TCP Maintenance Working Group
Internet-Draft
Intended status: Experimental
Expires: January 16, 2014

T. Flach
USC
N. Dukkipati
Y. Cheng
B. Raghavan
Google

July 15, 2013

TCP Instant Recovery: Incorporating Forward Error Correction in TCP
draft-flach-tcpm-fec-00.txt

Abstract

Ordinary TCP loss recovery takes at least one round-trip time and as such can increase application-perceived latency, especially for short flows such as Web transactions. TCP Instant Recovery (TCP-IR) is an experimental algorithm that allows a receiving end to recover lost packets without retransmissions, thus potentially saving at least one full round-trip time compared to standard TCP. TCP-IR achieves this by judiciously injecting encoded data segments within a TCP stream. This document describes the TCP-IR algorithm at the sending and receiving ends, along with the required protocol changes.

Table of Contents

1. Introduction

TCP Instant Recovery (TCP-IR) enables a receiver to recover lost data segments instantly without the need for retransmissions from a sender. Standard TCP retransmission-based loss recovery takes at least one RTT for loss detection and recovery.

The main motivation for TCP-IR is to reduce the tail latency of Web transactions. Most Web transfers are short and could finish within a few round-trip times (RTTs), but losses can add multiple RTTs to transfer times and increase the variance in Web page download times. The goal of TCP-IR is to reduce loss detection and recovery to zero RTT while still employing TCP's congestion control principles.

Recovery mechanisms, such as fast recovery and retransmission timeout, are fundamentally RTT dependent. Regardless of how fast network bandwidth grows, the number of RTTs that it takes to recover lost packets does not change. TCP-IR employs forward error correction (FEC) to scale the recovery time inversely with bandwidth and make it independent of RTT. It explicitly trades some network

bandwidth to reduce RTTs for short transfers. Most bandwidth in the Internet is used by large flows such as video, and thus short, latency-sensitive traffic can benefit using a small degree of FEC without hurting bulk flow throughput.

In this document, we specify the TCP-IR mechanism, which requires both sender and receiver changes, to achieve 1-RTT recovery for commonly observed loss scenarios. Instead of complete redundancy for every segment, we employ FEC within TCP. The sender transmits extra FEC segments, which encode previously transmitted segments, so that the receiver can repair a small number of losses. While the use of FEC for transport has been explored in the past, to our knowledge this is the first specification to place FEC within TCP in a way that is incrementally deployable across today's networks with middleboxes.

1.1. Terminology

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 [RFC2119].

2. Protocol Overview

The key idea in TCP-IR is the judicious introduction of a small number of checksum or XOR segments into TCP's data stream such that a receiver can immediately recover lost segment(s) without the need for retransmissions. The core design challenge is in injecting out of band XOR segments within the regular data stream. We outline the main aspects of the protocol below.

Several of the design choices we made are rooted in our measurements and observations of Internet loss patterns as documented in [RECOVERY-SIGCOMM13]. We find that among the flows experiencing losses, most flows lose only one or two consecutive packets, commonly at the tail of a burst. As an example, for bursts of at most 10 packets, ~35% experienced exactly one loss, and an additional 10% experienced exactly two losses. Further, the latter packets for any given burst size are more likely to be lost. Given these findings, we chose a simple XOR-based encoding scheme that can perform instant recovery of a small amount (one or two segments) of packet loss.

A TCP-IR sender and receiver first negotiate the use of instant recovery in the initial handshake. If both hosts support the use of instant recovery, every packet in the connection includes a TCP-IR option.

TCP-IR sender:

1. Periodically in every round-trip time, a TCP-IR sender places the XOR of newly transmitted segments into a single MSS-length checksum packet. The XOR is only computed for new segments not previously included in checksums.

2. Regardless of the sizes of original segments, the sender computes the XORs along MSS byte boundaries. Because every packet carries a payload of at most MSS bytes, such an encoding guarantees that the receiver can instantly recover any single packet loss.

3. The encoded XOR packet uses the same sequence number as the first segment it encodes. The encoded packet carries a flag in the TCP-IR option signaling that the payload is encoded. A receiver uses the flag to disambiguate an encoded packet from a regular (re)transmission, since both segments carry the same sequence number. The option also includes the number of bytes that the payload encodes.

There is no reliability provided for the XOR segments.

TCP-IR receiver:

1. A receiver first establishes if the payload of the received segment is encoded, by checking a flag in the TCP-IR option.

2. Once the receiver establishes that the payload is encoded, it obtains the encoded range of bytes by using the sequence number of the TCP-IR packet and the the number of bytes encoded.

3. The receiver checks for holes in the encoded range. If it received the entire sequence range, the receiver drops the encoded packet. Otherwise, if it is missing at most MSS contiguous bytes, the receiver uses the encoded payload to recover the lost sequence range and forwards it to the regular reception routine, thus allowing 0-RTT recovery.

4. For the purpose of recovering lost segments, a receiver buffers the last fifteen in-order MSS blocks that it ACKed, even if the application layer has already consumed these blocks. Because an encoded packet is the XOR of at most sixteen MSS segments, the receiver can recover any single lost packet by computing the XOR of the encoded payload and the buffered data in the encoding range.

5. If too much data is missing for the encoded packet to recover, the receiver sends a duplicate ACK. This ACK informs the sender that a recovery failed and also denotes the byte ranges lost via the TCP-IR option. The sender marks the byte ranges as lost and triggers a fast retransmit and recovery.

6. TCP-IR does not circumvent congestion control. If the receiver were to simply ACK a recovered packet, it would mask the loss and prevent congestion control during a known loss episode. To perform congestion window reduction upon a successful recovery at the receiver, TCP-IR uses a mechanism similar to explicit congestion notification (ECN). Upon a successful recovery, the receiver enables an R_SUCC flag in the TCP-IR option in each outgoing ACK. The sender in turn triggers a congestion window reduction and sets an R_ACK flag in the TCP-IR option of the next packet sent to the receiver. Once the receiver observes R_ACK in an incoming packet, indicating that the sender reduced the congestion window, it disables R_SUCC for future packets.

Figure 1 gives an example of TCP-IR in action.

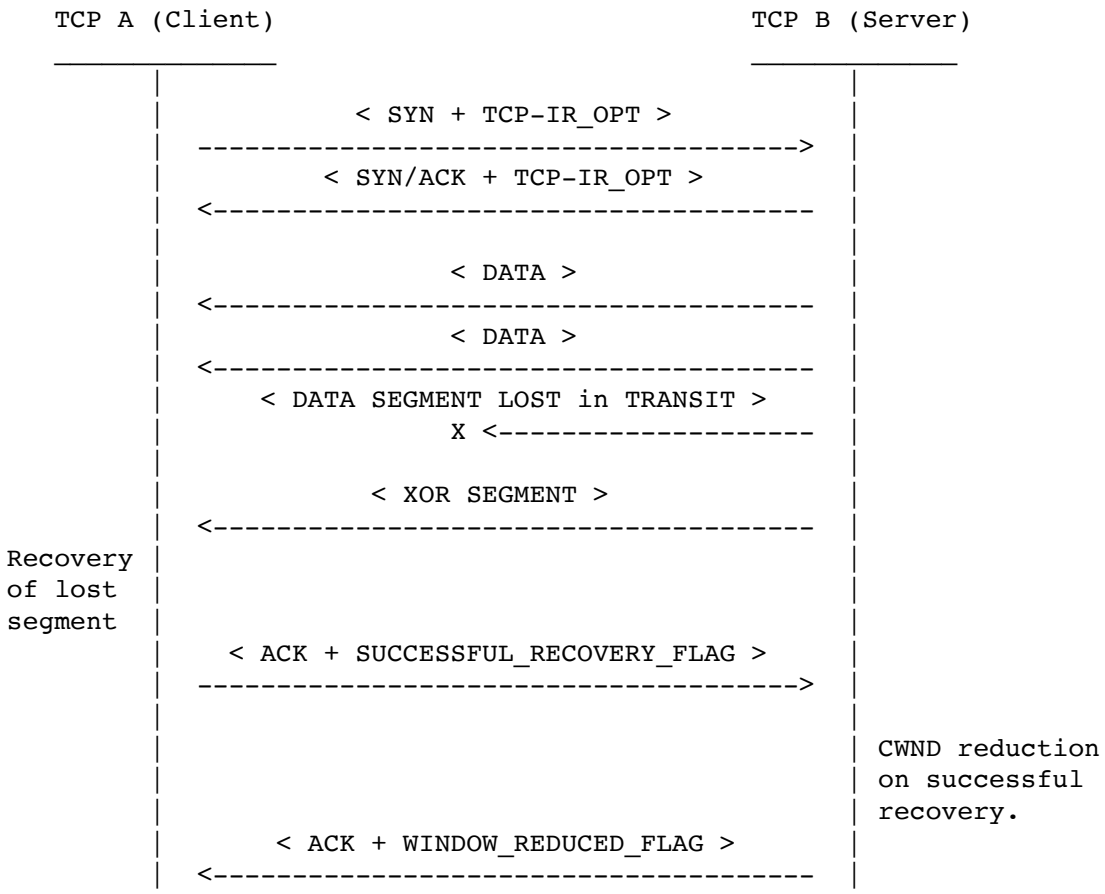

Figure 1: Sample flow using TCP-IR

3. Protocol Details

In the following, we describe the TCP-IR option design, negotiation of instant recovery, the supported encoding schemes, and finally the sender and receiver side algorithms.

3.1. TCP-IR Option

Both the server and the client use a new option to perform the following:

- Negotiate the use of TCP-IR, including the encoding type.
- Distinguish encoded packets from regular packets.
- Communicate the number of encoded bytes in an XOR packet.
- Acknowledge the recovery of segments and congestion window reductions.
- Indicate the loss of segments.

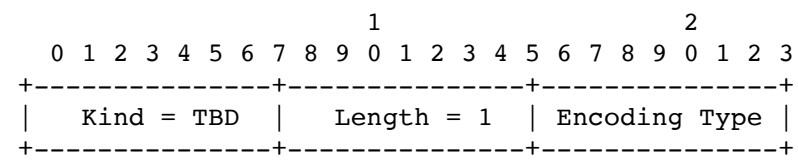

Figure 2: TCP-IR Option format for packets with SYN flag set

During the initial handshake (for packets with the SYN flag set), the option has the format shown in Figure 2. It contains the following fields:

Kind (8 bits)
: This MUST be set to the option number for TCP-IR to be determined by IANA.

Length (8 bits)
: This MUST be set to the length of the TCP option in octets; its value MUST be 1.

Encoding Type (8 bits)
: This SHOULD be set to a value corresponding to a supported encoding type (see Section 3.3).

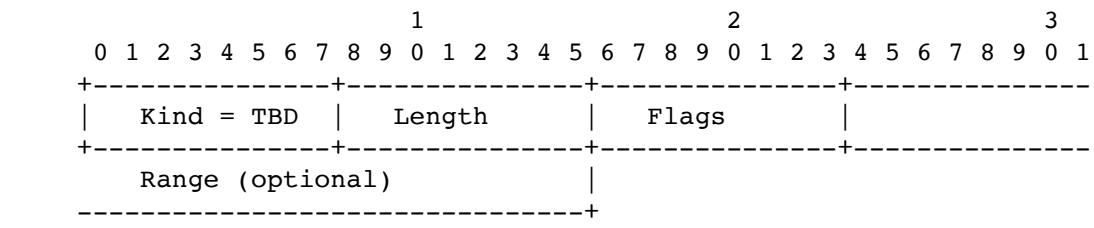

Figure 3: TCP-IR Option format (except for packets with SYN flag set)

For all other packets, the option has the format shown in Figure 3. It contains the following fields:

Kind (8 bits)
  This MUST be set to the option number for TCP-IR to be determined by IANA.

Length (8 bits)
  This MUST be set to the length of the TCP option in octets; its value MUST be 1, or 4 (if the "Range" field is appended).

Flags (8 bits)
  The field can carry the following flags (each represented by one bit):

Bit	Flag Name	Description
0	R_CWR	Congestion Window Reduction Acknowledgement
1	R_SUCCESS	Recovery successful
2	R_FAIL	Recovery failed
3	ENCODED	Packet is encoded
4-7		Unused

  The unused bits SHOULD NOT be set.

Range (24 bits, optional)
  This field is only used if either the ENCODED or R_FAIL bit in the "Flags" field is set. If the ENCODED bit is set, this field specifies the number of bytes encoded in the payload. If the R_FAIL bit is set, this field specifies the number of bytes considered lost (see Section 3.4 and Section 3.6).

3.2. Negotiation

TCP-IR MUST be explicitly negotiated during the initial handshake. If the negotiation succeeds, both endpoints can send and receive TCP-IR packets. More specifically TCP-IR is enabled if:

1. The receiver sends the SYN packet carrying the TCP-IR option. The encoding type field MUST carry a valid encoding type (see Section 3.3).
2. The sender responds with a SYN/ACK carrying the TCP-IR option. The encoding type field MUST carry the same encoding type as the option in the corresponding SYN packet.
3. All following packets (transmitted by either sender or receiver) MUST carry the TCP-IR option.

If any endpoint receives a packet (after negotiation succeeds) that does not carry the TCP-IR option, the connection MUST be reset. This is necessary because a receiver can no longer distinguish between regular and TCP-IR packets. We recommend tracking these cases to avoid TCP-IR negotiation on future connections.

3.3. Encoding Types

The client can select the encoding type to be used by the TCP-IR module but both endpoints have to support it and agree on it during the initial handshake (see Section 3.2).

Currently the following encoding types are supported and are therefore valid values in the "Encoding Type" field of the TCP-IR option during negotiation:

Value	Type	Description
0	Undefined	
1	Basic XOR	TCP-IR packets carry the XOR of every MSS length segment. One TCP-IR packet encodes up to 16 MSS length segments.
2	Interleaved XOR	TCP-IR packets carry the XOR of every other MSS length segment. One TCP-IR packet encodes up to 8 MSS length segments.
3-255	Undefined	

We selected these encoding types to demonstrate the flexibility of TCP-IR with respect to user preferences (like the acceptable amount of redundancy) and connection properties. Basic XOR can recover a single segment loss of up to MSS bytes in the encoding range. Interleaved XOR enables the recovery of two consecutive segments of up to MSS bytes. This makes Interleaved XOR suitable for connections observing bursty losses, but can double the number of generated TCP-IR packets.

The currently supported encoding types use the MSS value to determine the block size for the encoding process. If the MSS value changes after the initial handshake, TCP-IR MUST be disabled for the remainder of the connection.

3.4. TCP-IR Sender

All packets after the initial handshake carry the TCP-IR option to ensure that the receiver can always distinguish regular packets from encoded packets (packets carrying a payload encoded by the TCP-IR module), or detect the removal of the option by a middlebox. Encoded packets MUST set the ENCODED bit in the "Flags" field of the TCP-IR option; all other packets MUST NOT set the ENCODED bit.

The TCP-IR packet MUST use the same sequence number as the first byte it encodes. This prevents enforcing reliability for encoded packets as well as the overhead of specifying the index of the first encoded byte in a separate option field.

In addition to that, the option in encoded packets MUST carry the "Range" field. The value in the "Range" field specifies the index of the byte after the last encoded byte in the payload relative to the sequence number of the encoded packet.

TCP-IR adds a short delay in the transmission of encoded packets to the reduce the probability of losing both the original transmission and the encoded packet in the same loss burst.

The encoding and transmission routine works as follows:

1. Before a regular data packet is forwarded to the IP layer, the TCP-IR timer is armed (unless the timer is already armed). In our prototype implementation the timer is set to a value of RTT/4.
2. Once the timer fires, all transmitted segments not encoded before are now encoded according to the negotiated encoding type and the corresponding encoded packets are transmitted immediately. The maximum number of MSS length segments which can be encoded in a single TCP-IR packet depends on the negotiated encoding type (see Section 3.3). As a result, the number of encoded packets created in this step depends on the encoding type and the number of previously un-encoded segments. The option fields in the encoded packet are populated as described in Section 3.1.

3.5. TCP-IR Receiver

Receivers distinguish TCP-IR packets from regular packets by checking the ENCODED bit in the "Flags" field of the TCP-IR option. Encoded

packets are forwarded to the TCP-IR reception routine (described below). If the packet does not carry the TCP-IR option it is discarded and TCP-IR is disabled for the remainder of the connection. To inform the sender that TCP-IR can no longer be used, the receiver sends an acknowledgement without the TCP-IR option.

Additionally, the regular reception routine is modified as follows. The last 15 ACKed MSS length segments remain in the buffer, even if the application layer has already consumed these segments. Segments received out-of-order are already buffered by default and cannot be consumed by the application layer. Since a single TCP-IR packet encodes at most 8 (interleaved XOR) or 16 (basic XOR) MSS length segments, any single segment loss (up to MSS length) in the encoding range can be recovered by the decoder.

The reception routine for TCP-IR packets works as follows:

1. The encoding range of the TCP-IR packet is extracted. As mentioned earlier, the sequence number of the packet specifies the sequence number of the first encoded byte. The sequence number plus the value stored in the "Range" field in the TCP-IR option minus 1 specifies the sequence number of the last encoded byte.
2. If all bytes in the encoding range were already received, skip to Step 5.
3. If lost segments in the encoding range can be recovered (in the case of XOR encoding, a loss of at most one MSS length segment in the encoding range can be handled):

    a. The lost segments are reconstructed. The matching packet headers are appended to the reconstructed segments and the packets are forwarded to the regular reception routine.
    b. The R_SUCCESS bit in the "Flags" field of the TCP-IR option is set for all future packets. This includes the (potentially delayed) acknowledgement for the recovered segment. Further details are described in Section 3.6.
4. If none of the segments in the encoding range are recoverable:

    a. The sequence number of the last byte lost is extracted. The offset between the sequence number of the next expected byte (RCV.NXT) and the last byte lost defines the loss range.
    b. An acknowledgement is generated with the following requirements for the TCP-IR option.

        + The R_FAIL bit in the "Flags" field is set.

+ The option carries the "Range" field. The "Range" field encodes the loss range, as described above. The context is maintained, since the acknowledgement number will be set to RCV.NXT.

5. The TCP-IR packet is discarded.

3.6. Processing Acknowledgements

If a receiver instantly recovers losses we want to ensure the sender learns of it so as to not circumvent congestion control [RFC5681]. The R_SUCCESS bit in the "Flags" field of the TCP-IR option informs the sender that the receiver successfully recovered a lost packet. Once the sender observes the R_SUCCESS bit in a packet the following steps are executed:

1. The sender reduces its congestion window.
2. The sender sets the R_CWR bit in the "Flags" field of the TCP-IR option in the next outgoing packet only.
3. The sender does not act on any future observations of the R_SUCCESS bit being set until SND.UNA advances past the SND.NXT value observed at the time when Step 2 was executed. This ensures the congestion window is not reduced multiple times in the same loss episode.
4. Once the receiver observes the R_CWR bit being set in any incoming packet, the R_SUCCESS bit is reset for all future packets.

A failed recovery on the receiver side triggers an explicit acknowledgment sent to the sender to inform it about the segments that are considered lost. This is indicated by the R_FAIL bit being set in the "Flags" field of the TCP-IR option. If the sender observes this bit being set, the following steps are executed:

1. The sender extracts the loss range from the "Range" field in the TCP-IR option. The sequence numbers of the first and last byte lost are defined by the acknowledgement number of the packet, and the acknowledgement number plus the loss range value.
2. The sender marks the appropriate byte range as lost and triggers Fast Retransmit/Recovery.

Explicit notification of loss ranges has the benefit that lost segments are retransmitted faster, avoiding the extra wait time until the RTO fires.

4. Interaction with middleboxes

An important design goal of TCP-IR is compatibility with middleboxes and support for graceful fallback to standard TCP behavior in

situations where middlebox interference prevents proper use of TCP-IR.

Even if hosts negotiate TCP-IR during the initial handshake, it is possible for a middlebox to strip the option from a later packet. To be robust to this, if either host receives a packet without the option, it MUST discard the packet and reset the connection. This is necessary since receivers are no longer able to distinguish TCP-IR packets from regular packets.

TCP-IR uses relative sequence numbers to convey metadata (such as the encoding range) between endpoints. This prevents issues in the cases of middleboxes performing sequence number translations.

Some problems caused by middlebox interference (and their solutions in TCP-IR) are not discussed in the current version of this draft:

- Rewriting of the acknowledgement number if the acknowledged segment was not observed by the middlebox. With TCP-IR this can occur after recovering a lost segment. This issue can be circumvented by retransmitting the recovered segment, even though it is not needed by the other endpoint anymore. This plugs the "sequence hole" in the state of the middlebox.
- Rewriting payloads of previously seen segments.
- Packet coalescing and splitting.

5. Implementation and Performance

We implemented TCP-IR in Linux TCP in about 1600 lines of code. 20% of the modular implementation includes the parts common to both the sender and receiver, which are option encoding/decoding, and negotiation during connection setup. 50% of the implementation is the receiver components including detection of an encoded packet, decoding the TCP-IR payload, and generating the right acknowledgements upon a successful or failed recovery. The remainder 30% of the implementation is the sender component which consists mainly of payload encoding and transmission.

We conducted two kinds of experiments. The first was in an emulated setting using loss patterns similar to those observed in our measurement of real traffic. We used the netem module to emulate a 200 ms RTT and both random and correlated loss rates of 2%. TCP-IR reduced the latency for short transfers in lossy environments by 28% in the 90th percentile. The benefits diminish as the minimum number of RTTs necessary to complete the transaction increases (due to the message size) because the time to recover from losses no longer dominates the overall transmission time. TCP-IR is better suited for small transfers common in today’s Web.

In the second set of experiments, we used the Web-page-replay tool and dummynet to replay HTTP resource transfers for actual Web page downloads through controlled, emulated network conditions. We tested a variety of popular Web sites, and ran separate tests for Web pages tailored for desktop and mobile clients. As an example, with TCP-IR, the New York Times website takes 15% less time in the 90th percentile until the first objects are rendered on the screen.

Details on performance experiments with TCP-IR are in [RECOVERY-SIGCOMM13].

6. Related Work

Applying FEC to transport, at nearly every layer, is an old idea. [Coding-IEEE2011] suggested placing network coding in TCP, and [CodedTCP-2013] extended this work by implementing a variant over UDP mainly for high loss rate wireless environments. Among others, [AdaptiveFEC-2004] and Tickoo et al. [LT-TCP-2005] explored extending TCP to incorporate FEC. Finally, Maelstrom is an FEC variant for long-range communication between data centers leveraging the benefits of combining and encoding data from multiple sources into a single stream [Maelstrom-2011]. The focus of all of this work is on the performance aspects of using FEC over lossy links. None address the protocol level changes required in TCP to incorporate FEC.

7. Security Considerations

The security considerations outlined in [RFC5681] apply to this document. At this time we did not find any additional security problems with TCP-IR.

8. IANA Considerations

The two Options for TCP-IR used during negotiation and subsequently in every packet of the connection require IANA allocate one value from the TCP option Kind namespace. Experimentation prior to the allocation SHOULD follow [EXPOPT] and use experimental option kind 254 and two magic bytes 0xDC60, and migrate to the new option after the allocation accordingly.

Note to RFC Editor: this section may be removed on publication as an RFC.

9. References

[RECOVERY-SIGCOMM13]

Flach, T., Dukkipati, N., Terzis, A., Raghavan, B., Cardwell, N., Cheng, Y., Jain, A., Hao, S., Katz-Bassett, E., and R. Govindan, "Reducing Web Latency: the Virtue of Gentle Aggression", Proceedings of the 2013 ACM SIGCOMM , 2013.

[Coding-IEEE2011]
Sundararajan, J., Shah, D., Medard, M., Jakubczak, S., Mitzenmacher, M., and J. Barros, "Network Coding Meets TCP: Theory and Implementation", Proceedings of the IEEE , 2011.

[CodedTCP-2013]
Kim, M., Cloud, J., ParandehGheibi, A., Urbina, L., Fouli, K., Leith, D., and M. Medard, "Network Coded TCP (CTCP)", arXiv:1212.2291. , 2013.

[AdaptiveFEC-2004]
Baldantoni, L., Lundqvist, H., and G. Karlsson, "Adaptive end-to-end FEC for improving TCP performance over wireless links", IEEE Communications Society , 2004.

[LT-TCP-2005]
Tickoo, O., Subramanian, V., Kalyanaraman, S., and K. Ramakrishnan, "LT-TCP: End-to-End Framework to improve TCP Performance over Networks with Lossy Channels ", Proc. of IWQoS , 2005.

[Maelstrom-2011]
Balakrishnan, M., Marian, T., Birman, K., Weatherspoon, H., and L. Ganesh, "Maelstrom: transparent error correction for communication between data centers ", IEEE/ ACM Transactions on Networking , 2011.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

[EXPOPT] Touch, J., "Shared Use of Experimental TCP Options", draft-ietf-tcpm-experimental-options-06 (work in progress), October 2012.

Authors' Addresses

Tobias Flach
University of Southern California
941 Bloom Walk
Los Angeles, California 90089
USA

Email: flach@usc.edu
URI: http://nsl.cs.usc.edu/~tobiasflach

Nandita Dukkipati
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, California 94043
USA

Email: nanditad@google.com

Yuchung Cheng
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, California 94043
USA

Email: ycheng@google.com

Barath Raghavan
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, California 94043
USA

Email: barath@google.com