Transport Area Working Group	S. Baset
Internet-Draft	H. Schulzrinne
Intended status: Experimental	Columbia University
Expires: December 9, 2009	June 07, 2009

TOC

TCP-over-UDP
draft-baset-tsvwg-tcp-over-udp-01

Status of this Memo

Copyright Notice

Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

We present TCP-over-UDP (ToU), an instance of TCP on top of UDP. It provides exactly the same congestion control, flow control, reliability, and extension mechanisms as offered by TCP. It is intended for use in scenarios where applications running on two hosts may not be able to establish a direct TCP connection but are able to exchange UDP packets.

Table of Contents

1. Introduction

TOC

Network address translators (NATs) pose a challenge for establishing a direct TCP connection between hosts. While TCP connectivity works when a TCP client is behind a NAT device and the server is not, it is problematic when both the TCP client and server are behind different NAT devices. Thus, applications running on hosts behind different NAT devices may not be able to establish a direct TCP connection with each other. Instead, these applications must establish a TCP connection with a reachable host, which relays the traffic of the application on the first host to the application on the second host and vice versa. While this works, this is undesirable as it creates a dependency on a reachable host. With certain NAT types, even though the applications cannot establish a direct TCP connection, they may be able to exchange UDP traffic by using techniques such as ICE-UDP (Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols," October 2007.) [I-D.ietf-mmusic-ice]. Thus, using UDP is attractive for such applications as it removes the dependency on a reachable host. However, these applications have a requirement that the underlying transport be reliable. Further, these applications may run on machines with heterogeneous network connectivity, thereby requiring flow control. UDP does not provide reliability, congestion control, or flow control semantics. Therefore, these applications may either use TCP with a reachable host, or invent their own reliable, congestion control, and flow control transport protocol to establish a direct connection.

We present TCP-over-UDP (ToU), a reliable, congestion control, and flow control transport protocol on top of UDP. The idea is that TCP is a well-designed transport protocol that provides reliable, congestion control, and flow control mechanisms and these mechanisms must be reused as much as possible. Further, a transport protocol that provides reliability and flow control mechanisms must not be tied to a specific application and must be designed to provide modular functionality. To accomplish this, ToU almost uses the same header as TCP which allows to easily incorporate TCP's reliable and congestion control algorithms as defined in TCP congestion control (Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control," July 2009.)
[I-D.ietf-tcpm-rfc2581bis] document. In essence, ToU is not a new protocol but merely an instance (or profile) of TCP over UDP minus the TCP checksum, urgent flag, and urgent data.

We think that our approach is attractive for several reasons. First, we are not proposing a new congestion control algorithm. Designing new congestion control algorithms is complex, and requires a large validation effort. Second, our approach takes advantage of existing user-level-TCP (such as Daytona (Pradhan, P., Kandula, S., Xu, W., Sheikh, A., and E. Nahum, "Daytona : A User-Level TCP Stack," 2004.)

[Daytona] and MINET (Dinda, P., “The Minet TCP/IP Stack,” 2002.) [MINET]) or TCP-over-UDP implementations (such as atou (Dunigan, T. and F. Fowler, “A TCP-over-UDP Test Harness,” 2002.) [atou]). Finally, since we are replicating TCP semantics over UDP, any TCP options such as window scaling (Jacobson, V., Braden, B., and D. Borman, “TCP Extensions for High Performance,” May 1992.) [RFC1323], selective acknowledgement option (SACK) (Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, “TCP Selective Acknowledgment Options,” October 1996.) [RFC2018], or proposed TCP options such as TCP-Auth (Touch, J., Mankin, A., and R. Bonica, “The TCP Authentication Option,” March 2010.) [I-D.ietf-tcpm-tcp-auth-opt] can be easily incorporated in ToU without a new standardization effort.

1.1. Conventions

TOC

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.) [RFC2119].

1.2. Terminology

TOC

We use the terms such as congestion window (cwnd), initial window (IW), restart window (RW), receiver window (rwnd), and sender maximum segment size (SMSS) as defined in TCP congestion control (Allman, M., Paxson, V., and E. Blanton, “TCP Congestion Control,” July 2009.) [I-D.ietf-tcpm-rfc2581bis] document.

---

2. Model of Operation

TOC

Like TCP, ToU has a client and a server. A client connects to a TCP server to establish a ToU connection. Below, we describe the key ToU operations.

2.1. Setup and tear down

TOC

Like TCP, ToU uses a three-way handshake to establish a connection. Similarly, it follows TCP's semantics in tearing down the connection.

2.2. Connection tracking

TOC

A key difference between TCP and UDP is that the former is connection-oriented whereas the later is not. This means that a ToU server must provide a way to keep track of existing connections. It does so through the source port and IP address of the UDP packet.

2.3. MTU discovery

TOC

ToU uses packetization layer path MTU discovery (Mathis, M. and J. Heffner, “Packetization Layer Path MTU Discovery,” March 2007.) [RFC4821] to discover link MTU.
Some NAT devices placed in front of PPPoE devices perform MSS clamping, i.e., they rewrite TCP's MSS option in a SYN packet from 1500 bytes to 1492 bytes. This operation is performed because PPPoE has a MTU of 1492 bytes instead of Ethernet's 1500 bytes. MSS clamping is considered a 'faster' way of discovering MTU in such scenarios. MSS clamping does not work for ToU because NAT devices treat ToU packets as a stream of UDP packets. It is an open question how a ToU stack should deal with PPPoE MTU if faster MTU discovery is desired. One option is to configure ToU stack with a default MTU of 1492 bytes.

3. Congestion Control, Flow Control, and Reliability

TOC

ToU follows the TCP congestion control algorithms described in TCP congestion control (Allman, M., Paxson, V., and E. Blanton, “TCP Congestion Control,” July 2009.) [I-D.ietf-tcpm-rfc2581bis] document. Thus, a ToU sender goes through the slow-start and congestion-avoidance phases. A ToU sender starts with an initial window (IW) following the guidelines in RFC 3390 (Allman, M., Floyd, S., and C. Partridge, “Increasing TCP's Initial Window,” October 2002.) [RFC3390]. During slow start, a ToU sender increments congestion window (cwnd) by at most SMSS bytes for each ACK received that cumulatively acknowledges new data. It switches to congestion avoidance when the congestion window (cwnd) exceeds slow start threshold (ssthresh). A ToU receiver generates an acknowledgement following the guidelines in Section 4.2 of TCP congestion control (Allman, M., Paxson, V., and E. Blanton, “TCP Congestion Control,” July 2009.) [I-D.ietf-tcpm-rfc2581bis] document. It immediately generates an ACK when an out-of-order segment arrives. The ToU sender uses the fast retransmit algorithm to detect and repair

losses, and fast recovery algorithm to govern the transmission of new data until a non-duplicate ACK arrives. When ToU sender has not received a segment for more than one retransmission timeout (RTO), cwnd is reduced to the value of the restart window (RW) before transmission begins. The ToU sender may also use selective acknowledgement option (SACK) (Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options," October 1996.) [RFC2018] to improve loss recovery when multiple packets are lost from one window of data. Like TCP, it uses receiver window (rwnd) to achieve flow control.

3.1. Explicit Congestion Notification (ECN)

TOC

TCP-over-UDP operates above UDP. To use ECN (Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," September 2001.) [RFC3168] with ToU, a UDP socket must allow ToU to set and retrieve the ECN bits in the IP header. Currently, UDP sockets do not provide such a mechanism. However, ToU assumes that in future, UDP sockets will provide this mechanism so that ECN can be incorporated in the congestion control mechanism of ToU.
ToU endpoints also need to determine whether they both support ECN. Similar to ECE and CWR flags for TCP as defined in ECN (Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," September 2001.) [RFC3168], ToU header includes these flags.

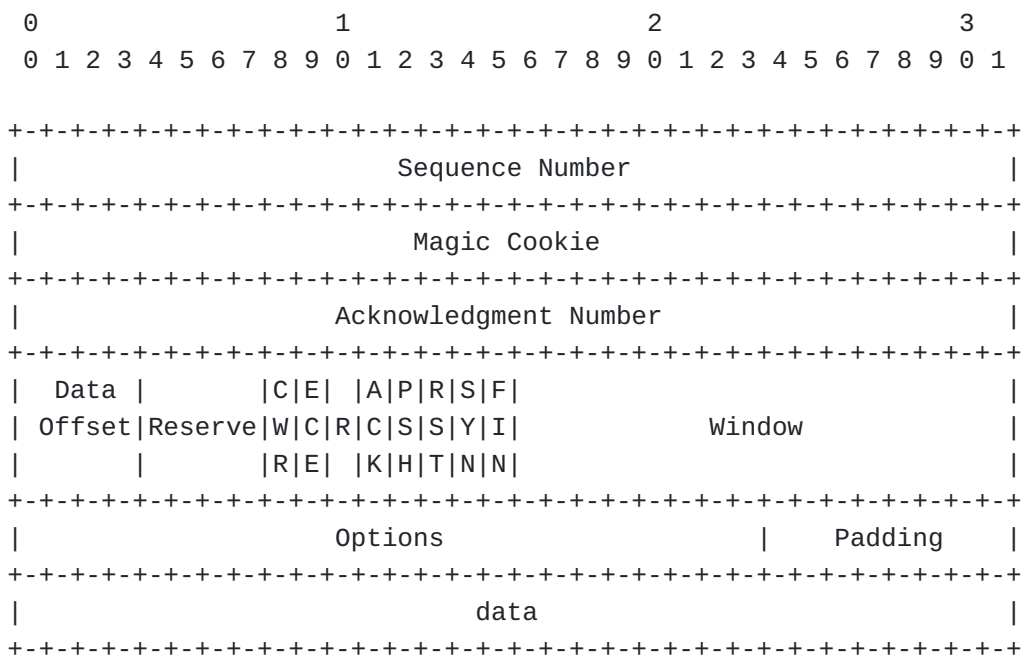
---

4. Header Format

TOC

A ToU header is like a TCP header (Postel, J., "Transmission Control Protocol," September 1981.) [RFC0793] except that it does not include source port, destination port, and checksum, as they are already included in the UDP header. ToU header also does not include the 1-bit Urgent flag and bit corresponding to this flags are reserved in the ToU header. Further, it also does not include the 16-bit Urgent Pointer. The reason for excluding Urgent flag and Urgent pointer is that they are only used in Telnet (Postel, J. and J. Reynolds, "Telnet Protocol Specification," May 1983.) [RFC0854] which is not a widely used protocol.
Between sequence number and acknowledgement number, ToU header has a 32-bit magic cookie to demultiplex it with other UDP-based protocols such as STUN (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)," October 2008.) [RFC5389].
A ToU header also includes ECE and CWR flags for negotiating ECN capabilities. These flags are defined in RFC 3168 (Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion

Notification (ECN) to IP," September 2001.) [RFC3168]. The rest of the fields in a ToU header have exactly the same meaning as those in a TCP header. The size of the fixed ToU header is 16 bytes, whereas the size of fixed TCP header is 20 bytes. The fixed ToU header and UDP header have a cumulative size of 24 bytes, four more than a fixed TCP header.



Header for TCP-over-UDP (ToU)

Figure 1

Since ToU header fields are exactly the same as TCP, we have borrowed their descriptions from the TCP RFC (Postel, J., "Transmission Control Protocol," September 1981.) [RFC0793].

> **Sequence Number (32-bits):** Same as a TCP sequence number.
>
> **Magic Cookie (32-bits):** A fixed value of 0x7194B32E in network byte order to demultiplex ToU from other application layer protocols.
>
> **Acknowledgement Number (32-bits):** Same as a TCP acknowledgement number.
>
> **Data offset (4-bits):** The number of 32-bit words in ToU header. Like a TCP header, ToU header is an integral number of 32-bits long.

Reserved (4-bits):
Reserved for future use. Must be zero.

Control Bits (8-bits): 8-bits from left to right. Unlike TCP, the Urgent bit is excluded.

CWR: Congestion window reduced flag as defined in RFC 3168 (Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," September 2001.) [RFC3168].

ECE: ECN-Echo flag as defined in RFC 3168 (Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," September 2001.) [RFC3168].

R: Reserved in ToU. In the TCP header, it is used for the Urgent bit.

ACK: Acknowledgment field significant

PSH: PSH function.

RST: Reset the connection

SYN: Synchronize sequence numbers

FIN: No more data from sender

Window (16-bits): Same as the window in TCP header. The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.

Options: Same as TCP options.

Padding: Like TCP, the ToU header padding is used to ensure that the ToU header ends and data begins on a 32 bit boundary. The padding is composed of zeros.

---

5. NAT related issues

TOC

This section discusses how to determine if hosts should use ToU and the impact of UDP NAT bindings on ToU connection management.

TOC

5.1. Using ToU

Hosts should only use ToU when establishing a direct TCP connection fails. It is outside the scope of this draft to specify a mechanism to determine if establishing a TCP connection fails between two hosts behind NATs. Hosts may use ICE-TCP (Perreault, S. and J. Rosenberg, "TCP Candidates with Interactive Connectivity Establishment (ICE)," October 2009.) [I-D.ietf-mmusic-ice-tcp] and ICE-UDP (Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols," October 2007.) [I-D.ietf-mmusic-ice] to determine if hosts can directly establish a TCP connection or directly exchange UDP packets, respectively. If hosts fail to establish a direct TCP connection but are able to directly exchange UDP packets, they can establish a ToU connection.

5.2. NAT bindings

TOC

NAT devices maintain a binding for mapping an internal IP address and port number to an external IP address and port number. The lifetime of bindings for UDP is much smaller than TCP because UDP is a connection less protocol. If an application does not send packets over ToU, the UDP binding may be lost resulting in a broken ToU connection.
ToU does not provide any mechanism to determine UDP binding lifetimes or to refresh these bindings. Rather, an application establishing a ToU connection can use STUN (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)," October 2008.) [RFC5389] to discover (MacDonald, D. and B. Lowekamp, "NAT Behavior Discovery Using STUN," September 2009.) [I-D.ietf-behave-nat-behavior-discovery] binding lifetimes and periodically refresh these bindings. Running STUN in conjunction with ToU has a design implication that a ToU packet must be differentiated from a STUN packet. The magic cookie in a ToU packet serves this purpose.

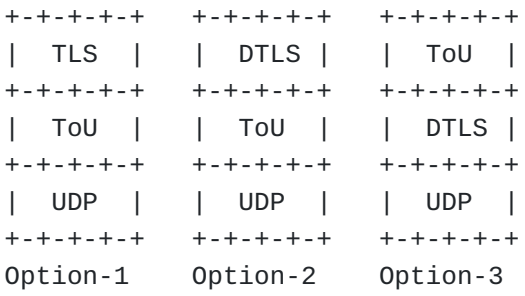
---

6. ToU, TLS, and DTLS

TOC

Transport layer security (TLS) (Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," August 2008.) [RFC5246] and Datagram transport layer security (DTLS) (Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security," April 2006.) [RFC4347] protocols provide privacy and data integrity between two communicating applications. TLS is layered on top of some reliable transport protocol such as TCP, whereas DTLS only assumes a datagram

service. A question is what is the layering relationship between ToU protocol, TLS, and DTLS. Figure 2 shows three possible options. Option-3 is not feasible since ToU layer must be made aware of the size of header which DTLS may add. Option-2 layers DTLS on top of ToU. Unlike TLS, DTLS carries a sequence number because it assumes a datagram service. However, the use of sequence number is made redundant because ToU provides reliable and inorder delivery semantics. Therefore, Option-1 is most feasible in which TLS is layered on top of ToU.



Layering options for ToU, TLS, DTLS

Figure 2

7. Implementation Guidelines

TOC

From the implementers perspective, the use of ToU should be as modular as possible. Once way to achieve this modularity is to implement ToU as a user-level library that provides socket-like function calls to the applications. The library may have its own thread of execution and can be instantiated at the start of the program. The library implements the reliable, inorder, congestion control, and flow control semantics of TCP. Applications can interact with the ToU library through socket-like function calls.

8. Design Alternatives

TOC

ToU is strictly meant for scenarios where end-points desire to establish a TCP connection but are unable to do so due to the presence

of NATs and firewalls. Below, we briefly discuss the design alternatives and address possible criticisms for ToU.

8.1. Changing IP protocol number

TOC

One solution is to change the IP protocol number of TCP packets to UDP before sending them on the wire. Similarly, when the packets are received, the protocol number is changed back to TCP and the received packets are passed to the TCP stack. The idea behind this approach is to reuse TCP stack as much as possible. This approach suffers from a number of problems. First, it requires a change in the operating system kernel to rewrite IP protocol number of TCP packets to UDP and it is unrealistic to expect all the OS kernels to implement this change. Second, TCP checksum has a different offset than a UDP checksum and many NAT devices parsing the UDP packet will reject the packet because the UDP checksum is incorrect. Third, since applications can use the same port number for TCP and UDP ports, it is unclear how the kernel will correctly differentiate between TCP and UDP packets for the same port number.

8.2. Simplified TCP

TOC

It may be argued that TCP semantics are too complicated and it might be easier to define a protocol that adds retransmission of individual UDP packets, and ACK mechanisms, and sequencing layer. However, unless one is content with stop-and-wait congestion control (and roughly modem data rates), it is necessary for a transport protocol to have AIMD or rate-based congestion control (TFRC). As discussed in Section 8.5 (TFRC), rate-based congestion control is not suitable for mid-sized transfers and is not any simpler than AIMD. Further, since hosts may have heterogeneous network connectivity, a transport protocol needs to provide flow control. Moreover, it may not be easy to validate a new transport protocol that only provides selective TCP semantics.

8.3. TCP-like mechanism within an application layer protocol

TOC

In this approach, key TCP mechanisms such as reliability, congestion control, and flow control are designed as part of the application layer protocol. This approach has several disadvantages. First, every application layer protocol that is unable to establish TCP connections in the presence of NAT and firewalls but may use UDP will need to

invent its own reliable, congestion control and flow control transport protocol. Second, it is non-trivial to get the first implementations of a conceptually new protocol right. Third, any new transport protocol, even if it is specified within an application layer protocol must undergo a large validation effort. Finally, most long-term successful protocols are those that provide modular functionality, and not extremely narrowly-tailored protocols.

8.4. Tunneling

TOC

Another design option is to provide a VPN-like tunnel for sending and receiving TCP packets over UDP. The idea is to use tunneling solutions between hosts so that hosts can use the kernel TCP stack and unmodified socket functions calls.
This approach is not desirable for several reasons. First, tunneling solutions typically require support from kernel or require kernel upgrades to work. Requiring kernel upgrades to work is not plausible for an application that is trying to get deployment traction. Second, establishing a tunnel typically requires root access to the system and it is unrealistic for user-space applications to require root access for proper functioning. Third, peer-to-peer applications, which are expected to use ToU, establish a large number of connections with other hosts. Even, if a tunneling solution does not require any kernel support, such a solution consumes significant bandwidth and CPU resources to maintain a large number of tunnels with other hosts. Popular P2P applications such as Skype and Bittorrent do not take advantage of a layer-3 tunneling solution.

8.5. TFRC

TOC

TFRC (Floyd, S., Handley, M., Padhye, J., and J. Widmer, “TCP Friendly Rate Control (TFRC): Protocol Specification,” September 2008.)
[RFC5348] is a congestion control mechanism (not a protocol) that is designed for long-lived media streams. Its main benefit is of smoothing rates to these media streams. It does not provide any packet formats, reliability, or flow control. It's congestion control mechanism is not suited for exchanging data objects that range from a few dozen to a few hundred packets. The reason is that TFRC is based on estimating loss rates within 8 loss intervals. With a loss rate of 1%, this translates, very roughly, into 800 packets or roughly 800 kB, before a reliable estimate of a better (higher) rate is computed. Further, its main benefit, smoothing rates, is of no importance to applications desiring to replicate TCP functionality over UDP.

8.6. SCTP

TOC

SCTP (Stewart, R., “Stream Control Transmission Protocol,” September 2007.) [RFC4960] is significantly more complicated than TCP in its implementation and its performance is generally the same, except in circumstances involving head-of-line blocking. Further, SCTP will have trouble getting traction in the consumer and enterprise Internet space unless it (also) runs over UDP, as there seem to be few NATs that know how to handle SCTP and thus it is effectively unusable by a fair fraction of the Internet user population.

8.7. Criticism

TOC

A criticism of the ToU approach is that it is deceptively simple to describe but difficult to implement and is likely to suffer from broken implementations. We think that this assertion is not valid for three reasons. First, ToU does not define a new congestion control protocol and thus stays away from all the validation issues associated with a new congestion control protocol. Second, a reasonable implementation approach is to first implement connection management and AIMD congestion control and test it with regular TCP to determine if the implemented congestion control mechanisms are broken. This implementation can be followed by implementing TCP options such as window scaling and SACK. Third, ToU like other protocols such as SIP will be implemented as a module or library and is likely to mature over time.

9. Acknowledgements

TOC

The draft incorporates comments from the discussion on TSVWG and P2PSIP mailing list. We also acknowledge an earlier draft by R. Denis-Courmont on UDP transports.

10. IANA Considerations

TOC

TBD.

TOC

11. Security Considerations

ToU is subject to the same security considerations as TCP.

---

12. References

TOC

12.1. Normative References

TOC

[I-D.ietf-tcpm-rfc2581bis]	Allman, M., Paxson, V., and E. Blanton, “TCP Congestion Control,” draft-ietf-tcpm-rfc2581bis-07 (work in progress), July 2009 (TXT).
[I-D.ietf-tcpm-tcp-auth-opt]	Touch, J., Mankin, A., and R. Bonica, “The TCP Authentication Option,” draft-ietf-tcpm-tcp-auth-opt-11 (work in progress), March 2010 (TXT).
[RFC0793]	Postel, J., “Transmission Control Protocol,” STD 7, RFC 793, September 1981 (TXT).
[RFC0854]	Postel, J. and J. Reynolds, “Telnet Protocol Specification,” STD 8, RFC 854, May 1983 (TXT).
[RFC1122]	Braden, R., “Requirements for Internet Hosts - Communication Layers,” STD 3, RFC 1122, October 1989 (TXT).
[RFC1323]	Jacobson, V., Braden, B., and D. Borman, “TCP Extensions for High Performance,” RFC 1323, May 1992 (TXT).
[RFC2018]	Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, “TCP Selective Acknowledgment Options,” RFC 2018, October 1996 (TXT, HTML, XML).
[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC3168]	Ramakrishnan, K., Floyd, S., and D. Black, “The Addition of Explicit Congestion Notification (ECN) to IP,” RFC 3168, September 2001 (TXT).
[RFC3390]	Allman, M., Floyd, S., and C. Partridge, “Increasing TCP's Initial Window,” RFC 3390, October 2002 (TXT).
[RFC4347]	Rescorla, E. and N. Modadugu, “Datagram Transport Layer Security,” RFC 4347, April 2006 (TXT).
[RFC4821]	Mathis, M. and J. Heffner, “Packetization Layer Path MTU Discovery,” RFC 4821, March 2007 (TXT).
[RFC4960]	Stewart, R., “Stream Control Transmission Protocol,” RFC 4960, September 2007 (TXT).
[RFC5246]	

	Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, August 2008 (TXT).
[RFC5348]	Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," RFC 5348, September 2008 (TXT).
[RFC5389]	Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)," RFC 5389, October 2008 (TXT).

12.2. Informative References

TOC

[Daytona]	Pradhan, P., Kandula, S., Xu, W., Sheikh, A., and E. Nahum, "Daytona : A User-Level TCP Stack," 2004.
[I-D.ietf-behave-nat-behavior-discovery]	MacDonald, D. and B. Lowekamp, "NAT Behavior Discovery Using STUN," draft-ietf-behave-nat-behavior-discovery-08 (work in progress), September 2009 (TXT).
[I-D.ietf-mmusic-ice]	Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols," draft-ietf-mmusic-ice-19 (work in progress), October 2007 (TXT).
[I-D.ietf-mmusic-ice-tcp]	Perreault, S. and J. Rosenberg, "TCP Candidates with Interactive Connectivity Establishment (ICE)," draft-ietf-mmusic-ice-tcp-08 (work in progress), October 2009 (TXT).
[MINET]	Dinda, P., "The Minet TCP/IP Stack," 2002.
[atou]	Dunigan, T. and F. Fowler, "A TCP-over-UDP Test Harness," 2002.

Appendix A. Change Log

TOC

A.1. Changes since draft-baset-tsvwg-tcp-over-udp-00

TOC

*Updated introduction to reflect that it is difficult for two hosts behind two different NATs to establish a TCP connection.

*Added PSH bit.

*Added MTU discovery to model of operation section.

*Added text on ECN to congestion control section.

*Added a section on NAT related issues.

*Updated text in design alternatives section.

Authors' Addresses

TOC

	Salman A. Baset
	Columbia University
	1214 Amsterdam Avenue
	New York, NY
	USA
Email:	salman@cs.columbia.edu
	Henning Schulzrinne
	Columbia University
	1214 Amsterdam Avenue
	New York, NY
	USA
Email:	hgs@cs.columbia.edu