

# SHARKFEST '12

Wireshark Developer and User Conference

## Christian Landström

Senior Consultant, Fast Lane GmbH  
IT Security, Network Analysis

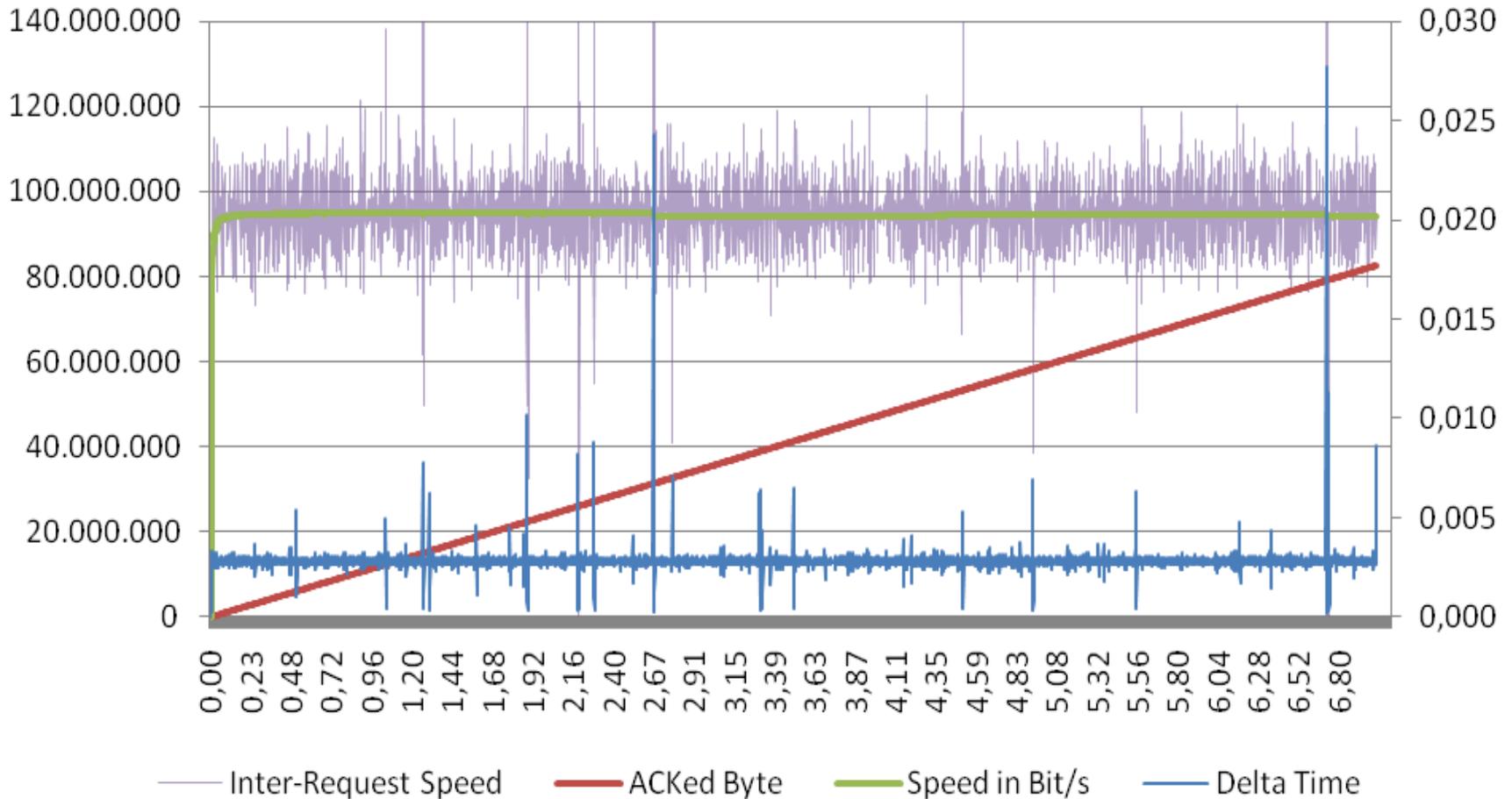
# The story...

- Customer just went through migration from WindowsXP to Vista plus implemented VoIP
- Afterwards experiencing heavy packet loss on infrastructure switches
- Users complaining that „everything is slow“

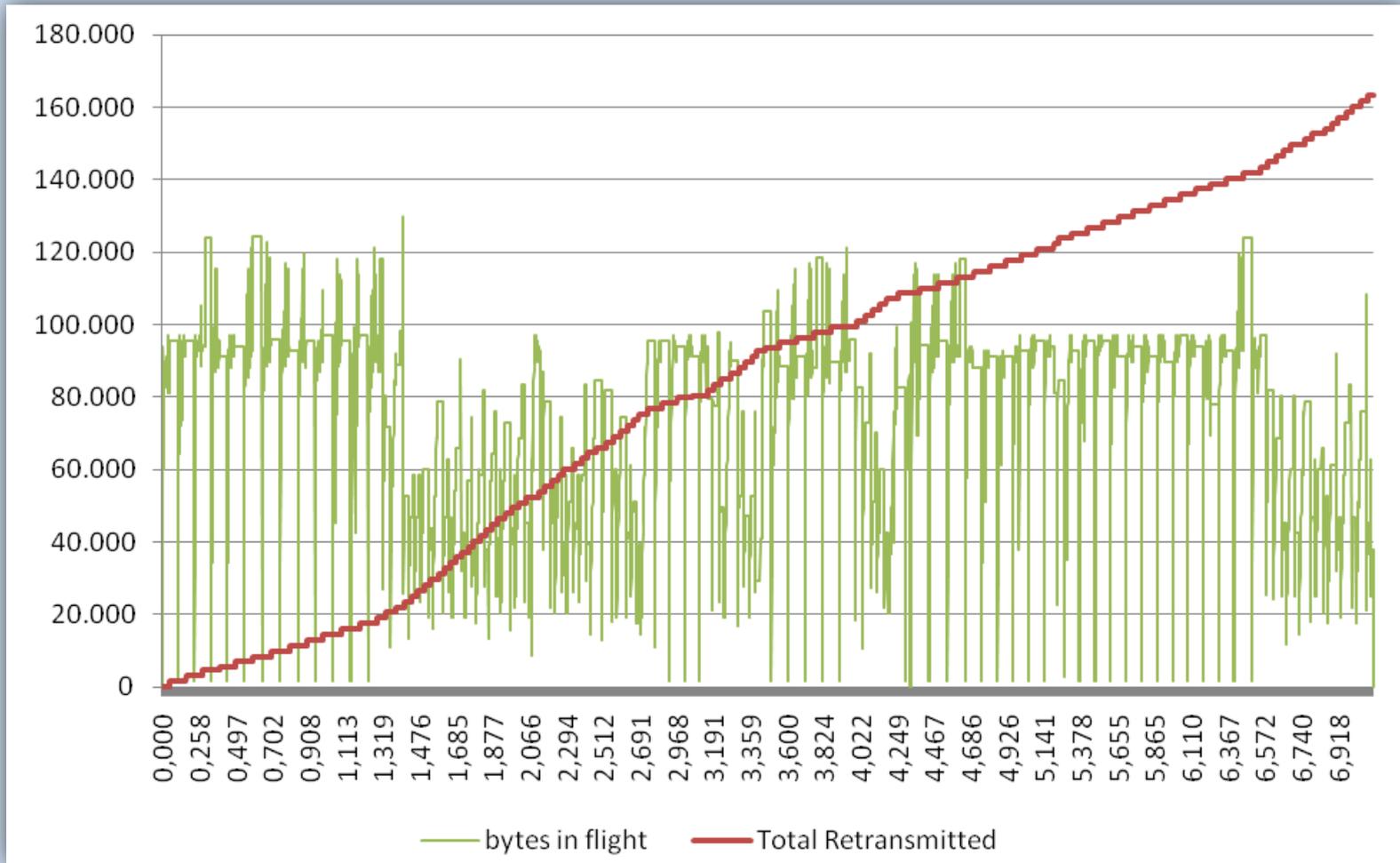
# Verifying that there is a problem

- Test plan with certain standard tasks
  - Uploads
  - Downloads
  - Mail attachments
  - ...
- Sniffing traffic at three points
  - Core
  - Distribution to Access
  - Client

# Performance Problem ?



# Constant packet loss



# Reaction to packet loss

## Sending side

- Retransmit
  - When?
  - What?
  - How fast?
- Continue sending
  - When?
- Ignore the fact a packet got lost?

## Receiving side

- Request more data?
- Request less data?
- Adjust receive window?

# TCP Receive Window

- 16 bit Value in the TCP Header
- Ranges from 0 to 65535
- Works as a receive buffer for incoming TCP payload bytes
- Adjusts the data transmission rate of the sender
  - specifies the maximum of bytes in travel
- Wireshark monitors rwnd (e.g. „TCP Window Full“)

# TCP „Send Window“

- Does it exist at all?
- If it does, since when?
- What is it good for anyway?
- Where could you observe the Send Window size in Wireshark?
- More details to come...

# Other TCP „windows“ ?

- Receiver Window (rwnd)
  - The most recently advertised receiver window
- Congestion Window (cwnd)
  - A TCP state variable that limits the amount of data a TCP can send. At any given time, a TCP MUST NOT send data with a sequence number higher than the sum of the highest acknowledged sequence number and the minimum of cwnd and rwnd
- Initial Window (IW)
  - The initial window is the size of the sender's congestion window after the three-way handshake is completed

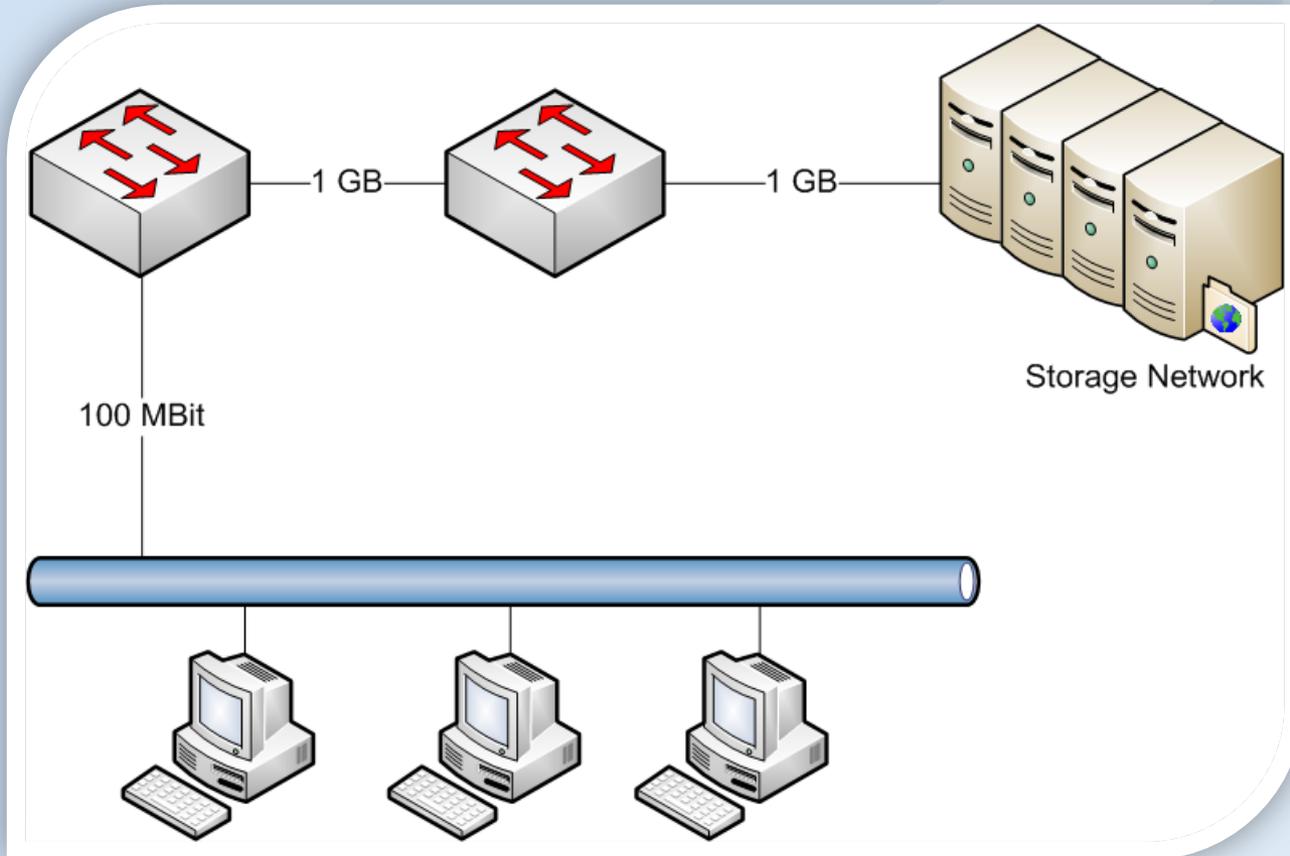
# More other „windows“ ...

- Loss Window (LW)
  - The loss window is the size of the congestion window after a TCP sender detects loss using its retransmission timer
- Restart Window (RW)
  - The restart window is the size of the congestion window after a TCP restarts transmission after an idle period (if the slow start algorithm is used)
- Flight Size
  - The amount of data that has been sent but not yet acknowledged aka “bytes in flight”

# How to react to packet loss?

- Mostly depends on the stack in use
- All stacks should reduce send rate on short notice
- Large receive buffers result in optimized retransmission of packets
- Fast Retransmission and SACK are standard nowadays → „true“ RTO triggers are rare

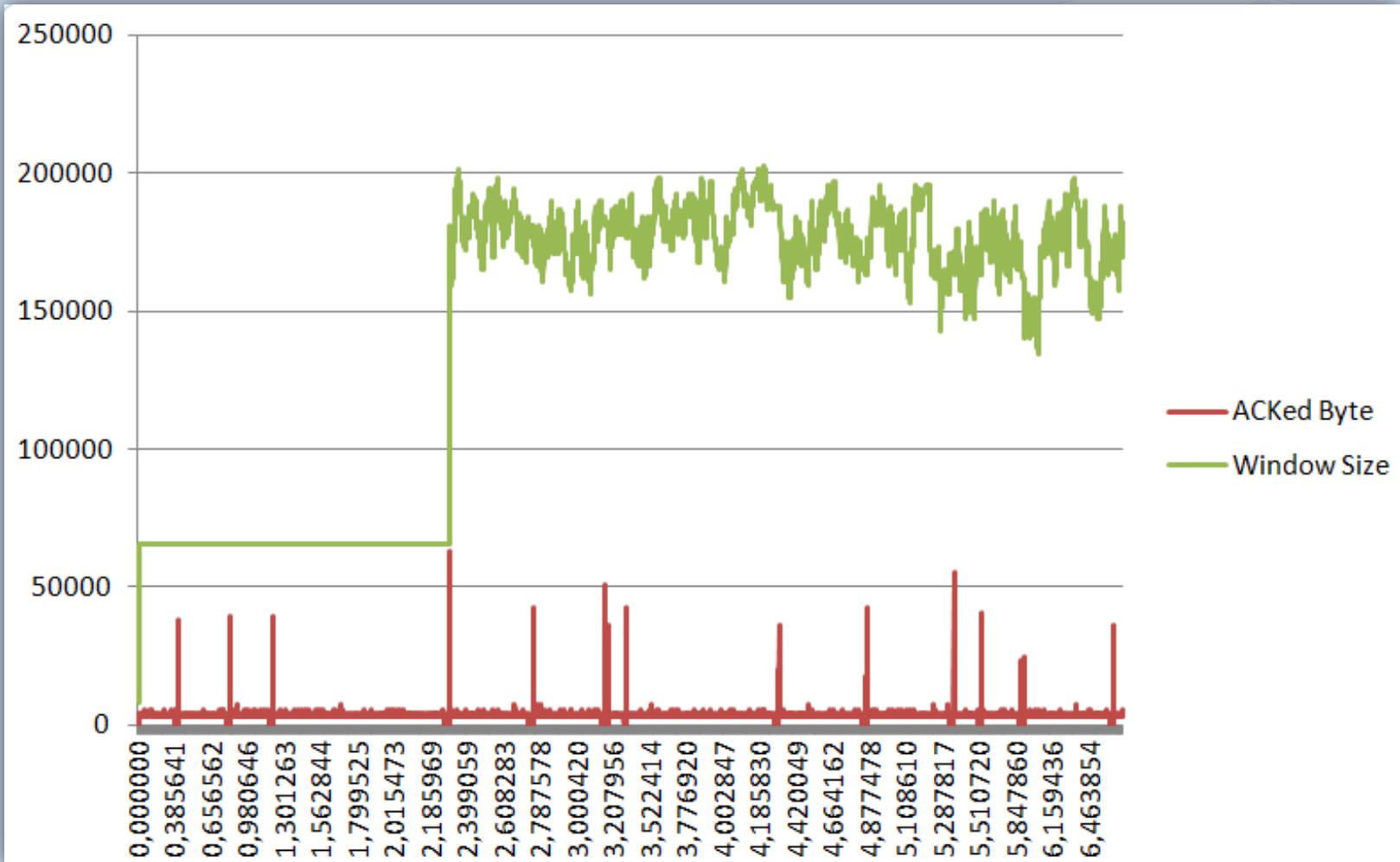
# The Lab setup...



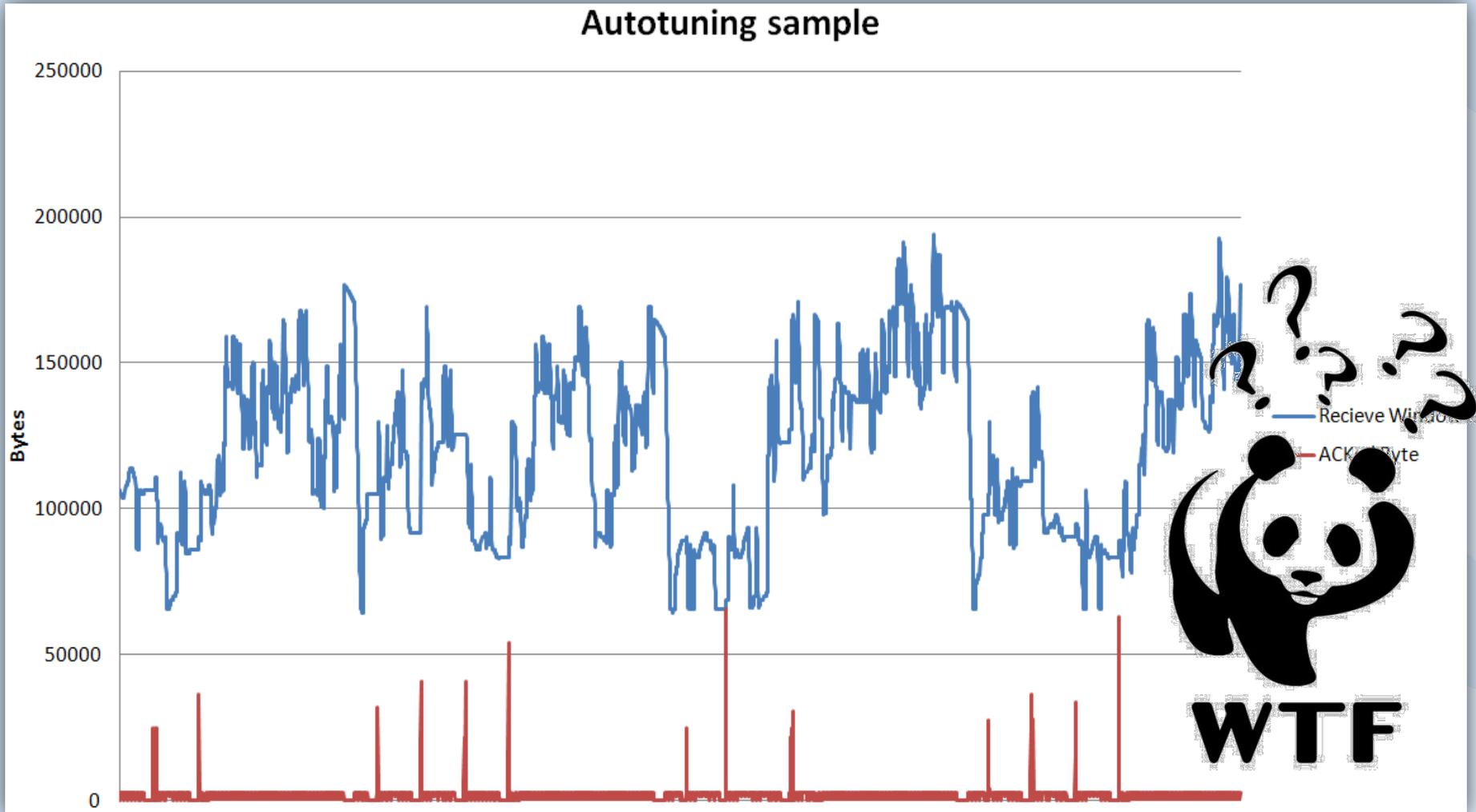
# TCP Receive Window Autotuning

- In use since implementing the NG-Stack of Microsoft Vista
- Stack and/or application can tune the TCP receive window for the duration of the tcp connection
- Was created for adjustments of rwnd on LFPs

# How does Autotuning behave?



# How does Autotuning behave?



# Autotuning Level

- Can be configured using the `netsh` command
- Available settings:
  - Disabled
  - Highlyrestricted
  - Restricted
  - Normal
  - Experimental

# Adjusting rwnd

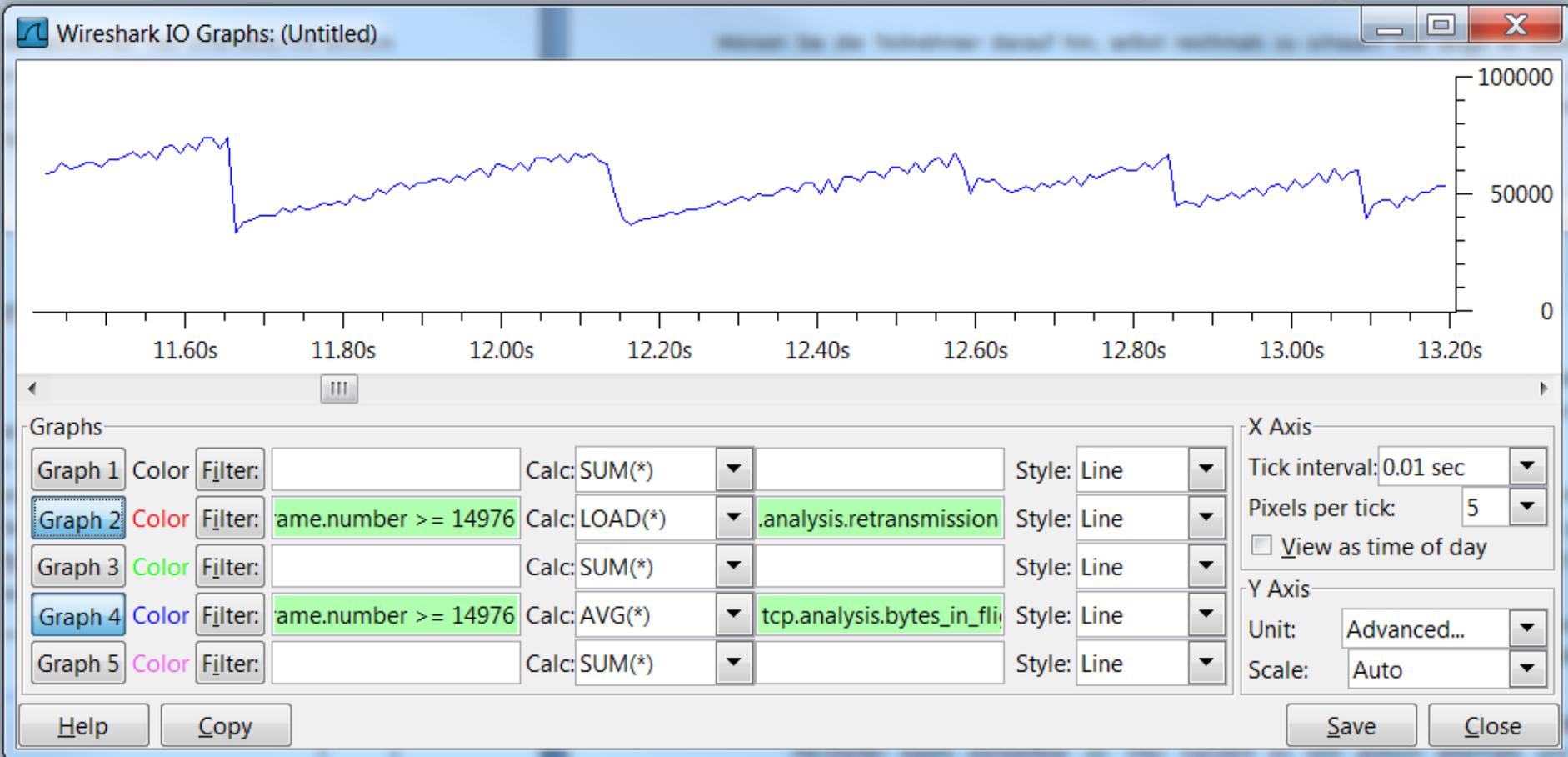
- Not available anymore: static configuration via Windows Registry settings
- Adjusting of the AutoTuning Level is possible:

<b>Disabled</b>	<b>65.536 bytes</b>
(Highly) Restricted	max. 262.144 bytes
Normal	max. 16.777.216 bytes (!)
Experimental	max. 1.073.741.824 bytes

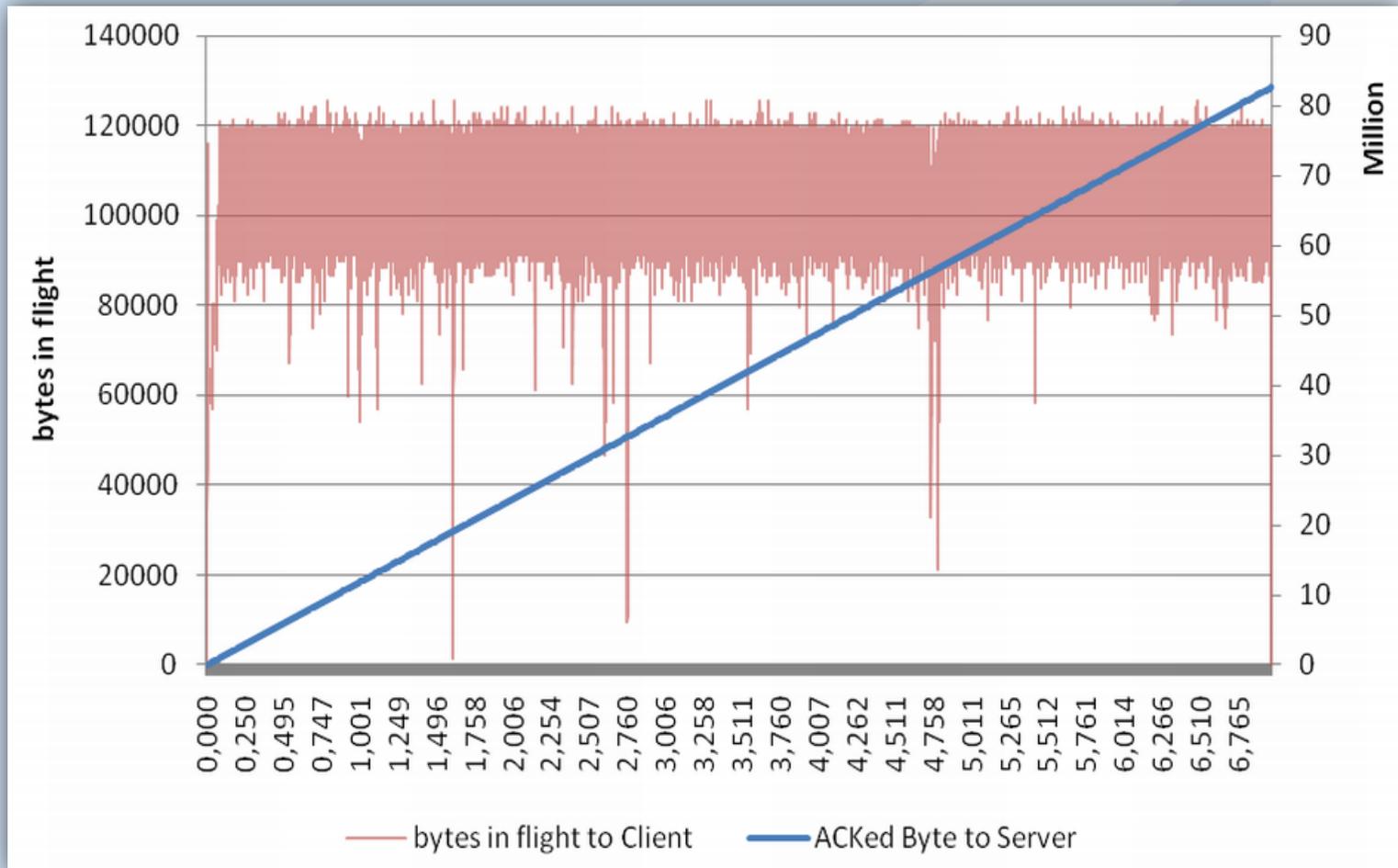
# Window Scaling Heuristics

- Settings of the TCP stack that calculates the Autotuning Level for different network profiles
- If the scaling heuristics are in use, the Autotuning Level provides a base for calculating the Receive Window and the scale factor
- Application can use this mechanism to take control of TCP stack behavior
- The use of Scaling Heuristics blocks deactivation of Receive Window AutoTuning!

# So whats happening ?



# Problem#1 - Constantly full buffers



# Problem#2 RTO

No. .	Time	Source	Destination	Protocol	Info
70	0.101	10.96.	10.1.1	TCP	49253 > 445 [PSH, ACK] Seq=1585769119 Ack=2092675820 Win=16310 Len=76
71	0.101	10.1.1	10.96.	TCP	445 > 49253 [PSH, ACK] Seq=2092675924 Ack=1585769195 Win=16384 Len=104
72	0.102	10.96.	10.1.1	TCP	49253 > 445 [PSH, ACK] Seq=1585769195 Ack=2092675924 Win=16284 Len=76
73	0.102	10.1.1	10.96.	TCP	445 > 49253 [PSH, ACK] Seq=2092676028 Ack=1585769271 Win=16384 Len=104
74	0.103	10.96.	10.1.1	TCP	[TCP Dup ACK 72#1] 49253 > 445 [ACK] Seq=1585769271 Ack=2092675924 Win=
75	1.138	10.1.1	10.96.	TCP	[TCP Retransmission] 445 > 49253 [PSH, ACK] Seq=2092675924 Ack=1585769271
76	1.138	10.96.	10.1.1	TCP	49253 > 445 [ACK] Seq=1585769271 Ack=2092676132 Win=16232 Len=0 SLE=20

# Problem#3 Delayed ACK

Source	Destination	Size	Info	delta disj
10.17.1.1	10.1.1.1	86	[TCP Dup ACK 6#5] 60421 > 445 [ACK] seq=1461501365 Ack=4066767320 win=16605 Len=0	0.000
10.1.1.1	10.17.1.1	1518	[TCP Fast Retransmission] NBSS Continuation Message	0.000
10.1.1.1	10.17.1.1	1518	[TCP out-of-order] NBSS Continuation Message	0.000
10.1.1.1	10.17.1.1	1518	[TCP out-of-order] NBSS Continuation Message	0.000
10.17.1.1	10.1.1.1	86	60421 > 445 [ACK] seq=1461501365 Ack=4066770240 win=16425 Len=0 SLE=4066808912	0.000
10.1.1.1	10.17.1.1	1518	[TCP out-of-order] NBSS Continuation Message	0.000
10.17.1.1	10.1.1.1	86	60421 > 445 [ACK] seq=1461501365 Ack=4066773160 win=16425 Len=0 SLE=4066808912	0.000
10.1.1.1	10.17.1.1	1518	[TCP Retransmission] NBSS Continuation Message	0.000
10.17.1.1	10.1.1.1	86	60421 > 445 [ACK] seq=1461501365 Ack=4066774620 win=16425 Len=0 SLE=4066808912	0.196
10.1.1.1	10.17.1.1	1518	[TCP Retransmission] NBSS Continuation Message	0.000
10.17.1.1	10.1.1.1	86	60421 > 445 [ACK] seq=1461501365 Ack=4066776080 win=16425 Len=0 SLE=4066808912	0.202
10.1.1.1	10.17.1.1	1518	[TCP Retransmission] NBSS Continuation Message	0.000
10.17.1.1	10.1.1.1	86	60421 > 445 [ACK] seq=1461501365 Ack=4066777540 win=16425 Len=0 SLE=4066808912	0.202
10.1.1.1	10.17.1.1	1518	[TCP Retransmission] NBSS Continuation Message	0.000
10.17.1.1	10.1.1.1	86	60421 > 445 [ACK] seq=1461501365 Ack=4066779000 win=16425 Len=0 SLE=4066808912	0.202

# What I learned from TCP RFCs

- TCP stacks **are** different
- There **MAY** be issues where:
  - RECOMMENDED reading is **REQUIRED**
  - you **MUST** read some RFCs and
  - **SHALL NOT** give up too early 😊

Thank you !

