

# SHARKFEST 2015

WIRESHARK DEVELOPER AND USER CONFERENCE



 COMPUTER HISTORY MUSEUM

# Why is my FTP Slooow?

# Agenda

- Throughput
- Sender delays
  - Segment size
  - Processing
- Receiver delays
  - Bytes in flight
  - Buffer size
- Congestion window
- SACK
- SSL

***A very  
practical  
look!***

# How Does An FTP Work?

- Control session (port 21)
- Data session (port 20)
- Active vs. passive FTP

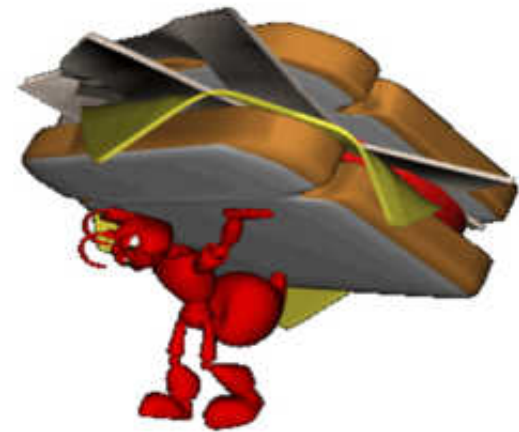
# Who Has What Job?

- Sender sends as fast as it can
- Network transports as fast as it can
- Receiver processes as fast as it can

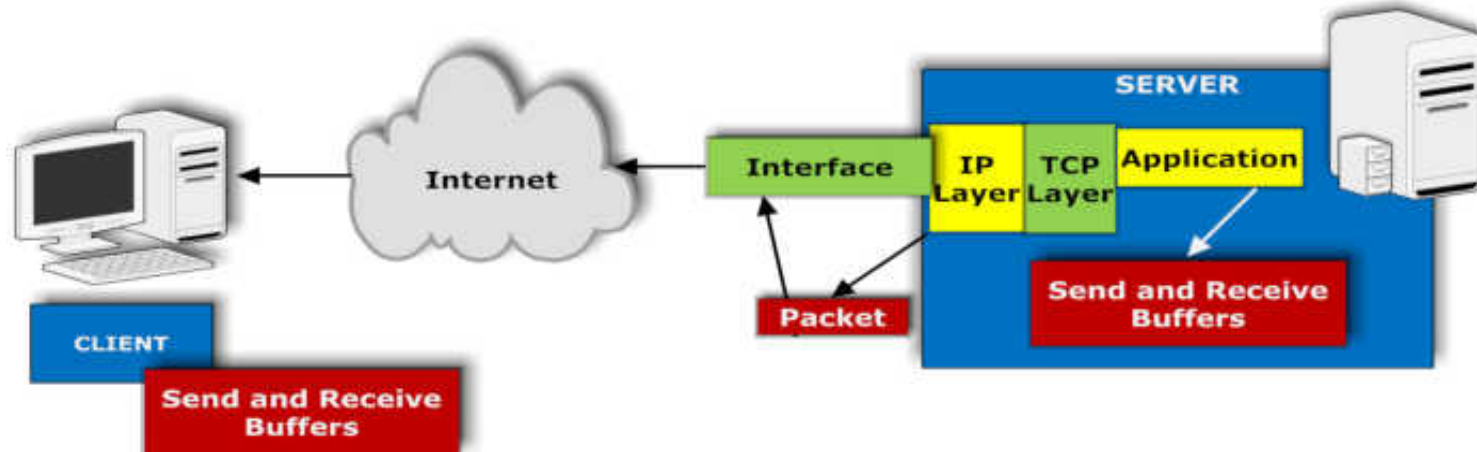


# Whose Fault Is It?

- When there is a problem, something does not have enough capacity
- (Or is slowing it down intentionally!)
- Which component is it?
- And... how do I fix it?



# Buffer / Network Structure



# Throughput

Source Address	Source Port	Destination Address	Destination Port	IP Protocol	Connection Time (ss.milli.micro)	Total Packets	Throughput Packets Per Second	Total Data Bytes	Throughput Data Bytes Per Second
38.109.217.115	20	76.231.80.42	61731	IPv4	102.141.228	14K (66.61%)	141	21M (100.0%)	206,008
76.231.80.42	61731	38.109.217.115	20	IPv4	102.071.452	7K (33.38%)	70	0 (0.0%)	-
-	-	-	-	-	-	21K	-	21M	-



- Source port 20 (FTP data: active)
  - Bytes per second: ~ 200,000
  - Total bytes sent: 21 million
- Destination port 20
  - No data bytes

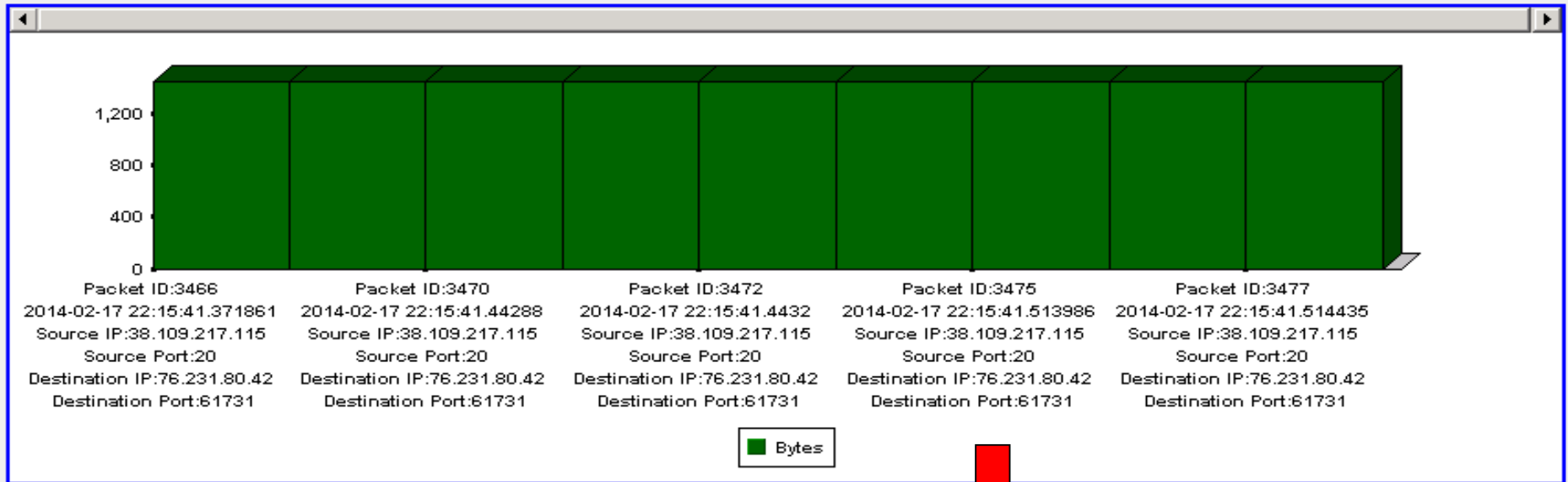
# Limits on the Sender

- Sender breaks things up into pieces to send out



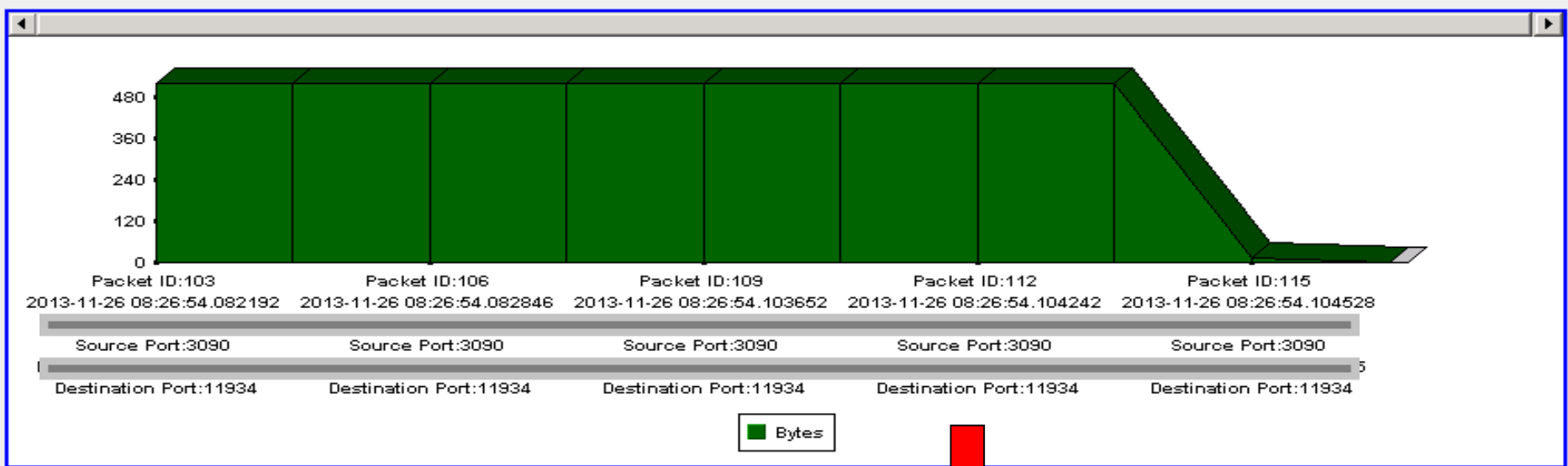
- What size? Does it matter?





-	-	Packet Number	Packet Date	Source Address	Source Port	Destination Address	Destination Port	Bytes
1		3466	2014-02-17 22:15:41.371861	38.109.217.115	20	76.231.80.42	61731	1,460
2		3467	2014-02-17 22:15:41.372079	38.109.217.115	20	76.231.80.42	61731	1,460
3		3470	2014-02-17 22:15:41.442880	38.109.217.115	20	76.231.80.42	61731	1,460
4		3471	2014-02-17 22:15:41.443057	38.109.217.115	20	76.231.80.42	61731	1,460
5		3472	2014-02-17 22:15:41.443200	38.109.217.115	20	76.231.80.42	61731	1,460
6		3473	2014-02-17 22:15:41.443375	38.109.217.115	20	76.231.80.42	61731	1,460
7		3475	2014-02-17 22:15:41.513986	38.109.217.115	20	76.231.80.42	61731	1,460
8		3476	2014-02-17 22:15:41.514284	38.109.217.115	20	76.231.80.42	61731	1,460
9		3477	2014-02-17 22:15:41.514435	38.109.217.115	20	76.231.80.42	61731	1,460
10		3478	2014-02-17 22:15:41.514635	38.109.217.115	20	76.231.80.42	61731	1,460



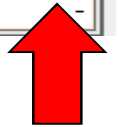


-	-	Packet Number	Packet Date	Source Address	Source Port	Destination Address	Destination Port	Bytes
1		103	2013-11-26 08:26:54.082192		3090		11934	524
2		104	2013-11-26 08:26:54.082580		3090		11934	524
3		106	2013-11-26 08:26:54.082846		3090		11934	524
4		107	2013-11-26 08:26:54.083196		3090		11934	524
5		109	2013-11-26 08:26:54.103652		3090		11934	524
6		110	2013-11-26 08:26:54.103930		3090		11934	524
7		112	2013-11-26 08:26:54.104242		3090		11934	524
8		113	2013-11-26 08:26:54.104332		3090		11934	524
9		115	2013-11-26 08:26:54.104528		3090		11934	9
10		119	2013-11-26 08:26:54.156370		3090		11934	0

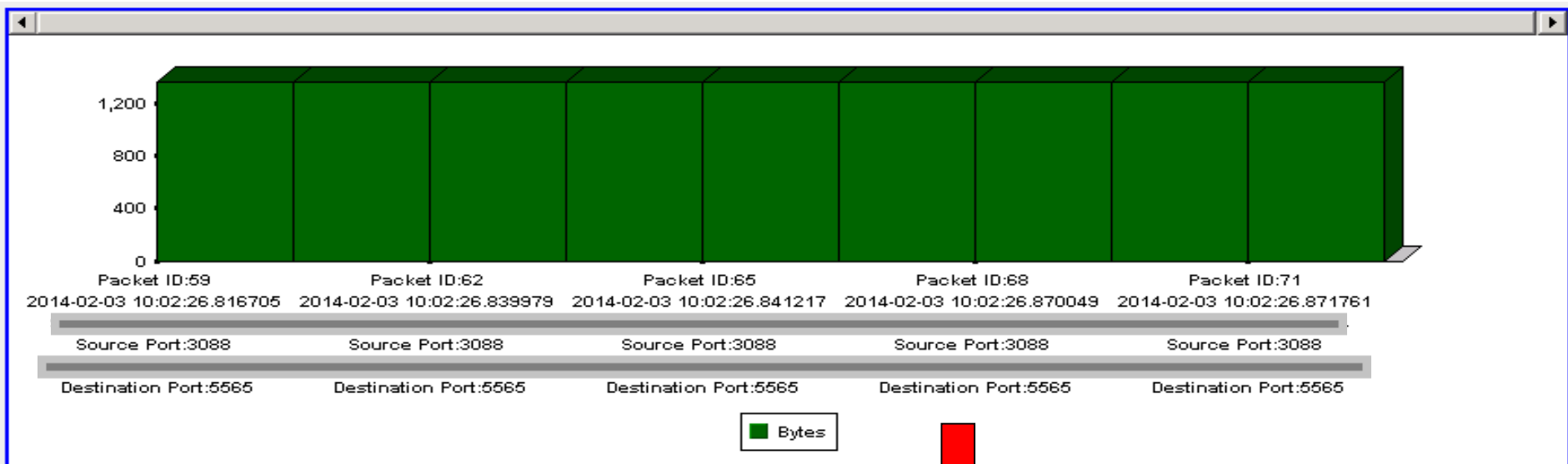


# Throughput

	Source Address	Source Port	Destination Address	Destination Port	IP Protocol	Connection Time (ss.milli.micro)	Total Packets	Throughput Packets Per Second	Total Data Bytes	Throughput Data Bytes Per Second
1		3090		11934	IPv4	116.652.092	9K (50.42%)	80	4M (99.99%)	41,948
2		11934		3090	IPv4	116.712.124	9K (49.57%)	78	418 (0.0%)	3
-	-	-	-	-	-	-	18K	-	4M	-



- Source port 3090 (passive FTP)
  - Bytes per second : ~ 42,000
- Destination port 3090
  - A few bytes sent
  - SSL session



		Packet Number	Packet Date	Source Address	Source Port	Destination Address	Destination Port	Bytes
1		59	2014-02-03 10:02:26.816705		3088		5565	1,368
2		61	2014-02-03 10:02:26.817119		3088		5565	1,368
3		62	2014-02-03 10:02:26.839979		3088		5565	1,368
4		64	2014-02-03 10:02:26.841011		3088		5565	1,368
5		65	2014-02-03 10:02:26.841217		3088		5565	1,368
6		67	2014-02-03 10:02:26.869649		3088		5565	1,368
7		68	2014-02-03 10:02:26.870049		3088		5565	1,368
8		70	2014-02-03 10:02:26.871239		3088		5565	1,368
9		71	2014-02-03 10:02:26.871761		3088		5565	1,368
10		73	2014-02-03 10:02:26.872301		3088		5565	1,368

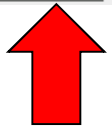


# Throughput

## After Segment Size Change

Source Address	Source Port	Destination Address	Destination Port	IP Protocol	Connection Time (ss.milli.micro)	Total Packets	Throughput Packets Per Second	Total Data Bytes	Throughput Data Bytes Per Second
	3088		5565	IPv4	66.995.292	4K (48.17%)	72	6M (99.99%)	99,285
	5565		3088	IPv4	67.304.420	5K (51.82%)	77	418 (0.0%)	6
-	-	-	-	-	-	9K	-	6M	-

- Source port 3088 (passive FTP)
- Bytes per second : ~ 100,000



## Before Segment Size Change

	Source Address	Source Port	Destination Address	Destination Port	IP Protocol	Connection Time (ss.milli.micro)	Total Packets	Throughput Packets Per Second	Total Data Bytes	Throughput Data Bytes Per Second
1		3090		11934	IPv4	116.652.092	9K (50.42%)	80	4M (99.99%)	41,948
2		11934		3090	IPv4	116.712.124	9K (49.57%)	78	418 (0.0%)	3
-	-	-	-	-	-	-	18K	-	4M	-

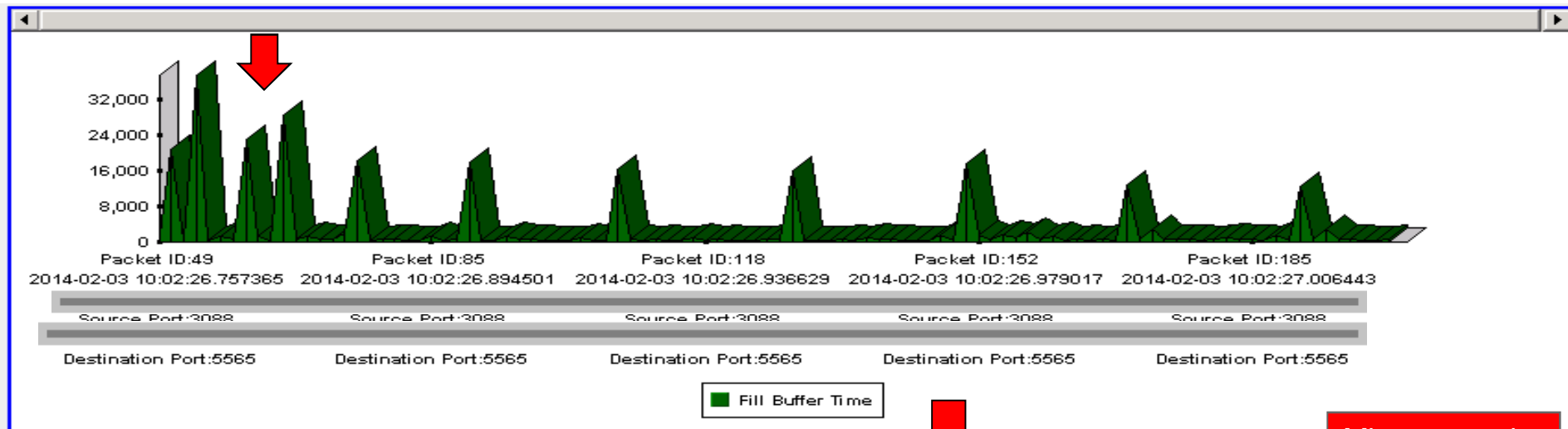
- Source port 3090 (passive FTP)
- Bytes per second : ~ 42,000



# Limits on the Sender

- Sender breaks things up into pieces to send out
- It takes him time to get the pieces ready and then to send them



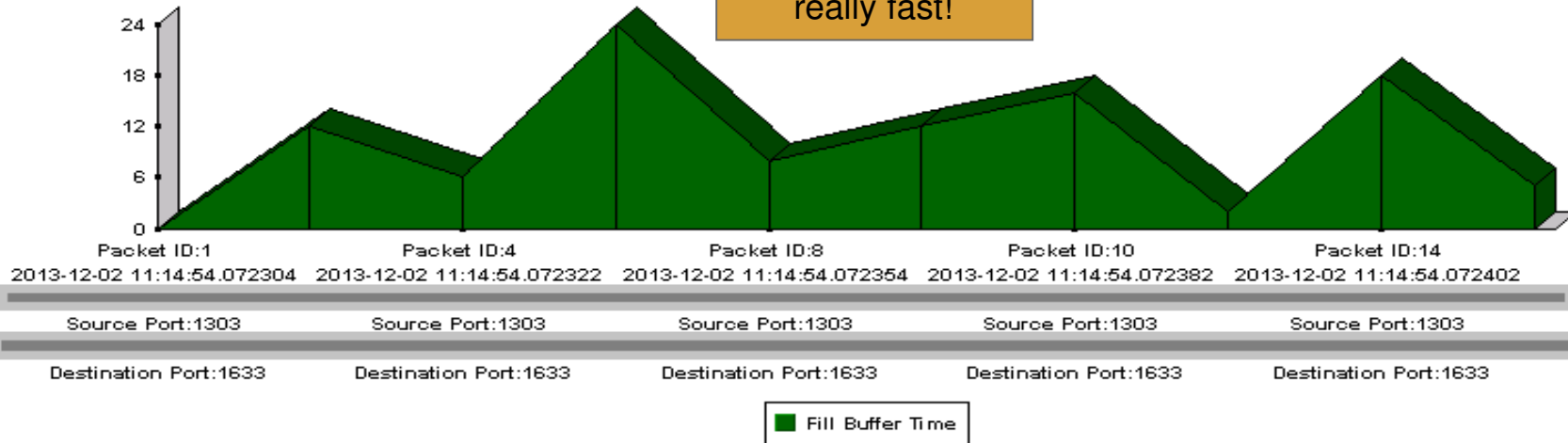


Microseconds

		Packet Number	Packet Date	Source Address	Source Port	Destination Address	Destination Port	Bytes	Fill Buffer Time
1		49	2014-02-03 10:02:26.757365		3088		5565	1,368	1,102
2		51	2014-02-03 10:02:26.777907		3088		5565	1,368	20,542
3		52	2014-02-03 10:02:26.778007		3088		5565	97	100
4		57	2014-02-03 10:02:26.815559		3088		5565	0	37,552
5		58	2014-02-03 10:02:26.815719		3088		5565	6	160
6		59	2014-02-03 10:02:26.816705		3088		5565	1,368	986
7		61	2014-02-03 10:02:26.817119		3088		5565	1,368	414
8		62	2014-02-03 10:02:26.839979		3088		5565	1,368	22,860
9		64	2014-02-03 10:02:26.841011		3088		5565	1,368	1,032
10		65	2014-02-03 10:02:26.841217		3088		5565	1,368	206
11		67	2014-02-03 10:02:26.869649		3088		5565	1,368	28,432
12		68	2014-02-03 10:02:26.870049		3088		5565	1,368	400
13		70	2014-02-03 10:02:26.871239		3088		5565	1,368	1,190



Filling buffer  
really fast!



Microseconds

-	-	Packet Number	Packet Date	Source Address	Source Port	Destination Address	Destination Port	Bytes	Fill Buffer Time
1		1	2013-12-02 11:14:54.072304		1303		1633	1,460	0
2		3	2013-12-02 11:14:54.072316		1303		1633	1,460	12
3		4	2013-12-02 11:14:54.072322		1303		1633	1,460	6
4		6	2013-12-02 11:14:54.072346		1303		1633	1,460	24
5		8	2013-12-02 11:14:54.072354		1303		1633	1,460	8
6		9	2013-12-02 11:14:54.072366		1303		1633	1,460	12
7		10	2013-12-02 11:14:54.072382		1303		1633	1,460	16
8		11	2013-12-02 11:14:54.072384		1303		1633	1,460	2
9		14	2013-12-02 11:14:54.072402		1303		1633	1,460	18
10		15	2013-12-02 11:14:54.072407		1303		1633	1,460	5



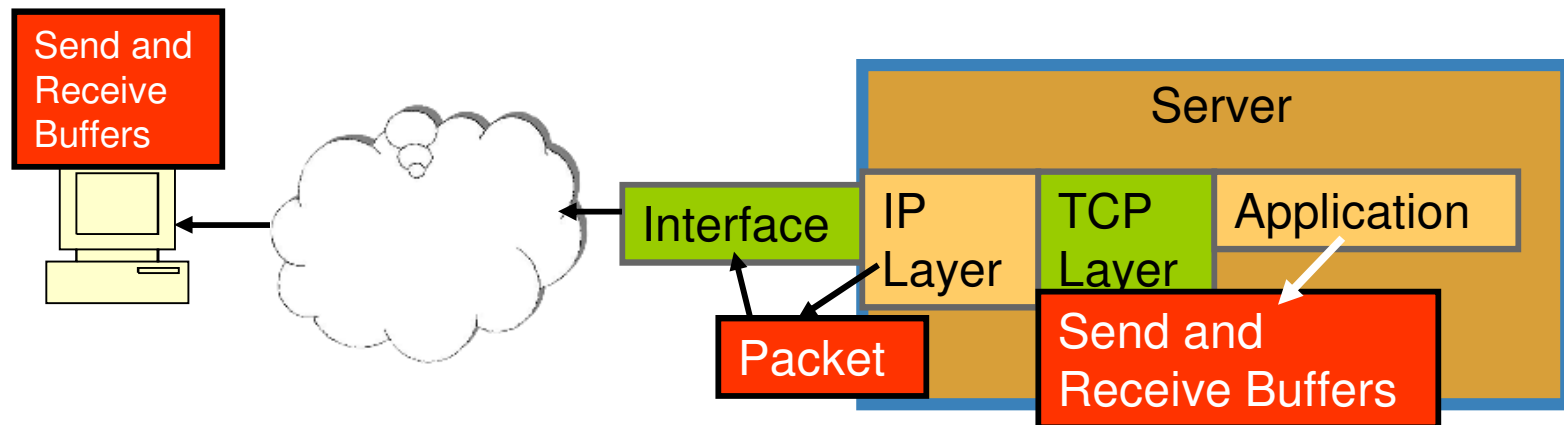
# Filling the Buffer

- Processing time
- Other side says stop sending
- TCP algorithms
  - slow start,
  - congestion avoidance,
  - fast retransmit, and
  - fast recovery

# TCP Buffers and MSS

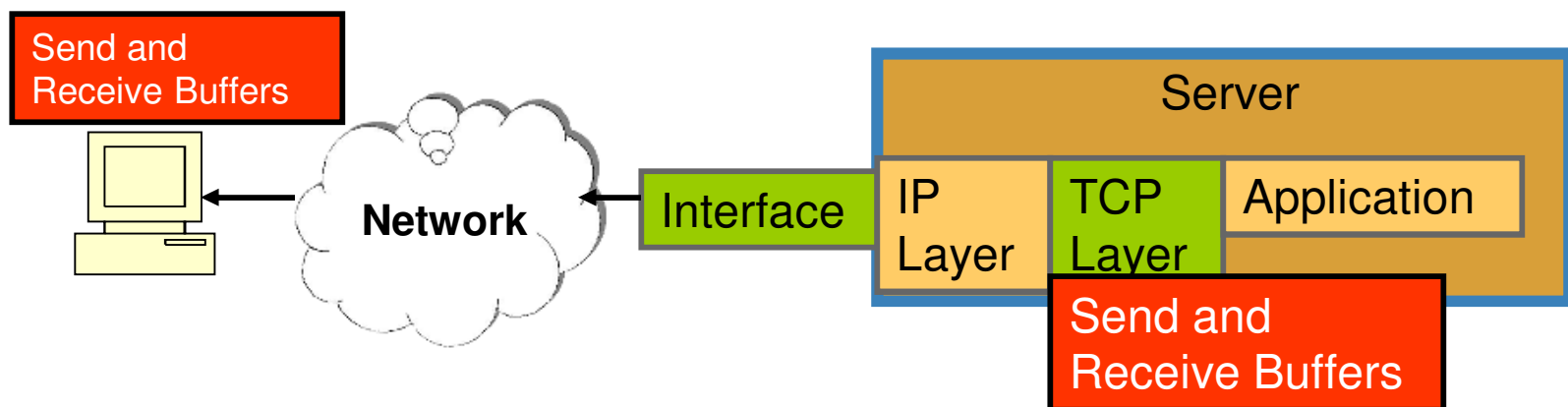
- In the TCP Open, the TCP Send / Receive buffers and MSS are set.
- The Send / Receive buffers are pools used to hold data prior to sending it out across the physical adapter.
- TCP Send and Receive Buffers can affect speed of transmission

- Max Segment Size
  - Can affect speed of transmission
  - Can be different in each direction.
- On most platforms each application can explicitly set the buffer size via a socket option call.



# How Should Buffers Be Set?

- Complicated area to provide recommendations
  - Kind of applications you are using,
  - Kind of network
  - Client window settings
- For applications which send a lot of data (FTP) across a high bandwidth network, then may want to make MTU sizes and buffer sizes as large as possible (IF other end can absorb the traffic)



# TCP Congestion Control - History

“In October of '86, the Internet had the first of what became a series of ‘congestion collapses’. During this period, the data throughput from LBL to UC Berkeley (sites separated by 400 yards and two IMPhops) dropped from 32 Kbps to 40 bps.”



Van Jacobson

“Congestion Avoidance and Control”

Van Jacobson Lawrence Berkeley Laboratory

Michael J. Karels University of California at Berkeley

November, 1988

# Congestion Control Solution

“In particular, we wondered if the 4.3BSD (Berkeley UNIX) TCP was mis-behaving or if it could be tuned to work better under abysmal network conditions. The answer to both of these questions was “yes”.

- TCP congestion control : each source determines network capacity
- ACK : Round Trip Time signals

# Current Work

- RFC 5681 : TCP Congestion Control
- This document defines TCP's four intertwined congestion control algorithms: slow start, congestion avoidance, fast retransmit, and fast recovery. In addition, the document specifies how TCP should begin transmission after a relatively long idle period, as well as discussing various acknowledgment generation methods.

# Max Segment - Window

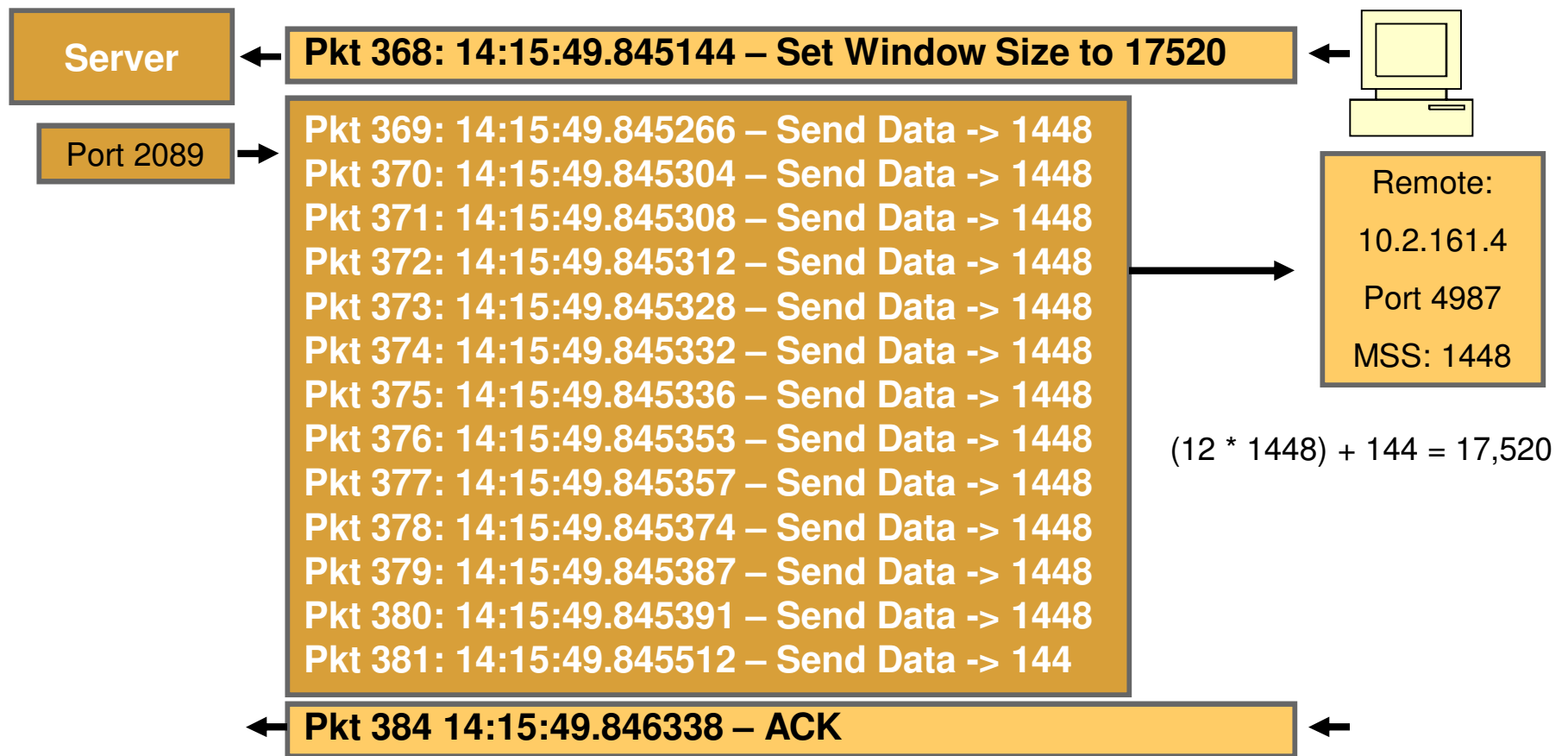
```
Filter: tcp.flags.syn == 1
Expression... Clear Apply Save

No.      Time                Source                Destination           Protocol  Info
-----
1176 52.537614000    74.125.228.71        192.168.2.11         TCP      443→58924 [SYN, ACK] Seq=0

Header Length: 32 bytes
... 0000 0001 0010 = Flags: 0x012 (SYN, ACK)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion Window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...1 = Acknowledgment: Set
.... .... 0... = Push: Not set
.... .... .0.. = Reset: Not set
... ..1. = Syn: Set
.... .... ...0 = Fin: Not set
Window size value: 42900 ←
[Calculated window size: 42900]
Checksum: 0x6789 [validation disabled]
Urgent pointer: 0
Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP),
... Maximum segment size: 1430 bytes ←
... No-Operation (NOP)
... No-Operation (NOP)
```




























-- SYN Packet: TCP Open Connection.  
-- Will send out 30 packets at 1,430 before waiting for an ACK.

# Filling Window





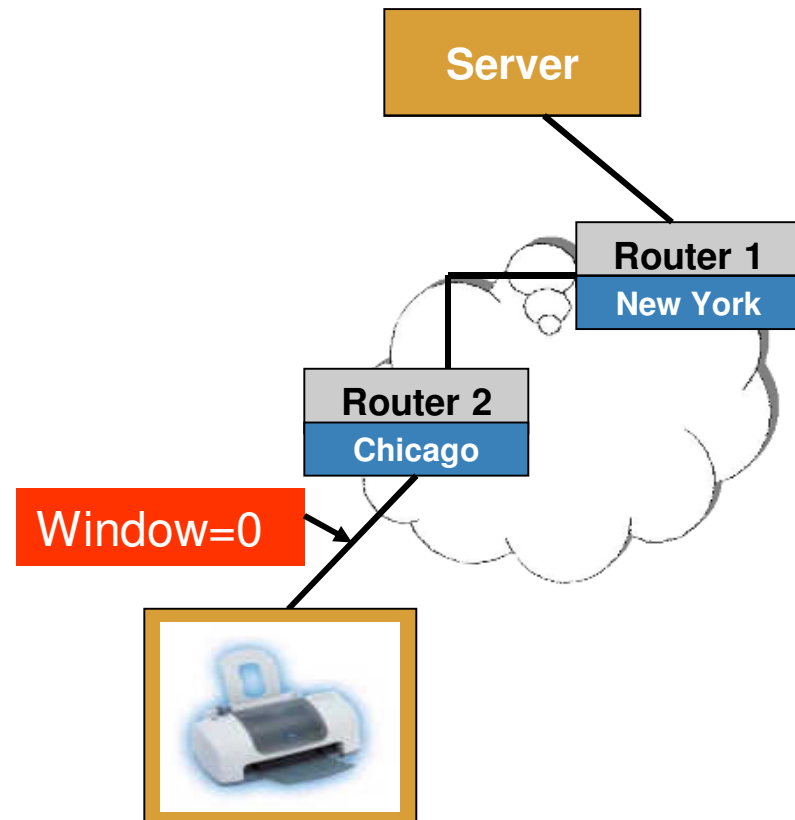
# ACKs will catch up

464	465	466	467	468	469	478	479	480
192.168.1.231	192.168.1.231	192.168.1.231	192.168.1.231	192.168.1.231	192.168.1.231	192.168.1.100	192.168.1.100	192.168.1.100
1036	1036	1036	1036	1036	1036	3713	3713	3713
								
MF: Did you get 6470?	MF: Did you get 7930?	MF: Did you get 9390?	MF: Did you get 0850?	MF: Did you get 2310?	MF: Did you get 2670?	PC: Got 6470	PC: Got 4570	PC: Got 7490
						-	-	-
1460 	1460 	1460 	1460 	1460 	360 	 0	 0	 0
-	-	-	-	-	-			

- You may see many packets sent with a small interval in between.
- Then, all the ACKs will follow.

# Congestion Window = 0

- Window size = 0 : stop sending
- Printers:
  - Paper jam in the printer
  - Printer is out of paper



# High Performance Networking Options

- **TCP Selective Acknowledgments (SACK, RFC2018):** SACKs allow receiver to acknowledge non-consecutive data
- **Large Windows (RFC1323):** Allows windows greater than 64K

# Window Size Limit (RFC1323)

- TCP header uses 16 bit field for receive window size
- Largest window is  $2^{16} = 65K$  bytes.
- TCP option, "Window Scale", defined to allow windows larger than  $2^{16}$
- Option defines an implicit scale factor used to multiply the window size value to obtain the true window size
- With RFC 1323, *TCP Extensions for High Performance*, maximum window size increased to 1,073,741,823 bytes
- Both sides of link must support window scaling or the default of 65,535 bytes will apply as maximum window size

# How Window Scaling is Sent

- The three-byte Window Scale option may be sent in a SYN segment by a TCP. It has two purposes:

- (1) indicate that the TCP is prepared to do both send and receive window scaling, and
- (2) communicate a scale factor to be applied to its receive window.

- Thus, a TCP that is prepared to scale windows should send the option, even if its own scale factor is 1. The scale factor is limited to a power of two and encoded logarithmically, so it may be implemented by binary shift operations.

**TCP Window Scale Option (WSopt):**

**Kind: 3 Length: 3 bytes**

```
+-----+-----+-----+
| Kind=3 |Length=3 |shift.cnt|
+-----+-----+-----+
```

# Window Scaling Both Ways

- Option is an offer, not a promise; both sides must send Window Scale options in their SYN segments to enable window scaling in either direction
- If window scaling enabled, then the sender will right-shift its true receive-window values by 'shift.cnt' bits
- 'shift.cnt' may be zero
- Window scaling can be done in either or both directions

```
Window scale: 7 (multiply by 128)
Kind: Window Scale (3) ←
Length: 3
Shift count: 7 ←
[Multiplier: 128] ←
```

# Window Scale Values

- The scaling parameter is base 2.
- Normal window size allows  $2^{16}$  bytes
- Maximum scaling factor is 14
- Maximum window size =  $2^{30}$  bytes
- Window becomes maximum of 1 gigabyte
- Window scale option of 1  $\rightarrow 2^{17}$

2.	2.2	=	4
3.	4.2	=	8
4.	8*2	=	16
5.	16*2	=	32
6.	32*2	=	64
7.	64*2	=	128
8.	128*2	=	256
9.	256*2	=	512
10.	512*2	=	1,024
11.	1,024*2	=	2,048
12.	2,048*2	=	4,096
13.	4,096*2	=	8,192
14.	8,192*2	=	16,384
15.	16,384*2	=	32,768
16.	32,768*2	=	65,536
17.	65,536*2	=	131,072

# SACK / BIF

- What is SACK?
- How does ACK work?
- Out of Order / Retransmit
- SACK packets
- How to calculate Bytes in Flight(BIF)
- Why?



# From RFC2018

- RFC2018 defines Selective Acknowledgement (SACK)

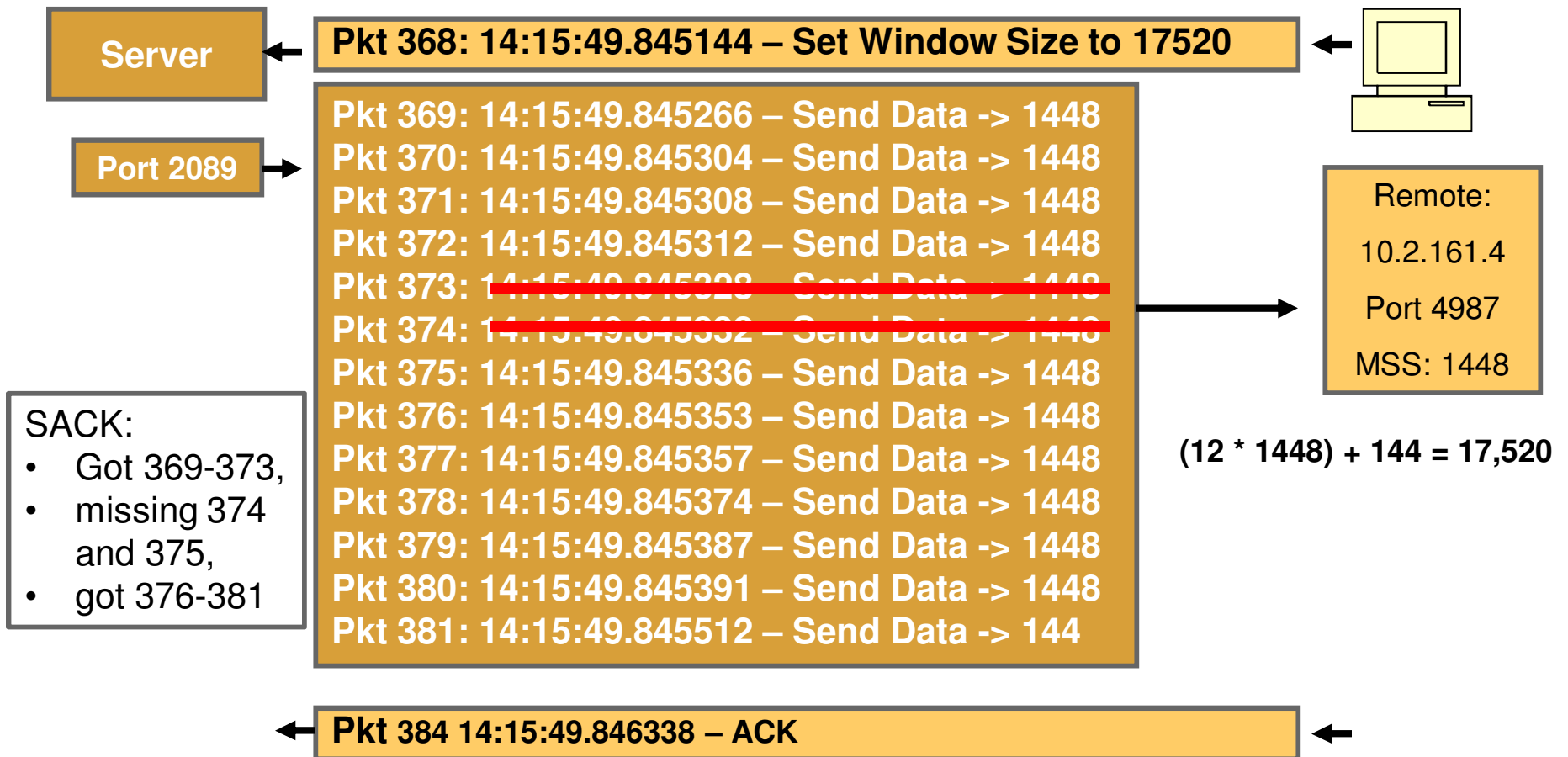
“TCP may experience poor performance when multiple packets are lost from one window of data. With the limited information available from cumulative acknowledgements, a TCP sender can only learn about a **single lost packet per round trip time**. An aggressive sender could choose to retransmit packets early, but such retransmitted segments may have already been successfully received.”

# From RFC2018

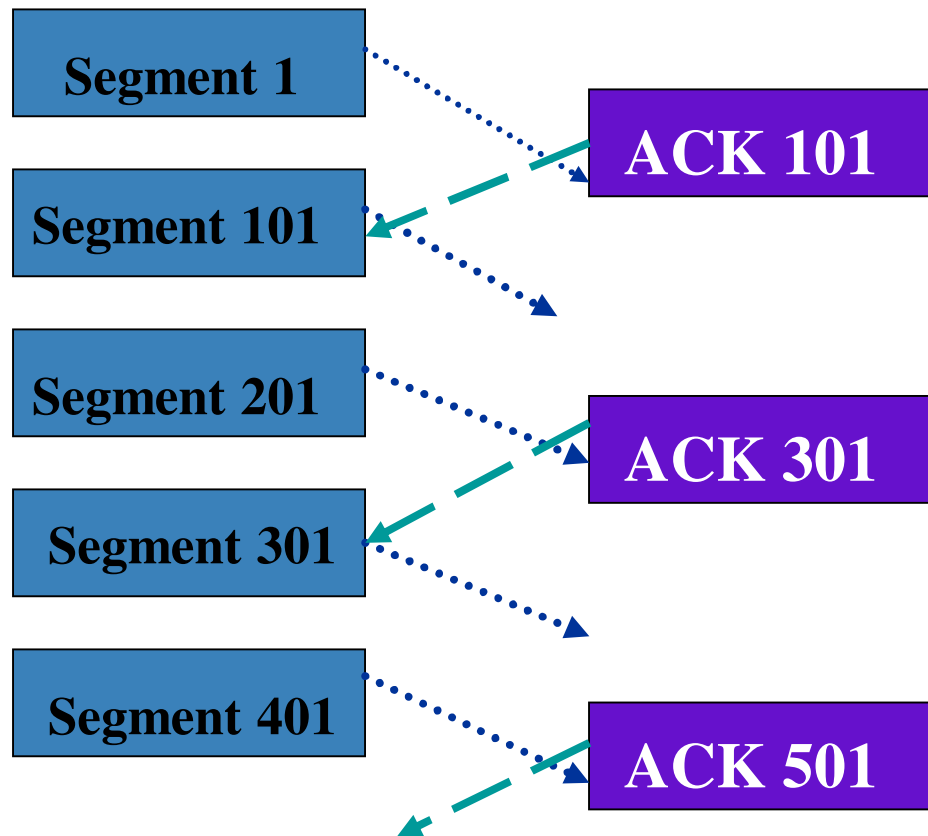
- So, what is the solution?

“A Selective Acknowledgment (SACK) mechanism, combined with a selective repeat retransmission policy, can help to overcome these limitations. The receiving TCP sends back SACK packets to the sender informing the sender of data that has been received. The sender can then retransmit only the missing data segments.”

# Let's go back to our example

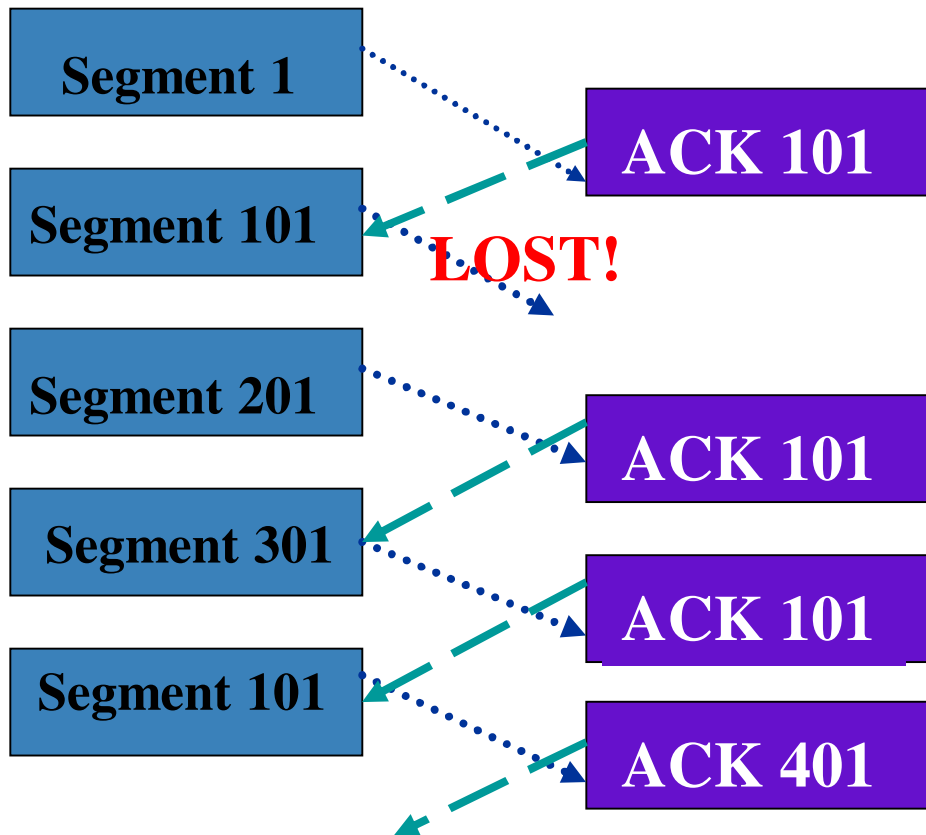


# How Does ACK Work?



- *With the limited information available from cumulative acknowledgements, a TCP sender can only learn about a single lost packet per round trip time.*
- Assume each segment has 100 bytes.
- ACK is for all the bytes received and indicates the next byte of data that is expected. This is the 'cumulative ACK' in the RFC above.

# Duplicate ACK - Retransmits



- Assume each segment has 100 bytes.
- ACK is for the next byte of data it is waiting for.
- A duplicate ACK is sent when a packet is received and the sequence number indicates that it does not contain the byte you are waiting for.
- After 3 duplicate ACKs, the packet is retransmitted.

# Let's Look at Some Packets

Source Address	Source Port	Destination Address	Destination Port	IP ID	Data Length	Sequence	ACK	Expected ACK
10.201.0.2	23	192.168.145.7	1080	81	3	2247247727	3010274481	2247247730
192.168.145.7	1080	10.201.0.2	23	27B	3	3010274481	2247247730	3010274484
10.201.0.2	23	192.168.145.7	1080	FB	3	2247247730	3010274484	2247247733
192.168.145.7	1080	10.201.0.2	23	27C	3	3010274484	2247247733	3010274487
10.201.0.2	23	192.168.145.7	1080	100	6	2247247733	3010274487	2247247739
192.168.145.7	1080	10.201.0.2	23	27D	18	3010274487	2247247739	3010274505

Sequence Number + Data Length = Expected ACK

# Multiple Packets in a Window

Source Address	Source Port	Destination Address	Destination Port	IP ID	Data Length	Sequence	ACK	Expected ACK	Protocol
10.197.4.1	58315	10.173.12.5	4726	5BEE	1260	2167520027	2822526086	2167521287	ACK
10.197.4.1	58315	10.173.12.5	4726	5BEF	1260	2167521287	2822526086	2167522547	ACK
10.197.4.1	58315	10.173.12.5	4726	5BF0	1260	2167522547	2822526086	2167523807	ACK, PSH



Sequence Number + Data Length = Next Sequence Number (Expected ACK)

# From RFC2018

- RFC2018 defines Selective Acknowledgement (SACK) as follows:
- TCP may experience poor performance when multiple packets are lost from one window of data. *With the limited information available from cumulative acknowledgements, a TCP sender can only learn about a single lost packet per round trip time. An aggressive sender could choose to retransmit packets early, but such retransmitted segments may have already been successfully received.*



# Out of Order

Source Address	Source Port	Destination Address	Destination Port	IP ID	Data Length	Sequence	ACK	Expected ACK	Protocol	Comment
10.197.4.1	58315	10.173.12.5	4726	5BF8	1260	2167531999	2822526086	2167533259	ACK	
10.173.12.5	4726	10.197.4.1	58315	5345	0	2822526086	2167533259	0	ACK	
10.197.4.1	58315	10.173.12.5	4726	5BF9	1260	2167533259	2822526086	2167534519	ACK	
10.197.4.1	58315	10.173.12.5	4726	5BFB	632	2167535779	2822526086	2167536411	ACK, PSH	TCP Out of Order. Expected Seq: 2167534519 In Pkt: 462

- What was expected in packet #4 is sequence # 2167534519.
- What was actually received is sequence # 2167535779
- The packet containing bytes 2167534519 through 2167535778 is lost (or not yet received).
- Since packet #4 contains a HIGHER sequence number than expected, it is noted as being out of order.


# SACK will keep track of Out of Order

Source Address	Source Port	Destination Address	Destination Port	IP ID	Data Length	Sequence	ACK	Expected ACK	Protocol	Comment
10.197.4.1	58315	10.173.12.5	4726	5BF9	1260	2167533259	2822526086	2167534519	ACK	
10.197.4.1	58315	10.173.12.5	4726	5BFB	632	2167535779	2822526086	2167536411	ACK, PSH	TCP Out of Order. Expected Seq:2167534519 In Pkt:462
10.173.12.5	4726	10.197.4.1	58315	5346	0	2822526086	2167534519	0	ACK	SACK: SEQ: 2167535779 (8131f8a3) 2167536411 (8131fb1b)

- SACK packets will contain the sequence numbers of the out of order packets received.
- May be multiple blocks (multiple packets received out of order).

# The SACK Fields are in TCP Options

```
⊕ .... ..1. = Syn: Set
  .... ..0 = Fin: Not set
Window size value: 8192
[Calculated window size: 8192]
⊕ Checksum: 0x0df2 [validation disabled]
Urgent pointer: 0
⊖ Options: (12 bytes), Maximum segment size,
  ⊕ Maximum segment size: 1460 bytes
  ⊕ No-Operation (NOP)
  ⊖ Window scale: 2 (multiply by 4)
    Kind: window Scale (3)
    Length: 3
    Shift count: 2
    [Multiplier: 4]
  ⊕ No-Operation (NOP)
  ⊕ No-Operation (NOP)
  ⊖ TCP SACK Permitted Option: True
    Kind: SACK Permitted (4)
    Length: 2
```



# SACK in Action!

```
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), SACK
  No-Operation (NOP)
  No-Operation (NOP)
  SACK: 1242401778-1242401779
    Kind: SACK (5)
    Length: 10
    left edge = 1242401778
    right edge = 1242401779
    [TCP SACK Count: 1]
  [SEQ/ACK analysis]
  [TCP Analysis Flags]
    [This is a TCP duplicate ack]
    [Duplicate ACK #: 1]
  [Duplicate to the ACK in frame: 4141]
```

- Missing ...1778-1779


# Multiple Out of Order Packets in SACK

Source Address	Source Port	Destination Address	Destination Port	IP ID	Data Length	Sequence	ACK	Expected ACK	Protocol	Comment
10.173.12.5	4726	10.197.4.1	58315	534A	0	2822526086	2167534519	0	ACK	TCP Dup Ack. Same as packet:465 SACK: SEQ: 2167535779 (8131f8a3) 2167541451 (81320ecb)
10.197.4.1	58315	10.173.12.5	4726	5C01	1260	2167542711	2822526086	2167543971	ACK	TCP Out of Order. Expected Seq:2167541451 In Pkt:472
10.173.12.5	4726	10.197.4.1	58315	534B	0	2822526086	2167534519	0	ACK	TCP Dup Ack. Same as packet:465 SACK: SEQ: 2167542711 (813213b7) 2167543971 (813218a3), 2167535779 (8131f8a3) 2167541451 (81320ecb)

# Bytes in Flight

- RFC2581 is first version of TCP Congestion Control (updated by RFC5681). Defines:
- **FLIGHT SIZE:** The amount of data that has been sent but not yet acknowledged.
- Why does this matter?


# Example #1



Source Address	Source Port	Destination Address	Destination Port	IP ID	Data Length	Sequence	ACK	Expected ACK	Delta (ss.milli.micro)	Bytes In Flight Sender	Bytes Acked This Packet
208.111.39.67	5902	24.6.68.48	1315	5A3F	4380	3899888273	1990669247	3899892653	0.001.852	4,380	0
208.111.39.67	5902	24.6.68.48	1315	5A42	606	3899892653	1990669247	3899893259	0.000.007	4,986	0
208.111.39.67	5902	24.6.68.48	1315	5A43	8760	3899893259	1990669247	3899902019	0.003.140	13,746	0
24.6.68.48	1315	208.111.39.67	5902	8782	0	1990669247	3899897639	0	0.027.942	0	13,746

- Sender sent packets with data length of 4,380, 606, and 8760 without waiting very long.
- Then, to get the ACK from the receiver, 27 milliseconds elapsed.
- The idea is to keep the pipe filled.

# Example #2



Source Address	Source Port	Destination Address	Destination Port	IP ID	Data Length	Sequence	ACK	Expected ACK	Delta (ss.milli.micro)	Bytes In Flight Sender
10.197.4.1	58315	10.173.12.5	4726	5BEE	1260	2167520027	2822526086	2167521287	0.004.378	1,260
10.197.4.1	58315	10.173.12.5	4726	5BEF	1260	2167521287	2822526086	2167522547	0.000.057	2,520
10.173.12.5	4726	10.197.4.1	58315	5340	0	2822526086	2167522547	0	0.000.031	0
10.197.4.1	58315	10.173.12.5	4726	5BF0	1260	2167522547	2822526086	2167523807	0.000.035	1,260
10.197.4.1	58315	10.173.12.5	4726	5BF1	1260	2167523807	2822526086	2167525067	0.000.579	2,520
10.173.12.5	4726	10.197.4.1	58315	5341	0	2822526086	2167525067	0	0.000.033	0

- In this example, the ACKs are coming very quickly, so even though the bytes in flight are low, there does not appear to be a lot of waiting.



# Bytes In Flight Helps Determine Window Size


- When deciding upon the most efficient window size for your network, the two most important considerations are:
  - round-trip latency and
  - the end-to-end bandwidth on the network.
- Both elements determine the amount of data that can be on the network ("in-flight") at any given time.



















# Throughput / BIF

Source Address	Source Port	Connection Time (ss.milli.micro)	Total Packets	Throughput Packets Per Second	Total Data Bytes	Throughput Data Bytes Per Second	Minimum Bytes In Flight	Average Bytes In Flight	Maximum Bytes In Flight
10.19.0.200	21	11.807.996	25 (1.52%)	2	922 (0.1%)	83	0	38	186
10.19.0.200	38571	0.081.236	6 (0.36%)	6	3K (0.44%)	3,812	0	1K	2K
10.19.0.200	56783	0.119.630	473 (28.91%)	473	551K (64.58%)	551,369	0	1K	2K

- We like to look at throughput as well as bytes in flight.
- Complicated area!

# Bytes In Flight / Receive Buffer



-	-	Packet Number	Packet Date	Source Address	Source Port	Destination Address	Destination Port	Bytes	Bytes in Flight	Receive Buffer Size
1		3466	2014-02-17 22:15:41.371861	38.109.217.115	20	76.231.80.42	61731	1,460	1,460	8,192
2		3467	2014-02-17 22:15:41.372079	38.109.217.115	20	76.231.80.42	61731	1,460	2,920	8,192
3		3470	2014-02-17 22:15:41.442880	38.109.217.115	20	76.231.80.42	61731	1,460	1,460	17,408
4		3471	2014-02-17 22:15:41.443057	38.109.217.115	20	76.231.80.42	61731	1,460	2,920	17,408
5		3472	2014-02-17 22:15:41.443200	38.109.217.115	20	76.231.80.42	61731	1,460	4,380	17,408
6		3473	2014-02-17 22:15:41.443375	38.109.217.115	20	76.231.80.42	61731	1,460	5,840	17,408
7		3475	2014-02-17 22:15:41.513986	38.109.217.115	20	76.231.80.42	61731	1,460	4,380	17,408
8		3476	2014-02-17 22:15:41.514284	38.109.217.115	20	76.231.80.42	61731	1,460	5,840	17,408
9		3477	2014-02-17 22:15:41.514435	38.109.217.115	20	76.231.80.42	61731	1,460	7,300	17,408
10		3478	2014-02-17 22:15:41.514635	38.109.217.115	20	76.231.80.42	61731	1,460	8,760	17,408
11		3480	2014-02-17 22:15:41.515809	38.109.217.115	20	76.231.80.42	61731	1,460	7,300	17,408
12		3481	2014-02-17 22:15:41.515957	38.109.217.115	20	76.231.80.42	61731	1,460	8,760	17,408
13		3482	2014-02-17 22:15:41.516115	38.109.217.115	20	76.231.80.42	61731	1,460	10,220	17,408
14		3483	2014-02-17 22:15:41.516258	38.109.217.115	20	76.231.80.42	61731	1,460	11,680	17,408
15		3487	2014-02-17 22:15:41.584863	38.109.217.115	20	76.231.80.42	61731	1,460	10,220	17,408
16		3488	2014-02-17 22:15:41.585073	38.109.217.115	20	76.231.80.42	61731	1,460	11,680	17,408
17		3489	2014-02-17 22:15:41.585212	38.109.217.115	20	76.231.80.42	61731	1,460	13,140	17,408
18		3490	2014-02-17 22:15:41.585350	38.109.217.115	20	76.231.80.42	61731	1,460	14,600	17,408

# Limits on the Sender

- Sender breaks things up into pieces to send out
- It takes him time to send the pieces
- Sender can only send as much as the other side can absorb



# Compare IPv6 and IPv4

Destination Address	IP Protocol	Connection Time (ss.milli.micro)	Total Packets	Throughput Packets Per Second	Total Data Bytes	Throughput Data Bytes Per Second	Minimum Bytes In Flight	Average Bytes In Flight	Maximum Bytes In Flight
208.111.39.67	IPv4	35.327.127	2K (52.01%)	65	11K (0.57%)	324	0	7	28
208.111.39.67	IPv4	0.020.015	5 (0.11%)	5	480 (0.02%)	480	0	96	480
208.111.39.67	IPv4	0.019.998	5 (0.11%)	5	480 (0.02%)	480	0	96	480
208.111.39.67	IPv4	5.371.834	8 (0.18%)	1	6K (0.31%)	1,218	0	761	1K
2607:F740::3F:216:3EFF:FE68:72C0	IPv6	20.940.750	11 (0.25%)	< 0	9K (0.5%)	495	0	908	8K
2607:F740::3F:216:3EFF:FE68:72C0	IPv6	20.792.770	8 (0.18%)	< 0	9K (0.48%)	474	0	1K	8K
2607:F740::3F:216:3EFF:FE68:72C0	IPv6	20.797.733	9 (0.2%)	< 0	1K (0.06%)	60	0	139	579
2607:F740::3F:216:3EFF:FE68:72C0	IPv6	20.795.714	8 (0.18%)	< 0	1K (0.09%)	90	0	234	1K
2607:F740::3F:216:3EFF:FE68:72C0	IPv6	20.793.697	8 (0.18%)	< 0	1K (0.06%)	60	0	156	579
2607:F740::3F:216:3EFF:FE68:72C0	IPv6	20.741.833	8 (0.18%)	< 0	9K (0.5%)	497	0	1K	8K
2607:F740::3F:216:3EFF:FE68:72C0	IPv6	6.440.179	76 (1.73%)	12	42K (2.15%)	7,026	0	1K	7K
2607:F740::3F:216:3EFF:FE68:72C0	IPv6	6.108.390	69 (1.57%)	11	2K (0.12%)	419	0	47	1K

- As integration of IPv6 happens, it is interesting to compare both protocols in terms of throughput and bytes in flight.

# IPv4 vs. IPv6 FTP



Source Address	Source Port	Destination Address	Destination Port	IP Protocol	Connection Time (ss.milli.micro)	Total Packets	Throughput Packets Per Second	Total Data Bytes	Throughput Data Bytes Per Second
38.109.217.115	20	76.231.80.42	61731	IPv4	102.141.228	14K (66.61%)	141	21M (100.0%)	206,008
76.231.80.42	61731	38.109.217.115	20	IPv4	102.071.452	7K (33.38%)	70	0 (0.0%)	-
-	-	-	-	-	-	21K	-	21M	-



Source Address	Source Port	Destination Address	Destination Port	IP Protocol	Connection Time (ss.milli.micro)	Total Packets	Throughput Packets Per Second	Total Data Bytes	Throughput Data Bytes Per Second
2605:6F00:877::31DA:4B3B	20	2001:5C0:1000:A::1563	61749	IPv6	65.798.625	17K (66.14%)	265	21M (99.14%)	327,692
2001:5C0:1000:A::1563	61749	2605:6F00:877::31DA:4B3B	20	IPv6	65.669.597	8K (33.85%)	135	183K (0.85%)	2,817
-	-	-	-	-	-	26K	-	21M	-



# Bytes in Flight






























Source Address	Source Port	Destination Address	Destination Port	IP Protocol	Connection Time (ss.milli.micro)	Total Packets	Minimum Bytes In Flight	Average Bytes In Flight	Maximum Bytes In Flight
38.109.217.115	20	76.231.80.42	61731	IPv4	102.141.228	14K (66.61%)	0	15K	17K
76.231.80.42	61731	38.109.217.115	20	IPv4	102.071.452	7K (33.38%)	0	0	0
-	-	-	-	-	-	21K	-	-	-



Source Address	Source Port	Destination Address	Destination Port	IP Protocol	Connection Time (ss.milli.micro)	Total Packets	Minimum Bytes In Flight	Average Bytes In Flight	Maximum Bytes In Flight
2605:6F00:877::31DA:4B3B	20	2001:5C0:1000:A::1563	61749	IPv6	65.798.625	17K (66.14%)	0	42K	65K
2001:5C0:1000:A::1563	61749	2605:6F00:877::31DA:4B3B	20	IPv6	65.669.597	8K (33.85%)	0	0	0
-	-	-	-	-	-	26K	-	-	-





















# Receive Buffer Full!

-	-	Packet Number	Packet Date	Source Address	Source Port	Destination Address	Destination Port	Bytes	Bytes in Flight	Receive Buffer Size	
1		5338	2014-02-17 22:15:49.005881	38.109.217.115	20	76.231.80.42	61731	1,460	16,060	17,408	
2		5340	2014-02-17 22:15:49.018754	38.109.217.115	20	76.231.80.42	61731	1,460	14,600	17,408	
3		5341	2014-02-17 22:15:49.019013	38.109.217.115	20	76.231.80.42	61731	1,460	16,060	17,408	
4		5343	2014-02-17 22:15:49.024007	38.109.217.115	20	76.231.80.42	61731	1,460	14,600	17,408	
5		5344	2014-02-17 22:15:49.024393	38.109.217.115	20	76.231.80.42	61731	1,460	16,060	17,408	
6		5346	2014-02-17 22:15:49.037733	38.109.217.115	20	76.231.80.42	61731	1,460	14,600	17,408	
7		5347	2014-02-17 22:15:49.037931	38.109.217.115	20	76.231.80.42	61731	1,460	16,060	17,408	
8		5349	2014-02-17 22:15:49.067876	38.109.217.115	20	76.231.80.42	61731	1,460	14,600	17,408	
9		5350	2014-02-17 22:15:49.068057	38.109.217.115	20	76.231.80.42	61731	1,460	16,060	17,408	
10		5351	2014-02-17 22:15:49.068621	38.109.217.115	20	76.231.80.42	61731	1,348	17,408	17,408	
11		5353	2014-02-17 22:15:49.074916	38.109.217.115	20	76.231.80.42	61731	1,460	15,948	17,408	
12		5354	2014-02-17 22:15:49.075095	38.109.217.115	20	76.231.80.42	61731	1,460	17,408	17,408	
13		5356	2014-02-17 22:15:49.088302	38.109.217.115	20	76.231.80.42	61731	1,460	15,948	17,408	
14		5357	2014-02-17 22:15:49.088483	38.109.217.115	20	76.231.80.42	61731	1,460	17,408	17,408	
15		5359	2014-02-17 22:15:49.093578	38.109.217.115	20	76.231.80.42	61731	1,460	15,948	17,408	
16		5360	2014-02-17 22:15:49.093797	38.109.217.115	20	76.231.80.42	61731	1,460	17,408	17,408	
17		5363	2014-02-17 22:15:49.107212	38.109.217.115	20	76.231.80.42	61731	1,460	15,948	17,408	
18		5364	2014-02-17 22:15:49.107384	38.109.217.115	20	76.231.80.42	61731	1,460	17,408	17,408	



# Impact?

-	-	Packet Number	Packet Date	Source Address	Source Port	Destination Address	Destination Port	Bytes	Bytes in Flight	Receive Buffer Size	Fill Buffer Time
1		5338	2014-02-17 22:15:49.005881	38.109.217.115	20	76.231.80.42	61731	1,460	16,060	17,408	224
2		5340	2014-02-17 22:15:49.018754	38.109.217.115	20	76.231.80.42	61731	1,460	14,600	17,408	12,873
3		5341	2014-02-17 22:15:49.019013	38.109.217.115	20	76.231.80.42	61731	1,460	16,060	17,408	259
4		5343	2014-02-17 22:15:49.024007	38.109.217.115	20	76.231.80.42	61731	1,460	14,600	17,408	4,994
5		5344	2014-02-17 22:15:49.024393	38.109.217.115	20	76.231.80.42	61731	1,460	16,060	17,408	386
6		5346	2014-02-17 22:15:49.037733	38.109.217.115	20	76.231.80.42	61731	1,460	14,600	17,408	13,340
7		5347	2014-02-17 22:15:49.037931	38.109.217.115	20	76.231.80.42	61731	1,460	16,060	17,408	198
8		5349	2014-02-17 22:15:49.067876	38.109.217.115	20	76.231.80.42	61731	1,460	14,600	17,408	29,945
9		5350	2014-02-17 22:15:49.068057	38.109.217.115	20	76.231.80.42	61731	1,460	16,060	17,408	181
10		5351	2014-02-17 22:15:49.068621	38.109.217.115	20	76.231.80.42	61731	1,348	17,408	17,408	564
11		5353	2014-02-17 22:15:49.074916	38.109.217.115	20	76.231.80.42	61731	1,460	15,948	17,408	6,295
12		5354	2014-02-17 22:15:49.075095	38.109.217.115	20	76.231.80.42	61731	1,460	17,408	17,408	179
13		5356	2014-02-17 22:15:49.088302	38.109.217.115	20	76.231.80.42	61731	1,460	15,948	17,408	13,207
14		5357	2014-02-17 22:15:49.088483	38.109.217.115	20	76.231.80.42	61731	1,460	17,408	17,408	181
15		5359	2014-02-17 22:15:49.093578	38.109.217.115	20	76.231.80.42	61731	1,460	15,948	17,408	5,095
16		5360	2014-02-17 22:15:49.093797	38.109.217.115	20	76.231.80.42	61731	1,460	17,408	17,408	219
17		5363	2014-02-17 22:15:49.107212	38.109.217.115	20	76.231.80.42	61731	1,460	15,948	17,408	13,415
18		5364	2014-02-17 22:15:49.107384	38.109.217.115	20	76.231.80.42	61731	1,460	17,408	17,408	172



# It could be the network!

- RTT
- Retransmissions
- Duplicate acknowledgements / segments
- Out of order packets

# Doing it Securely

- Sometimes security can be configured incorrectly
- Will slow down traffic because of unneeded overhead
- How?



# SSL Application Data Packet

No.	Time	Source	Destination	Protocol	Length	Identification
5162	210.620942000	173.194.34.100	10.32.165.157	TLSv1.2	95	0xba29 (47657)

⊕	Frame 5162: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on interface 0
⊕	Ethernet II, Src: IETF-VRRP-VRID_69 (00:00:5e:00:01:69), Dst: IntelCor_20:1a:d0 (9c:4e:36:20:1a:d0)
⊕	Internet Protocol Version 4, src: 173.194.34.100 (173.194.34.100), Dst: 10.32.165.157 (10.32.165.157)
⊕	Transmission Control Protocol, Src Port: https (443), Dst Port: 59622 (59622), Seq: 4582, Ack: 1788, Len: 41
⊖	Secure sockets Layer
⊖	<b>TLSv1.2 Record Layer: Application Data Protocol: http</b>
	Content Type: Application Data (23)
	Version: TLS 1.2 (0x0303)
	Length: 36
	Encrypted Application Data: 000000000000000640fbb320309dbf1be9e51f958137dced...

0000	9c 4e 36 20 1a d0 00 00 5e 00 01 69 08 00 45 00	.N6 .... ^..i..E.
0010	00 51 ba 29 00 00 3a 06 46 9a ad c2 22 64 0a 20	.Q.)...: F... "d.
0020	a5 9d 01 bb e8 e6 68 f1 ed 66 5d 6b b4 95 50 18	.....h. .f]k..P.
0030	02 95 80 34 00 00 17 03 03 00 24 00 00 00 00 00	...4... . .\$.....
0040	00 00 06 40 fb b3 20 30 9d bf 1b e9 e5 1f 95 81	...@.. 0 .....
0050	37 dc ed 10 f1 ea aa 54 21 f4 f9 58 98 0f 51	7.....T !..X..Q

- Application data record has the actual data

# SSL Application Record Overhead

No.	Time	Source	Destination	Protocol	Length	Identification
5162	210.620942000	173.194.34.100	10.32.165.157	TLSv1.2	95	0xba29 (47657)

⊕ Frame 5162: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on interface 0	
⊕ Ethernet II, Src: IETF-VRRP-VRID_69 (00:00:5e:00:01:69), Dst: IntelCor_20:1a:d0 (9c:4e:36:20:1a:d0)	
⊕ Internet Protocol Version 4, Src: 173.194.34.100 (173.194.34.100), Dst: 10.32.165.157 (10.32.165.157)	
⊕ Transmission Control Protocol, Src Port: https (443), Dst Port: 59622 (59622), Seq: 4582, Ack: 1788, Len: 41	
⊖ Secure Sockets Layer	
⊖ TLSv1.2 Record Layer: Application Data Protocol: http	
Content Type: Application Data (23)	
Version: TLS 1.2 (0x0303)	
Length: 36	
Encrypted Application Data: 000000000000000640fbb320309dbf1be9e51f958137dced...	

0000	9c 4e 36 20 1a d0 00 00	5e 00 01 69 08 00 45 00	.N6 .... A..f..E.
0010	00 51 ba 29 00 00 3a 06	46 9a ad c2 22 64 0a 20	.Q.).... F..."d.
0020	a5 9d 01 bb e8 e6 00 00	00 00 00 00 00 00 00 00	.....h. .f]k..P.
0030	02 95 80 34 00 00	17 03 03 00 24 00 00 00 00 00	...4.... ..\$. ....
0040	00 00 06 40 fb b3 20 30	30 81 1b e9 e5 1f 95 81	...@.. 0 .....
0050	37 dc ed 10 f1 ea aa 54	21 f4 f9 58 98 0f 51	7.....T !..X..Q

- Overhead with each data record

Secure Sockets Layer

```
TLV1 Record Layer: Handshake Protocol: Encrypted Handshake Message
  Content Type: Handshake (22)
  Version: TLS 1.0 (0x0301)
  Length: 36
  Handshake Protocol: Encrypted Handshake Message
TLV1 Record Layer: Application Data Protocol: Application Data
  Content Type: Application Data (23)
  Version: TLS 1.0 (0x0301)
  Length: 21
  Encrypted Application Data: d22f3e155448dc89b6c8a38a69c5764eb27d4288f6
TLV1 Record Layer: Application Data Protocol: Application Data
  Content Type: Application Data (23)
  Version: TLS 1.0 (0x0301)
  Length: 21
  Encrypted Application Data: c5122f341a2bdc8276f12799fbcfc6f39220946b37
TLV1 Record Layer: Application Data Protocol: Application Data
  Content Type: Application Data (23)
  Version: TLS 1.0 (0x0301)
  Length: 21
  Encrypted Application Data: 2aa944eba04cf680c0d2b7c15851c1355bb8b81660
TLV1 Record Layer: Application Data Protocol: Application Data
  Content Type: Application Data (23)
  Version: TLS 1.0 (0x0301)
  Length: 21
  Encrypted Application Data: 3d50bade465c59ffc40aabeb87ae8c69b6ce7a87b6
TLV1 Record Layer: Application Data Protocol: Application Data
  Content Type: Application Data (23)
  Version: TLS 1.0 (0x0301)
  Length: 21
  Encrypted Application Data: 80bb2ab4f8317680b8e92be8ec59e3ffbfdd4e7571
TLV1 Record Layer: Application Data Protocol: Application Data
  Content Type: Application Data (23)
  Version: TLS 1.0 (0x0301)
  Length: 21
  Encrypted Application Data: 09aa30b68debae4b4714ca9bc9c136fb01416e8d47
TLV1 Record Layer: Application Data Protocol: Application Data
  Content Type: Application Data (23)
  Version: TLS 1.0 (0x0301)
  Length: 21
  Encrypted Application Data: 3c6383717eaf23da03e32fa92376343e5c718c14b1
TLV1 Record Layer: Application Data Protocol: Application Data
```

Port 3088  
Sending about 50  
application data  
records per  
packet at 21 bytes  
each.

1 byte of data –  
20 bytes of  
overhead!

# Summary

- It will only get worse!
- More SSL!



# Contact Info

- Nalini Elkins
- [Nalini.elkins@insidethestack.com](mailto:Nalini.elkins@insidethestack.com)
- (831) 659-8360
  
- Love to hear from you!