A Transmission Controlled Tunnel for Datagrams

Sameer Jain, Viren Sinai Nadkarni, Pranav Prem, Nagraj Vernekar
Computer Engineering Department,
Goa College of Engineering, Farmagudi, Ponda- Goa
samir4nov@gmail.com, viren.nadkarni@gmail.com, pranavprem93@gmail.com, nkv_2447@yahoo.com

***Abstract*—Conventionally a TCP network setup is considered better than a UDP network setup because TCP offers connection establishment and acknowledgements ensuring delivery of every packet. However in certain situations a UDP connection outperforms a TCP connection such as that of real time data transfer in the scenario of a VoIP examined in this paper. This paper establishes that the choice between a TCP and a UDP connection is application specific and then provides a means of securing UDP transmission by creating a hybrid tunnel that uses connection establishment principles of a TCP network and transmitting UDP packets over established tunnel with enforced encryption at the application layer itself and no retransmission or ordering. This combines some of the security and integrity features of TCP with the speed of UDP.**

***Keywords*—User Datagram Protocol (UDP), Transmission Control Protocol (TCP), Datagrams, Packets, Tunneling**

I. INTRODUCTION

UDP (User Datagram Protocol) is a communications protocol that offers a limited amount of service when messages are exchanged between computers in a network that uses the Internet Protocol (IP) to get packets of data (called datagrams in this case) from source to destination [5]. It provides only a single service over IP which is a port number to distinguish between user requests. UDP doesn't provide sequencing of the packets that the data arrives in. Neither does it provide any handshaking or connection establishment. While UDP is at transport layer in the OSI schema, it becomes the responsibility of the application at application layer to make sure comprehensible data is received. TCP (Transmission Control Protocol) is a set of communications protocol used along with the Internet Protocol(IP) to send data in the form of message units between computers over the Internet [6]. TCP divides the message into packets and provides the services (to IP) of ensuring the delivery of every packet by using connection establishment and handshaking signals to acknowledge the delivery of every packet. TCP works in the transport layer and additionally maintains the order of receiving packets at the destination. It establishes a connection and maintains it until such time that the messages to be exchanged by the application programs have been exchanged.

From this it is fairly evident why TCP is considered a standard while UDP is used for more trivial tasks that require less security. This misconception is stressed on in many cases such as the usage of UDP in a file transfer protocol that is labeled Trivial File Transfer Protocol (TFTP). TFTP is extremely limited and provides no authentication. It is known for its simplicity.

With the increase of bandwidth and the revolution of hand-held devices, real time media transfer has increased significantly over the last few years. This shift has led to focus returning to UDP transmissions for communication.

In early April 2014, a critical bug that manifested in SSL/TLS implementation was discovered. The bug was named "Heartbleed". The bug was a buffer underflow issue that came from 'heartbeat' signals that are sent in SSL/TLS to check if a server is alive. Each 'heartbeat' involves data to be returned and the size of data to be returned if server is alive. The server is to read the size and return the requested data of that size if it is alive. The problem arises when the size of the data requested to be returned is larger than the data supplied to be returned. According to SSL/TLS mechanism, the server continues sending data from the input buffer until the amount of data asked for is provided. This data could contain session data of other people logged into the server causing major breach of security [2].

This problem is a clear example of how security that spans through more than one layer of the OSI architecture is prone to serious flaws because of inter-layer co-ordination. Hence, the paper proposes a system that has all its security implemented in the application layer itself. All packets are encrypted in all other layers as well as shielded by the tunnel which maintains the connection and makes sure there is no leak of information.

The paper is organized as follows; the second section is an analysis of existing tunneling technologies, their merits and demerits. Following that is an analysis to show why and by how is a UDP channel better than a TCP channel in case of real time applications. The fourth section shows how the paper proposes to secure a UDP connection and combine some essential features of both TCP and UDP using a hybrid tunnel. The section after that shows how this hybrid tunnel is different from the existing tunneling mechanisms. The final section is about the scope of implementing such architecture to secure a UDP channel.

II. Existing Tunneling Technology

A. TOR Network

TOR (The Onion Router) protocol is an internet security protocol that is used for complete anonymity over the World Wide Web. To connect to the TOR network, a TOR circuit is established with relays between a source and destination. In TOR routing, the intermediate relays themselves do not know the entire circuit. Only the source and destination are privy of that information. The data is encrypted at the source multiple times and passed into the relay circuit. Each relay in the circuit decrypts only the header to find out the address of the next relay in the circuit and then forwards the packet onwards with one less layer of security. The final exchange will involve plain text exchange between the last relay in the circuit and the destination. To avoid this, data is secured with SSL / TLS before entering the TOR circuit [1].

The TOR network uses a tunneling proxy to serve as the TOR tunnel and to maintain anonymity. SOCKS is the Internet Protocol used by TOR to maintain anonymity. It manages exchange of data between source and destination by routing data through a proxy server. SOCKS server proxies TCP connection to an arbitrary IP address and provides a means for UDP packets to be forwarded.

B. VPN tunnel

VPN (Virtual Private Network) is the extension of a private network that includes links across shared or public networks such as the internet. VPN helps send packets from source to destination over a network in a way that it emulates a point to point link. This is done by creating a tunnel between source and destination.

In case of Windows VPN, Point to Point Tunneling Protocol (PPTP) is used to tunnel packets over the public network. Tunnel is established with both of the tunnel endpoints negotiating the configuration variables such as address assignments, encryption and comparison parameters. After the tunnel is established, data is sent. The tunnel uses a tunnel data transfer protocol to prepare the data for transfer. This involves appending tunnel data transfer protocol headers or encryption.

PPTP encapsulates Point to Point Protocol frames into IP datagrams for transmission over an IP based network [4]. PPTP is described in RFC 2637 in IETF Database. It uses a TCP connection known as the PPTP control connection to create, maintain and terminate the tunnel. PPTP uses a modified version of Generic Routing Encapsulation (GRE) to encapsulate PPP frames as tunneled data. The payloads of the encapsulated frames can be encrypted, compressed or both.

Additionally, L2TP (Layer Two Tunneling Protocol) is a combination of PPTP and Layer 2 Forwarding, a technology developed by Cisco systems. L2TP encapsulated PPP frames to be sent over IP, X.25, frame relay or ATM networks when sent over an IP network, L2TP frames are encapsulated as User Datagram Protocol (UDP) messages [3]. L2TP can be used as a tunneling protocol over internet or over private intranets.

L2TP uses UDP messages over IP networks for both tunnel maintenance and Tunneled data. This creates the issue that L2TP tunnel maintenance and tunnel data have the same structure.

III. UDP Preferred Scenario

To establish the performance difference between a UDP connection and a TCP system the specific case of a VoIP system is considered. In such a case, there is a real time exchange of Voice data which implies a steady stream of information is needed more than assured delivery of data. Hence typically a UDP connection should be preferred.

To analyze the given condition a prototype capable of carrying out basic VoIP function is developed in Python which uses PyAudio to capture audio and then transfers it from a server machine to a client machine. It is capable of switching between UDP and TCP modes of transmission. The Microsoft Network Emulator for Windows toolkit is used once a channel was established to simulate various real life packet loss scenarios.

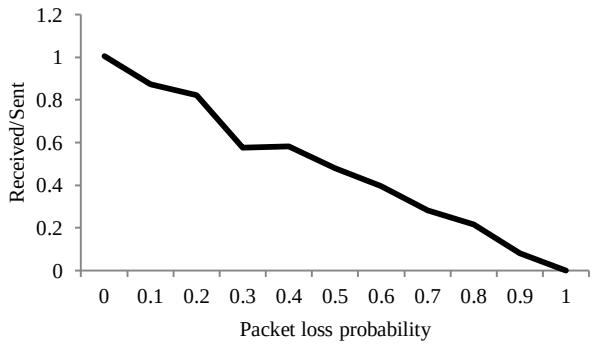

Fig. 1Packet loss ratio at different network conditions

The prototype is tested using the network tool to obtain the table and graph with various packet loss condition simulations in one way traffic for UDP and TCP. It is immediately seen that while the packet loss percentage was increased, the average delay time in case of TCP connection runs into seconds and then into minutes while the latency of a UDP connection remains almost unaffected. At the same time, the amount of packets lost in case of UDP is seen to be considerable.

The effect of the phenomenon as seen in the graphs on the audio transmitted is profound. In case of TCP, there is a significant amount of latency and stutter whereas in case of UDP there is a steady stream of audio which keeps getting more difficult to recognize with increasing loss percentages. At

highloss percentages it while it is difficult to recognize the audio track with UDP, it is impossible to do so with TCP which played pieces of sounds at intervals of up to a couple of minutes.

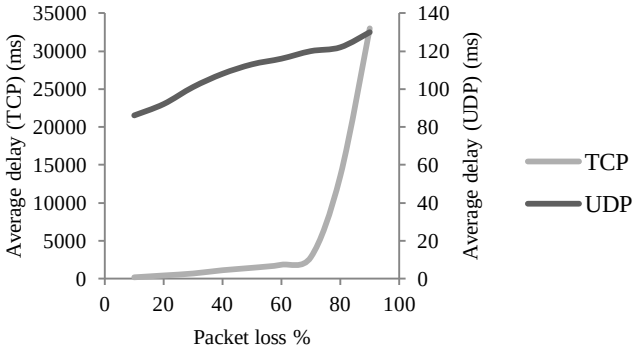

Fig. 2 Latency comparison of UDP and TCP

Thus it is seen that in such cases involving real time data transfer, a steady stream of data without latency and stutter is always preferred because lost packets are less important than packets yet to be received.

Hence a UDP connection is preferred in such scenarios. Table I summarizes the comparison of UDP with TCP.

TABLE **I**
COMPARISON OF UDP WITH TCP

TCP	**UDP**
Broken streams of data exchange	Steady stream of data exchange
Priority for next packet to be received	No priorities
All packets received	Packets are lost
Used when integrity of data is more important than a steady stream of data to be transmitted. When an old packet that has not been received is more important than a new packet of the same.	Used where real time data exchange is needed. When a new packet is more important than an old one that has not been received

IV. PROPOSED HYBRID TUNNEL

Once established that datagram traffic was essential, a means of securing datagram communication is prepared which involves a tunneling architecture.According to the scheme, a tunnel is established using TCP mechanism involving handshakes and acknowledgements. This tunnel, once established is then made to transmit UDP packets without any mechanism for retransmission of lost packets or ordering as per UDP style. The tunnel establishes the ports for each channel before transmission and then enforces strict encryption on every UDP packet that was transmitted through it thereby ensuring security(in an SSH fashion).

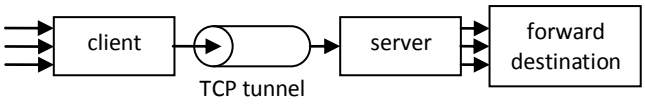

Fig. 3 Hybrid Tunnel Architecture

The TCP tunnel is prepared, as per the diagram, to transmit datagrams. The diagram shows unidirectional traffic. The bi-directional implementation involves the same process in the opposite direction. The tunnel itself is capable of handling bi-directional traffic.

The architecture involves multiple clients exchanging UDP data connected to one station. The station listens on a preset port for incoming data. Any data transmitted onto the said port is then tunneled via the established tunnel to a preset destination that is set up with the tunnel. From this destination, the address of the received packets are then resolved and retransmitted to the ports excepting the data.

The design involves a server thread for each port on client side (supporting UDP) per client/port on server side (with NAT). Client side server sends everything to the same forward destination with different source forwarding port to same forward destination port on server side. UDP packets are encapsulated appropriately before transmission.

Thus the system takes advantage of TCP services while establishing the tunnel to make sure ports are available and then uses the merits of UDP services during communication to ensure speedy delivery of data over the tunnel.

This system also provides the additional advantage of allowing applications to decide the kind of encryption they wish to enforce by using conventional encryption mechanisms such as 3DES, IDEA, Blowfish, Twofish or CAST. Security implemented on the application layer implies there will be no lapse of security arising from the clash of functions between different OSI layers in a multilayer security implementation like SRTP and SSL/TLS.

V. COMPARISON OF EXISTING TUNNELS WITH PROPOSED HYBRID

In the established tunnel implementation, a means of creating a TCP connection and forwarding UDP packets to various ports is used by NAT translating packet headers to send them to the right ports. This implements additional anonymity over the tunnel, similar to the TOR protocol. Unlike TOR however, there is no relay circuit and hence even if someone were to latch on to the socket on the client side, the proxy would

TABLE II
COMPARISON OF EXISTING TUNNELS WITH PROPOSED HYBRID

TOR	**PPTP**	**L2TP**	**Proposed model**
Uses Socks5 proxy system with a proxy tunnel involving all data transmitted through a proxy server	No proxy in implementation	No proxy in implementation	Address translation at sending and receiving end to implement proxy on data.
Uses a relay circuit with layered encryption on all packets such that last hop transfers plain text unless text is encrypted before entering TOR circuit	Encryption is dependent on user. No relays	Encryption is dependent on user. No relays.	All data through tunnel is encrypted. No possibility of plain text being transferred. No relays.
Tunnel established with proxy server as medium using SOCKS5 proxy mechanism	TCP tunnel establishment and maintenance	UDP tunnel establishment and maintenance.	TCP tunnel establishment and maintenance
TCP and UDP traffic supported but not optimized for either. SOCKS5 proxy tunnel used	Optimized for TCP traffic	Optimized for UDP traffic	Optimized for UDP traffic

disable metadata analysis attacks and there would be no chance of him obtaining plain text like in case of TOR.

In a manner similar to that of PPTP, the system establishes a TCP connection to create, maintain and terminate the tunnel. In a manner similar to that of L2TP, the packets passing through the tunnel are encapsulated and encrypted into an optimized hybrid for transmission of UDP packets in the tunnel. Unlike PPTP, the system works better for a UDP transmission and unlike L2TP, the tunnel is maintained by a TCP connection which has a different encapsulated structure thereby making it more secure.

To summarize, the hybrid tunnel model being used combines a proxy solution similar to SOCKS5 used by TOR network, the tunnel establishment technique used by PPTP and the transmission and encapsulation technique used by L2TP to create an optimized hybrid for secure real time data transmission.

Table II summarizes the comparison of Existing Tunnels with Proposed Hybrid.

VI. CONCLUSION AND FUTURE SCOPE

Existing tunneling solutions and their merits and demeritsare discussed in the paper and that the choice between TCP and UDP is specific to the application in question has been established. A means of securing a UDP communication channel thereby improving the existing means in which datagrams are transmitted has been provided. The proposed system can be adapted to be used in every case where datagram traffic is preferred over packet data in TCP format.

Some classic cases include VoIP scenarios (as established earlier), establishing connections with Access Points (both wired and otherwise) involvespassword exchange using datagrams that are currently not secure enough and license exchange in original software purchases are done with datagrams. Other than the described scenarios, all real-time data transfer takes place on a UDP connection and the system provides a means to secure all such channels.

ACKNOWLEDGEMENT

We would like to thank Dr. J. A. Laxminarayana, Head of Dept. of Computer Engineering forproviding us the necessary resources and infrastructure; Prof. Nagraj Vernekar from Dept. of Computer Engineering for supporting and mentoring us; Prof. Sherica Menezes for her valuable advice on writing this paper; and Prof. Siddesh Salelkar for initial insight in the field.

REFERENCES

[1] Dingledine, Roger; Mathewson, Nick; Syverson, Paul (13 August 2004). "Tor: The Second-Generation Onion Router". Proc. 13th USENIX Security Symposium. San Diego, California.

[2] Heartbleed - https://heartbleed.com (published 7th of April 2014, ~19:00 UTC)

[3] RFC 2341 Cisco Layer Two Forwarding (Protocol) "L2F" (a predecessor to L2TP)

[4] RFC 2637 Point-to-Point Tunneling Protocol (PPTP) (a predecessor to L2TP)

[5] RFC 768 – User Datagram Protocol

[6] RFC 793 – TCP v4; RFC 1323 – TCP-Extensions;RFC 675– Specification of Internet Transmission Control Program, December 1974 Version