

# Improving Tor using a TCP-over-DTLS Tunnel

Joel Reardon  
Google Zurich

Ian Goldberg  
University of Waterloo

18th USENIX Security Symposium

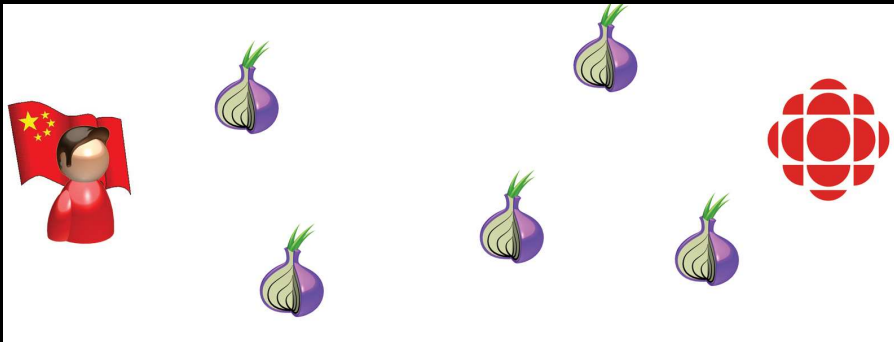
August 12th, 2009

Tor: Internet anonymity tool

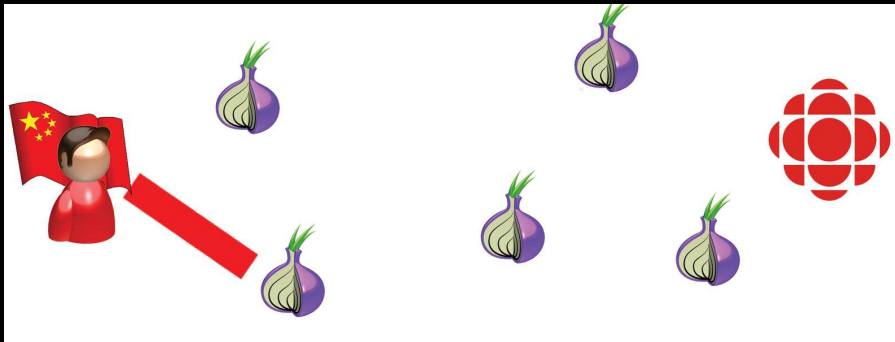
# Problem



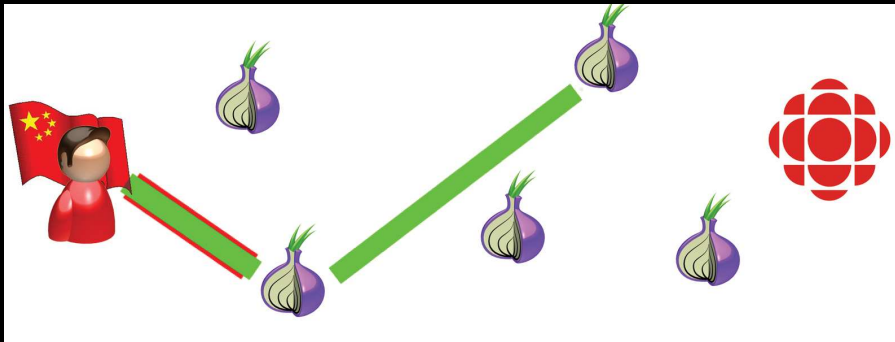
# Tor Network



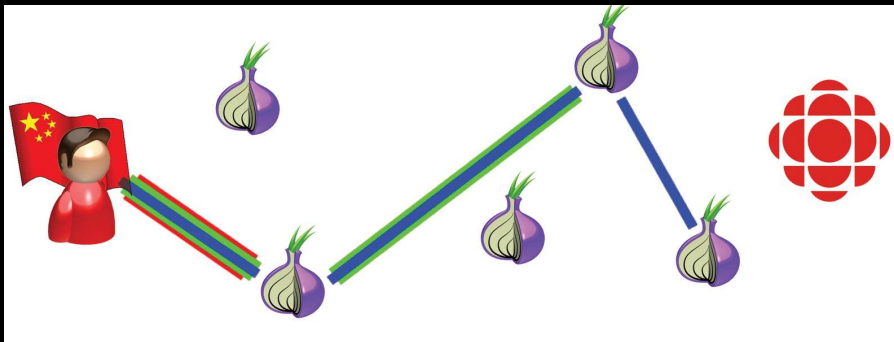
# Tor: circuit construction



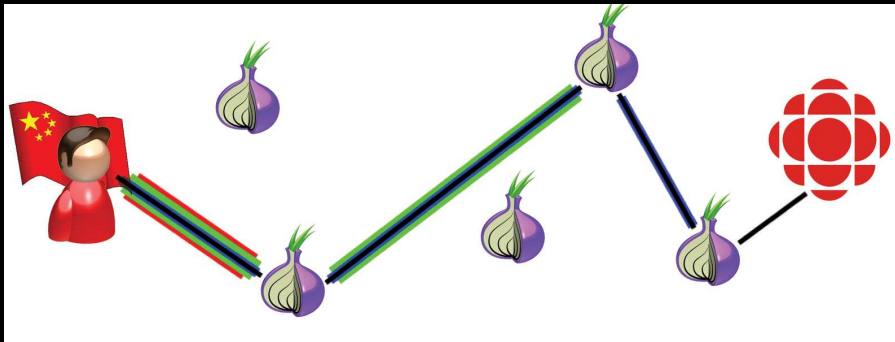
# Tor: circuit construction



# Tor: circuit construction



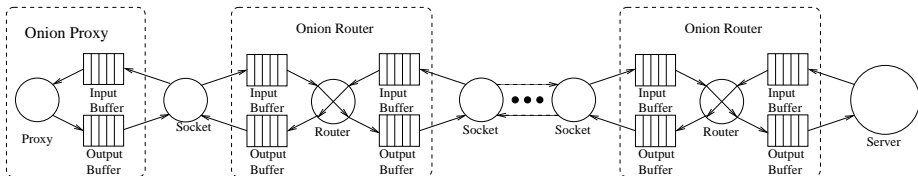
# Tor: circuit construction



- Privacy for usable, low-latency communication.
- However it can be slow, and that discourages casual usage.

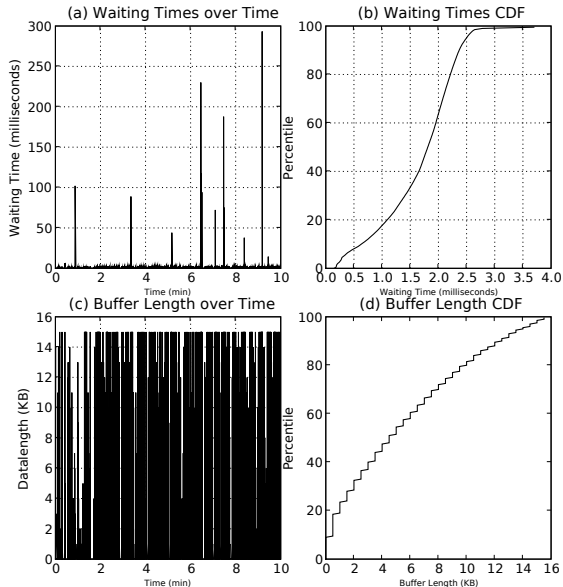
Where is the observed latency?

# Tor's Datapath



# Output buffers do introduce some latency

## Output Buffer Size and Latency

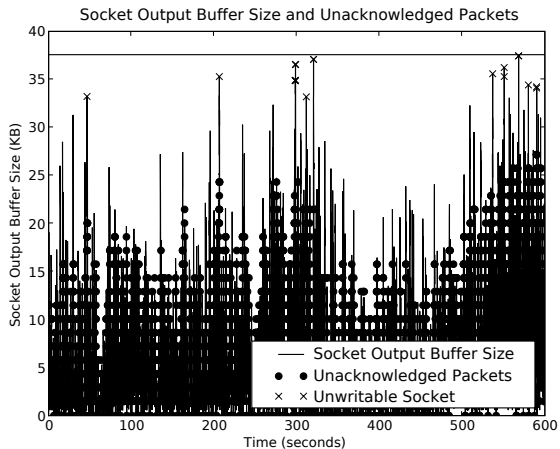


This occurs when the socket is unwritable

## A brief outline of TCP

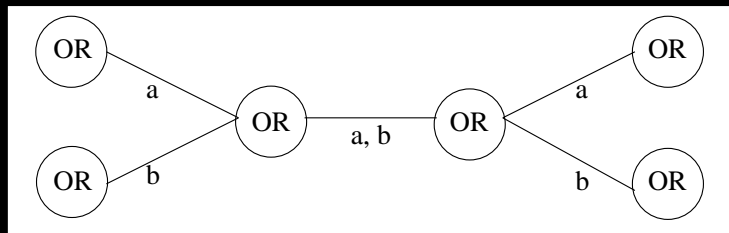
- TCP is designed to reliably send streams of data using packets
- Congestion controls throttles sending to maximize throughput while avoiding packet drops.

# Of what are TCP output buffers composed?



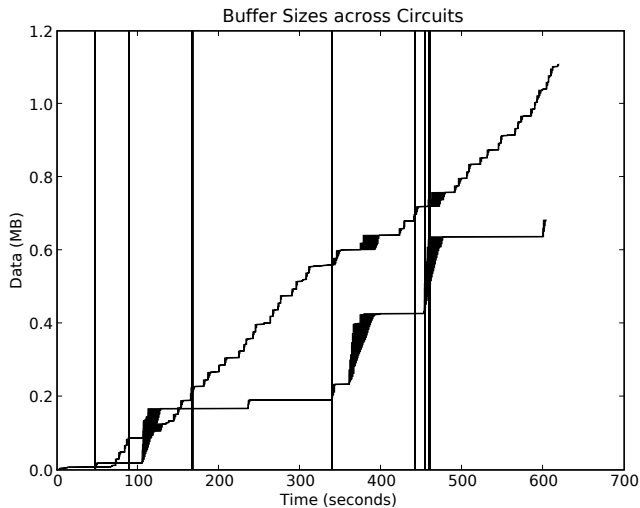
TCP Congestion Control (C/C) is to blame.

## Tor's multiplexing of circuits over TCP

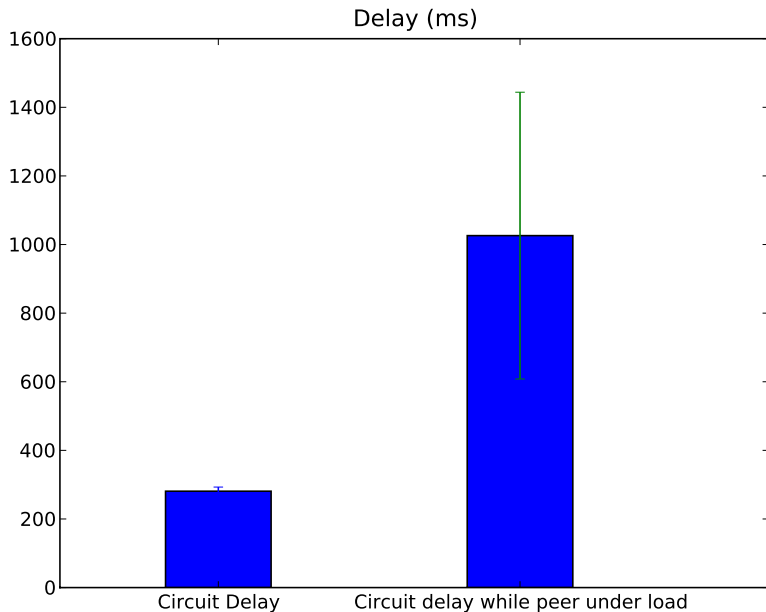


- If  $C/C$  is applied to  $a$ , then it is also applied to  $b$
- This is suboptimal; TCP is designed to throttle individual connections based on whether they witness a packet drop—proportional to their traffic.

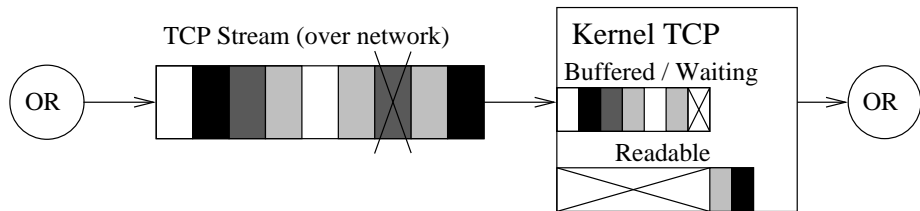
# An example of cross-circuit interference



# Experiment to observe interference by bulk senders



# Packet Dropping / Reordering



We want to use a separate TCP connection for each circuit

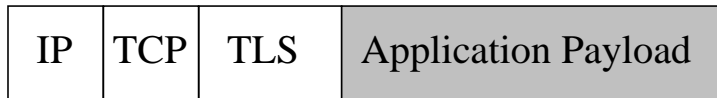
## Concerns for separate TCP connections

- Individual TCP streams leak precise information about the size and rate of data to an adversary
- Tor already faces some scalability concerns regarding its clique topology
- Some versions of Windows suffer when opening many TCP sockets already
- Any modification must be backwards compatible with the existing Tor network

## Our novel proposal: a TCP-over-DTLS tunnel

- DTLS - a secure (cf. TLS) protocol for transporting datagrams (UDP sockets)
- TCP implementation in user-space is used to generate TCP/IP packets, which are sent over DTLS
- The other end injects the received packet into their user-level TCP stack, and reads from user-level sockets

## How TCP-over-DTLS addresses our issues



(a) TCP Tor

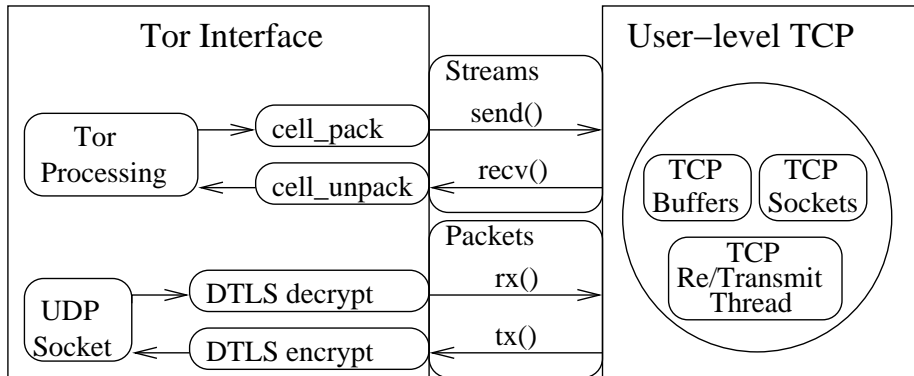


(b) TCP-over-DTLS Tor

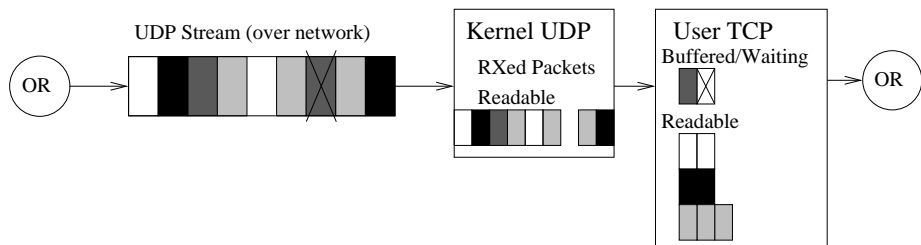
## How TCP-over-DTLS addresses our issues

- UDP operates in an unconnected mode, so it accepts packets from any destination
- Each node advertises a UDP socket that multiplexes data for all connections
- The sender is used to demultiplex the proper connection that is used to decrypt the DTLS payload
- Nodes that do not offer a UDP socket will use the existing transport, assuring backwards compatibility

# Packet / Streams translation

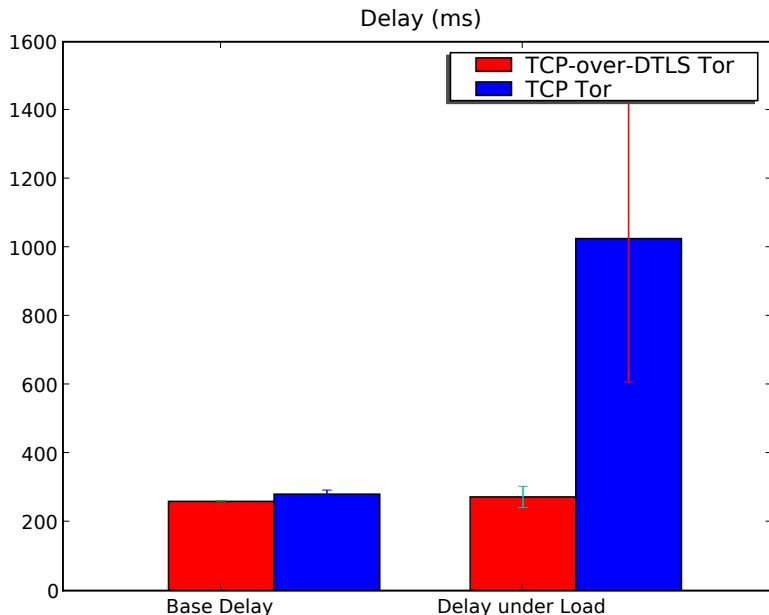


# How TCP-over-DTLS addresses our issues



Experimental results from our implementation

# Circuit latency comparison



# Future Work

# Improved Memory Management

## Cell Pool

0	1	2	3
4	5	6	7
8	9	10	11

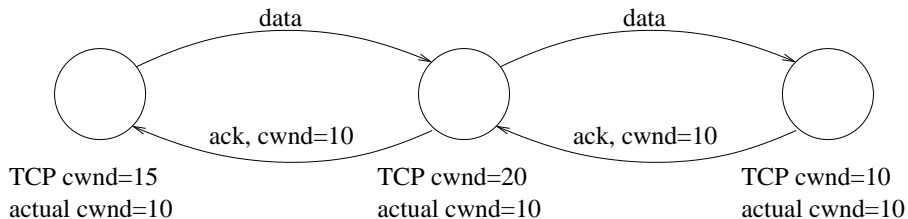
empty list: 2, 5, 9, 4

socket 1's input buffer: 0, 6, 7

socket 1's output buffer: 1, 3, 11, 8

cell\_t data: 10

# Back-propagation of Congestion Window



# Summary

- We determined that TCP congestion control introduces latency into Tor's datapath
- We determined that multiplexing circuits over TCP results in the unfair application of congestion control
- We proposed TCP-over-DTLS: a solution to address this issue that also addresses scalability issues and is backwards compatible with the existing Tor network
- We implemented our proposal and showed it successfully addressed cross-circuit interference.