologies*, *Canadian Center of Science and Education,* 2(1), 1-18.

Postel, J. (1981a). Transmission control protocol. *DARPA Internet Program Protocol Specification*, RFC 793.

Postel, J. (1981b). Internet protocol. *DARPA Internet Program Protocol Specification*, RFC 791.

Postel, J.(1980) User Datagram Protocol. *DARPA Internet Program Protocol Specification,* RFC 768.

Ramakrishnan, K., Floyd, S., & Black, D. (2001). The addition of explicit congestion notification to IP. *Network Working Group, Standard Track,* RFC 3168.

Rhee, I., *et al.* (2018). CUBIC for fast long-distance networks. *Internet Engineering Task Force,* RFC 8312.

Stadler, J. S., & Gelman, J. (1999). Performance enhancement for TCP/IP on a satellite channel. *IEEE Military Communications Conference*, 1, 270-276.

Stanislaus, W., Fairhurst, G., & Radzik, J. (2004) Cross layer techniques for flexible transport protocol using UDP-Lite over satellite network. *Internet Society Network Working Group,* RFC 3828.

Stevens, W. (1997). TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. *Network Working Group, Standard Track,* RFC 2001.

Trivedi, S., Jaiswal, S., Kumar, R., & Rao, R. (2010, December 15-17). Comparative performance evaluation of TCP HYBLA and TCP cubic for satellite communication under low error conditions. *IEEE International Conference on Internet Multimedia Services Architecture and Application,* Bangalore, India.

Xu, L., Harfoush, K., & Rhee, I. (2004). Binary increase congestion control for fast long-distance networks. *IEEE INFOCOM*, 4, 2514-2524.