SHARKFEST'14

WIRESHARK DEVELOPER AND USER CONFERENCE

JUNE 16-20 2014 · DOMINICAN UNIVERSITY

B5 – TCP Analysis - First Steps

Jasper Bongertz, Senior Consultant

Airbus Defence and Space

About this presentation file

Since this presentation contains lot of animated slides I decided against converting it to a static PDF and offer that for the Sharkfest retrospective page. Instead, you get the PPT, so you can watch stuff happen in presentation mode.

Use the slides in your own trainings if you like. If you do, don‘t forget to mention where you got them from – it was a lot of work creating these ☺

Cheers,

Jasper

Agenda

- Basics of managing Data Transfers
- The Sliding Window
- Packet Loss - „when Things go wrong“

Basics of managing Data Transfers

Basics of managing data transfers

1. Application data is segmented in small chunks before transfer

Basics of managing data transfers

2. Sucessful Transfer of segmented data is not guaranteed by the physical network...

Basics of managing data transfers

3. ...so let‘s number those segments and confirm which ones are received...

Basics of managing data transfers

- **Question:** is numbering the packets good enough to ensure everything is fine?
- **Answer:** unfortunately not...
  - ...because we only know a certain segment was received, but not if it was <u>complete</u>

Basics of managing data transfers

How about this?

#1
Len 100

Got #1
Len 100

Can we improve that further?

Basics of managing data transfers

So let‘s see how TCP does it:

Start at
100
Len 100

Got 200

The confirmation is the number of <u>continuous</u> bytes received (meaning: no gaps)

TCP Sequence and Acknowledge

- The „Start at“ number is called „Sequence Number“
- The „Got it“ number is called „Acknowledgement“
- This is how it looks like in Wireshark:

```
⊞ Internet Protocol Version 4, Src: 192.168.124.100 (192.168.124.100), Dst: 81.209.179.69 (81.209.179.69)
⊟ Transmission Control Protocol, Src Port: 4016 (4016), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 351
    Source port: 4016 (4016)
    Destination port: 80 (80)
    [Stream index: 0]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 352    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header length: 20 bytes
```

```
⊞ Internet Protocol Version 4, Src: 81.209.179.69 (81.209.179.69), Dst: 192.168.124.100 (192.168.124.100)
⊟ Transmission Control Protocol, Src Port: 80 (80), Dst Port: 4016 (4016), Seq: 1, Ack: 352, Len: 293
    Source port: 80 (80)
    Destination port: 4016 (4016)
    [Stream index: 0]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 294    (relative sequence number)]
    Acknowledgment number: 352    (relative ack number)
    Header length: 20 bytes
```

TCP Session Start

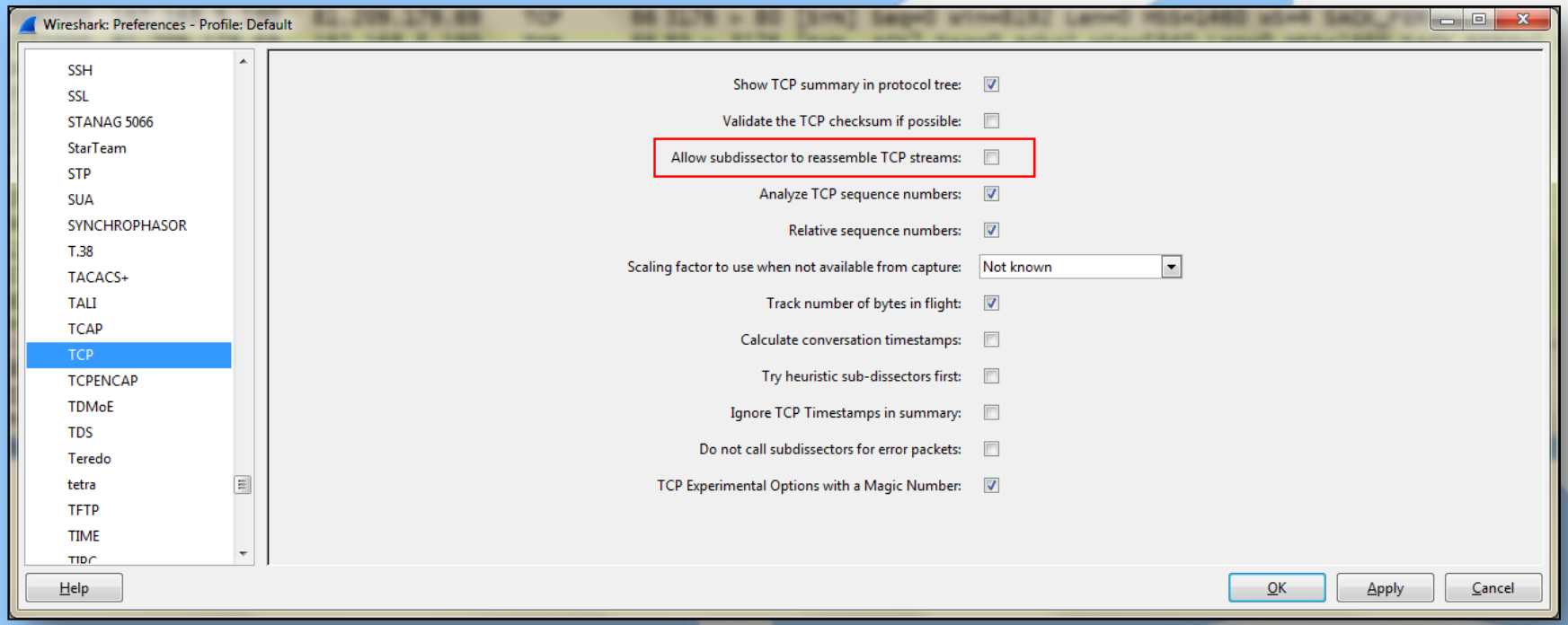
- Before exchanging data, TCP needs to establish the session
  - „Three Way Handshake“: SYN -> SYN/ACK -> ACK

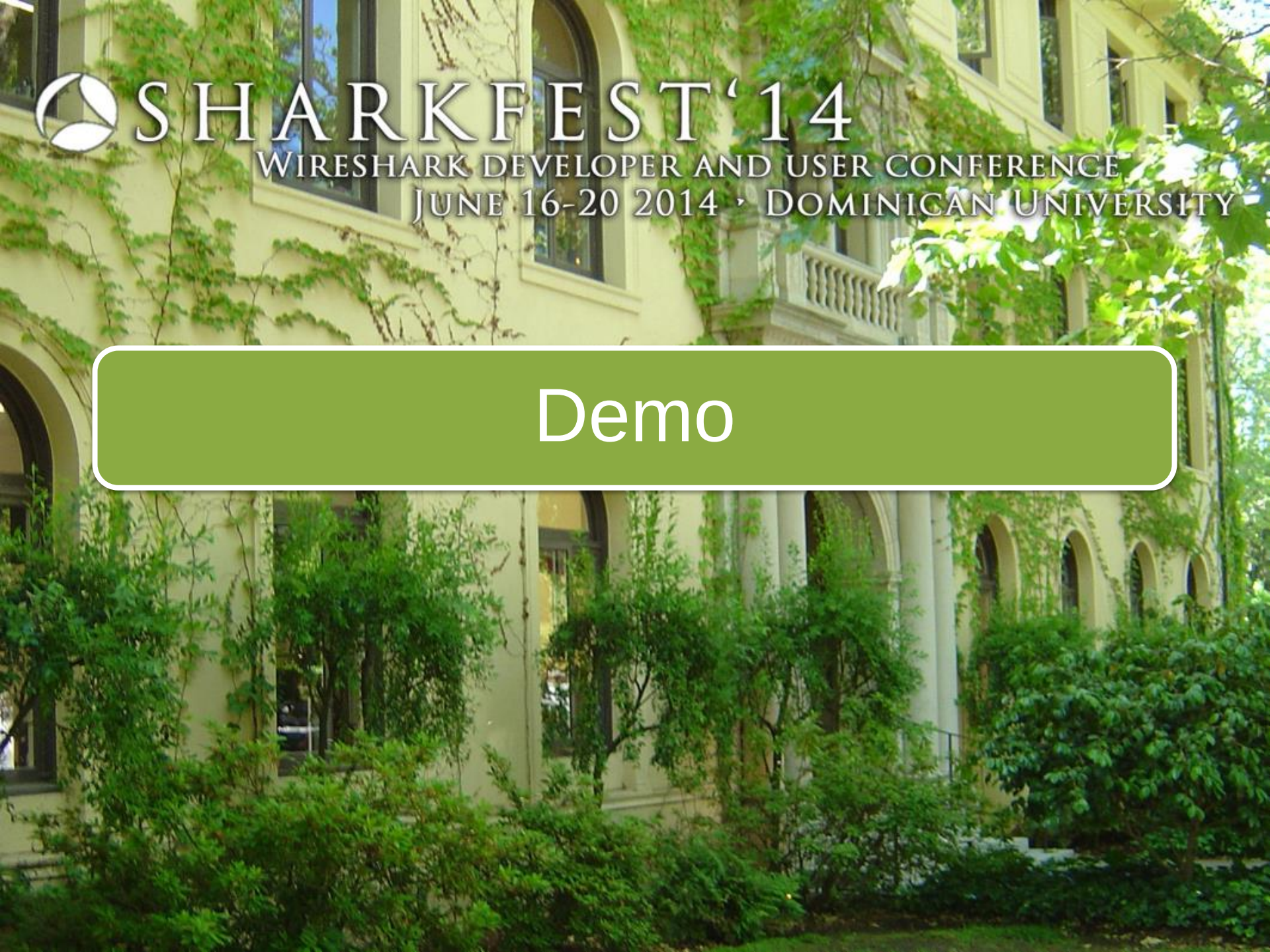
```
Info
3176 > 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
80 > 3176 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=64
3176 > 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0
GET / HTTP/1.1
```

  - ** **Special rule:** SYN flags count as 1 byte! **

It‘s Wireshark time

- First, let‘s turn of TCP reassembly:

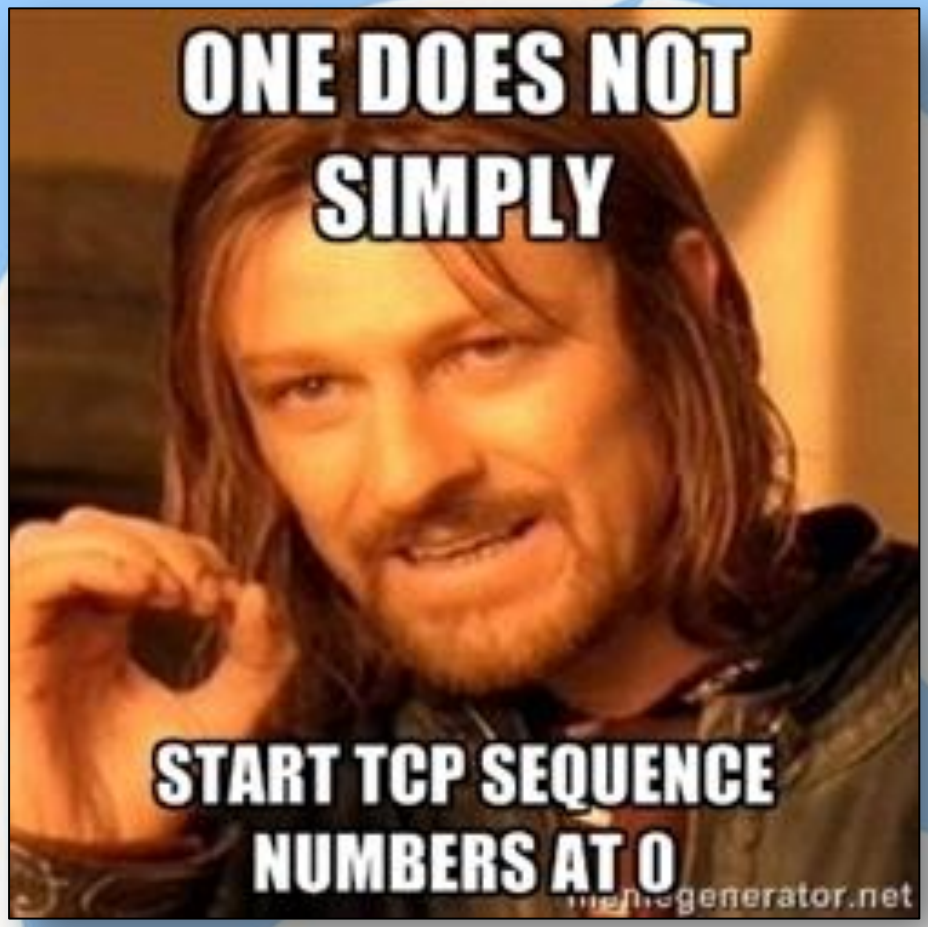



SHARKFEST'14
WIRESHARK DEVELOPER AND USER CONFERENCE
JUNE 16-20 2014 · DOMINICAN UNIVERSITY

Demo

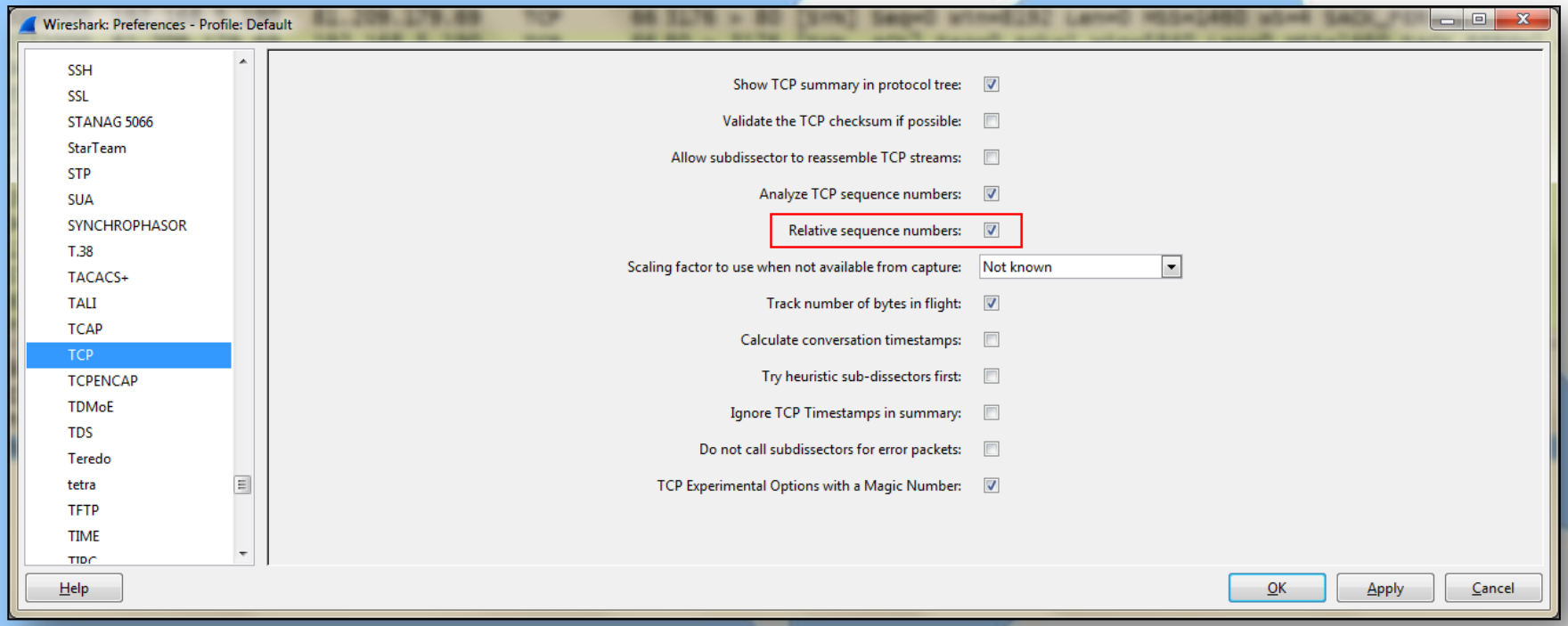
Initial Sequence Numbers

- Wireshark displays „Relative Sequence Numbers“ by default
- In reality, the initial sequence number is **random**
- It can be anything between 0 and 2^{32}:
  - 0 - 4294967296



Relative Sequence Numbers

- You can turn them on and off in the TCP preferences:



Sequence Numbers – The Rules

1. Each TCP sequence starts with random number
2. It is increased by 1 for each byte transmitted
3. SYN and FIN flags count as 1 Byte („Phantom Byte“)

Okay, let's see…

SYN, SeqNo = 100,000 →

← SYN, SeqNo = 500,000
ACK, AckNo = 100,001

ACK, SeqNo = 100,001
AckNo = 500,001 →

80 Byte Data
SeqNo = 100,001 →

← ACK, SeqNo = 500,001
AckNo = 100,081

Additional things to consider

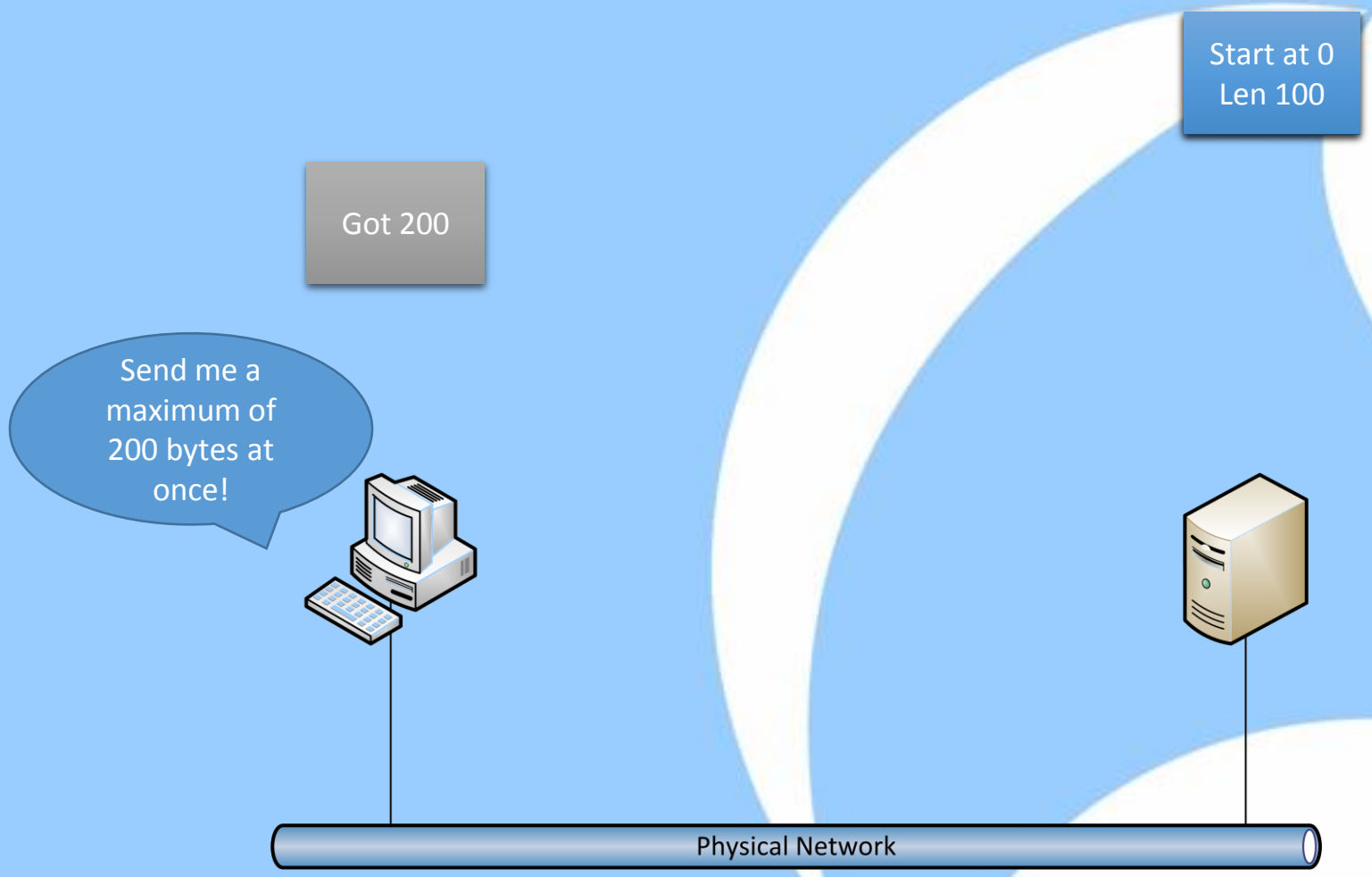
- Every direction tracks its own sequence number
- Relative sequence numbers can fool you because they may look similar for both directions

The Sliding Window

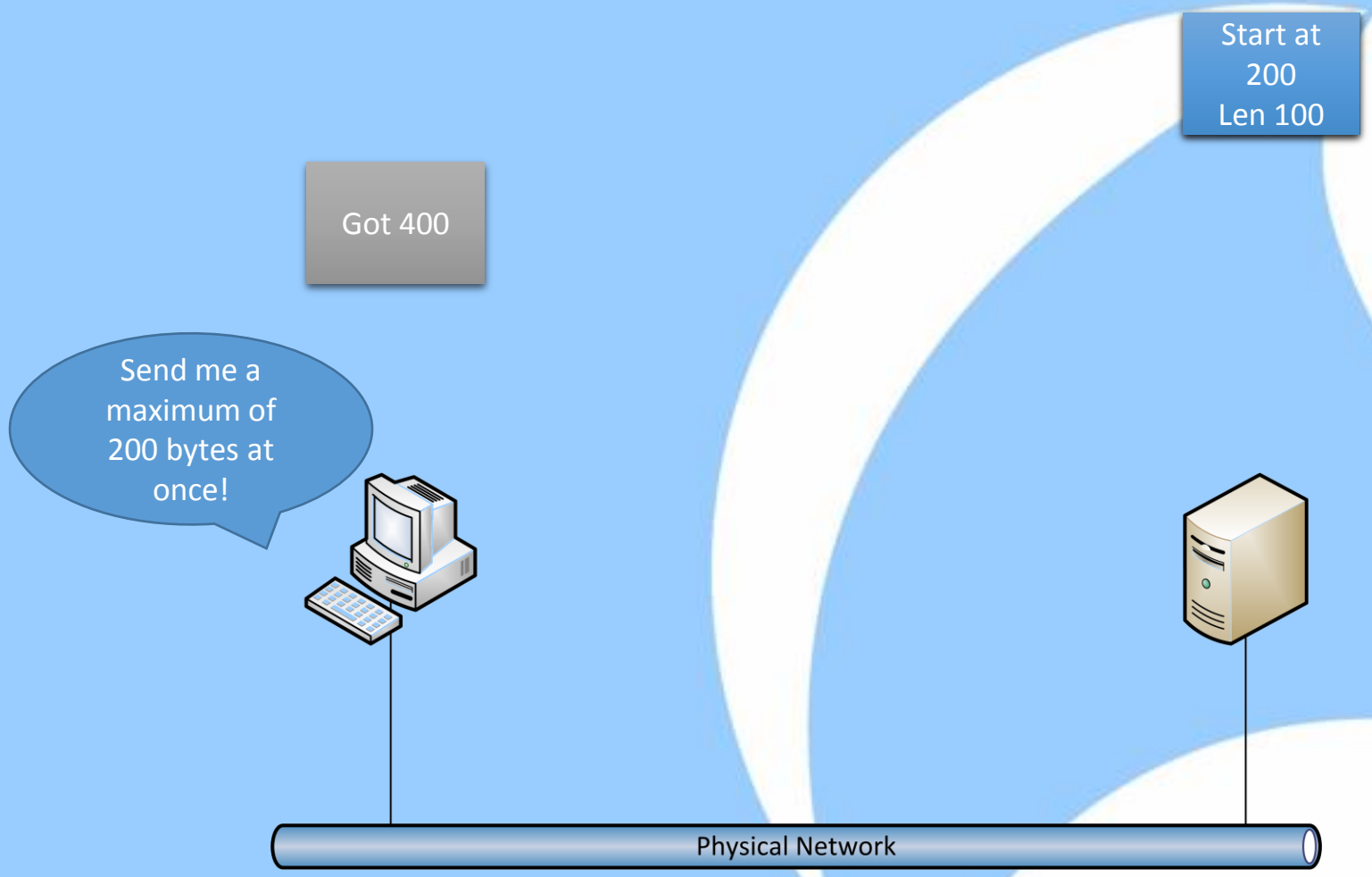
Positive Ack with Retransmission

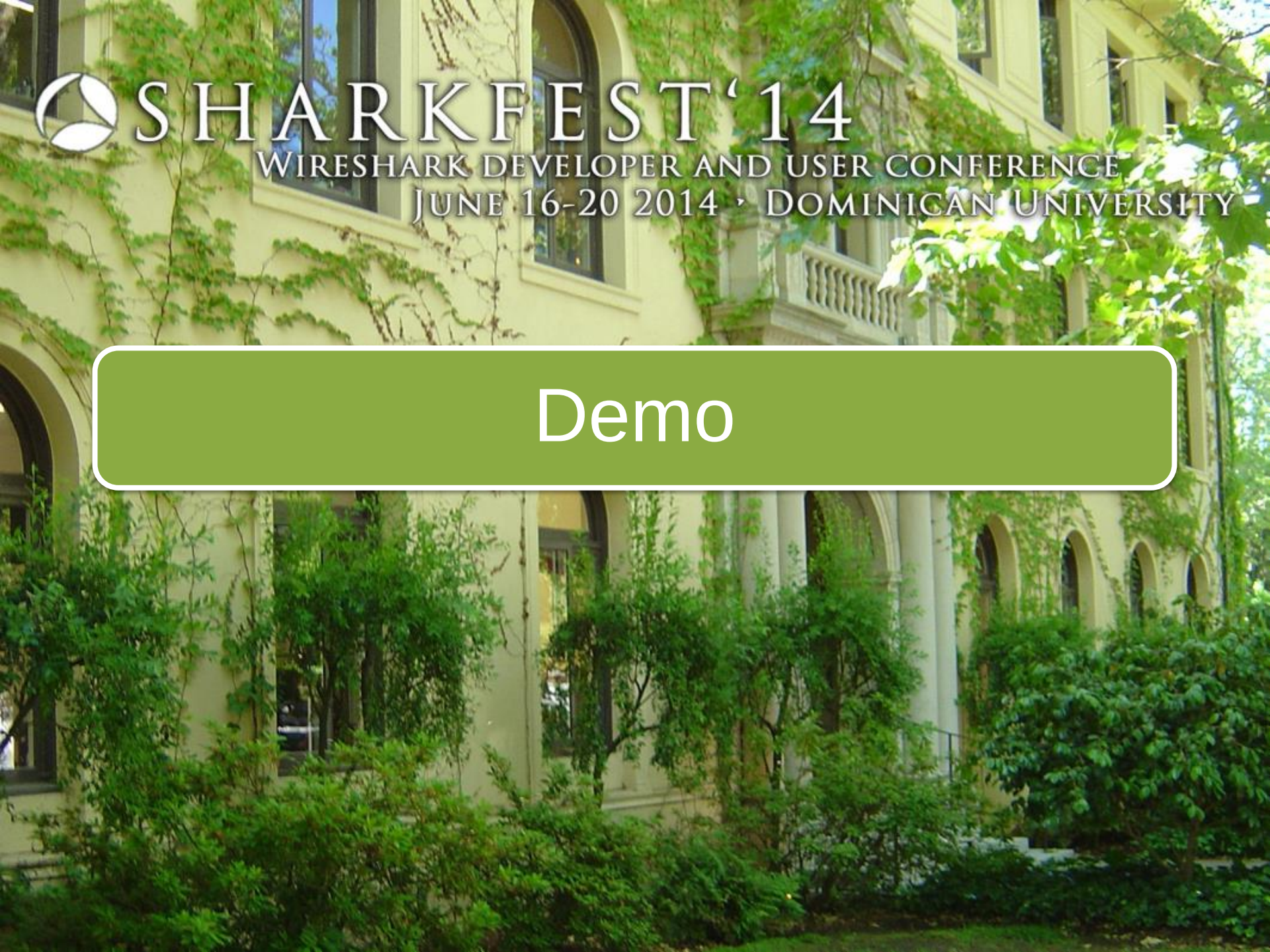
- It is not very efficient to send single packets back and forth:

Instead, send more…



Instead, send more…




SHARKFEST'14
WIRESHARK DEVELOPER AND USER CONFERENCE
JUNE 16-20 2014 · DOMINICAN UNIVERSITY

Demo

TCP Window Size

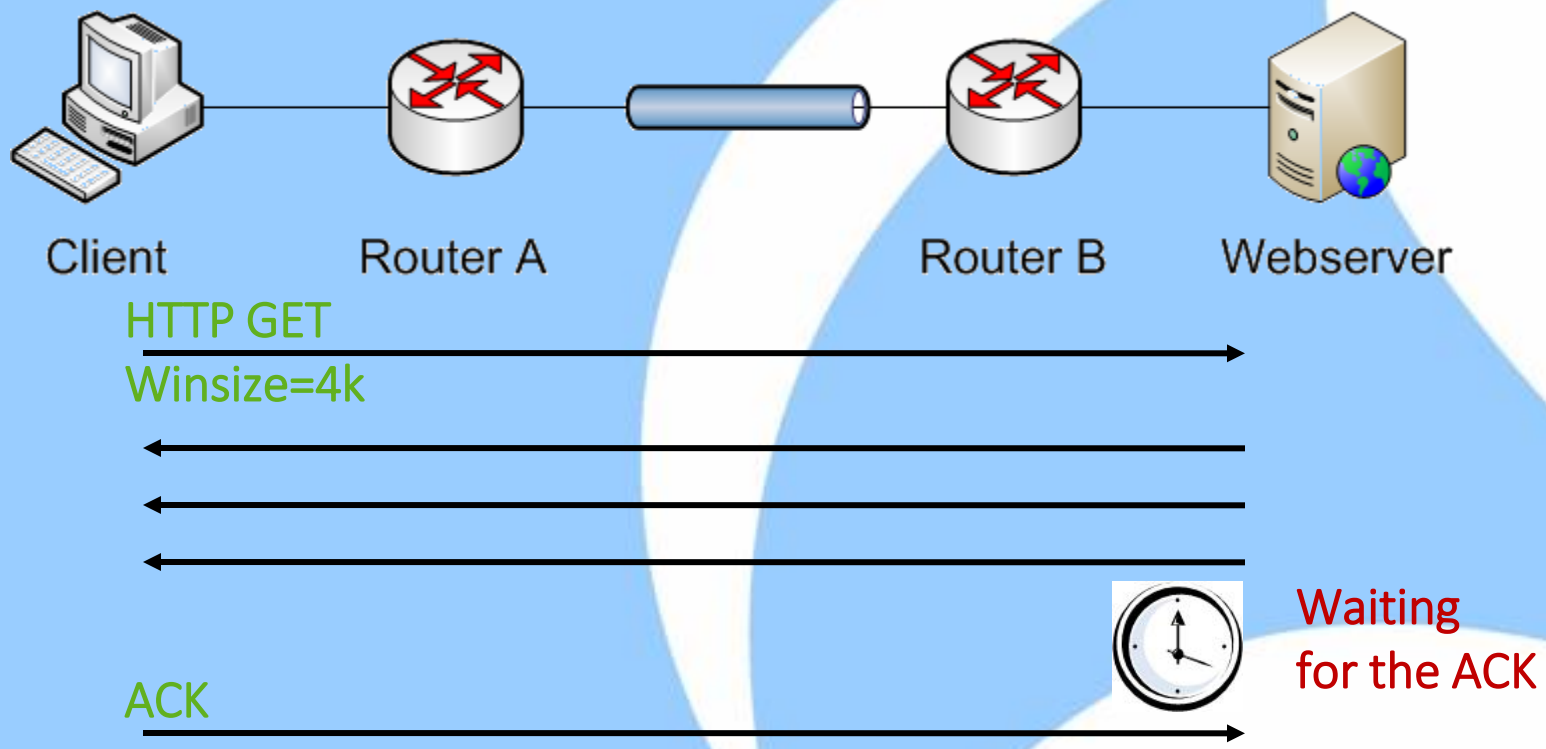
- The TCP window size is very important as a speed and congestion factor

```
⊟ Transmission Control Protocol, Src Port: 80 (80), Dst Port: 3176 (3176), Seq: 1, Ack: 372, Len: 0
    Source port: 80 (80)
    Destination port: 3176 (3176)
    [Stream index: 0]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 372    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 108
    [Calculated window size: 6912]
    [Window size scaling factor: 64]
  ⊞ Checksum: 0xeaf2 [validation disabled]
  ⊞ [SEQ/ACK analysis]
```

- Look for „Calculated Window Size“ to read the current value

Insufficient TCP window size

- A small window can slow down the transmission:

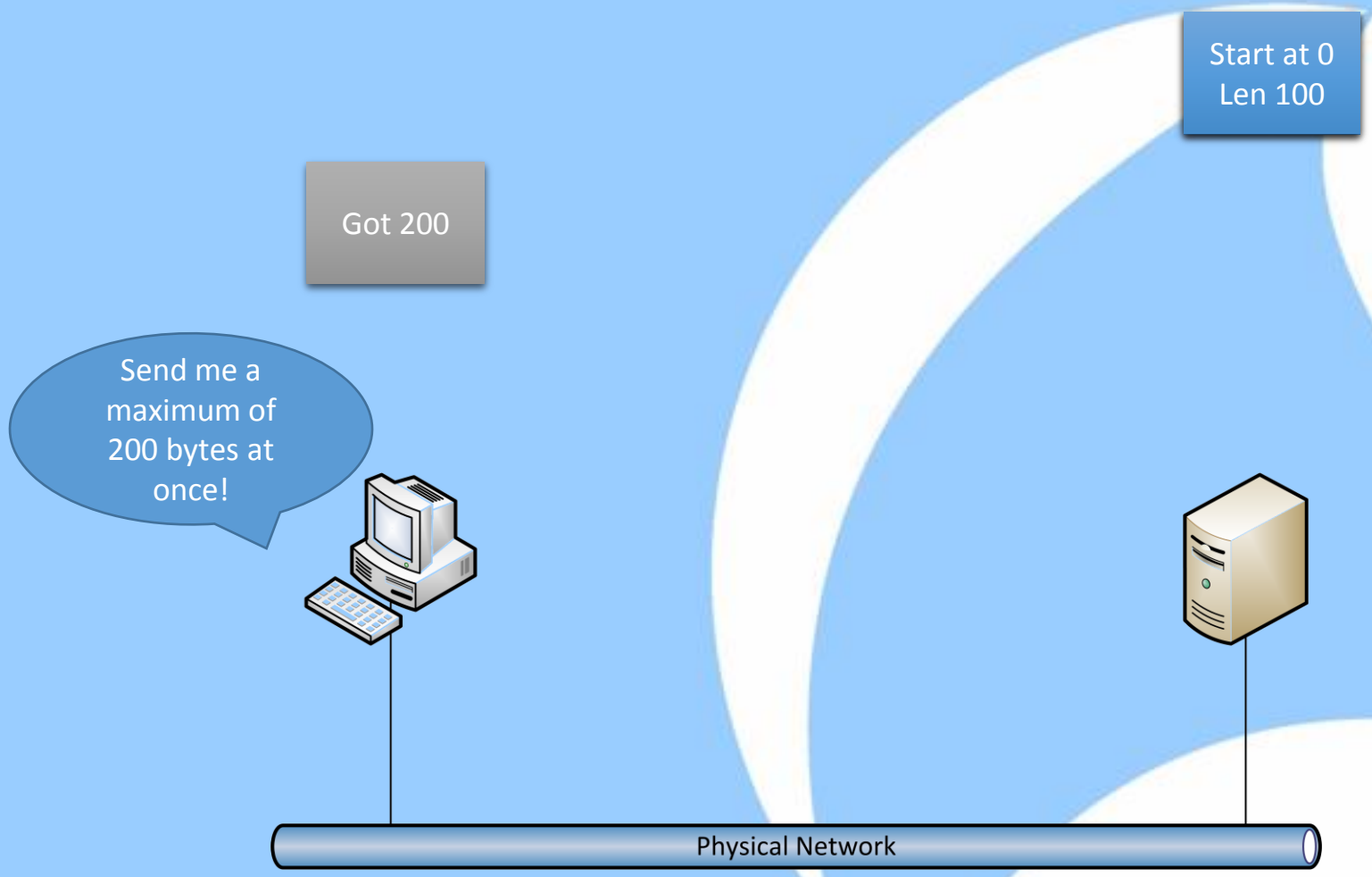


Performance Problems

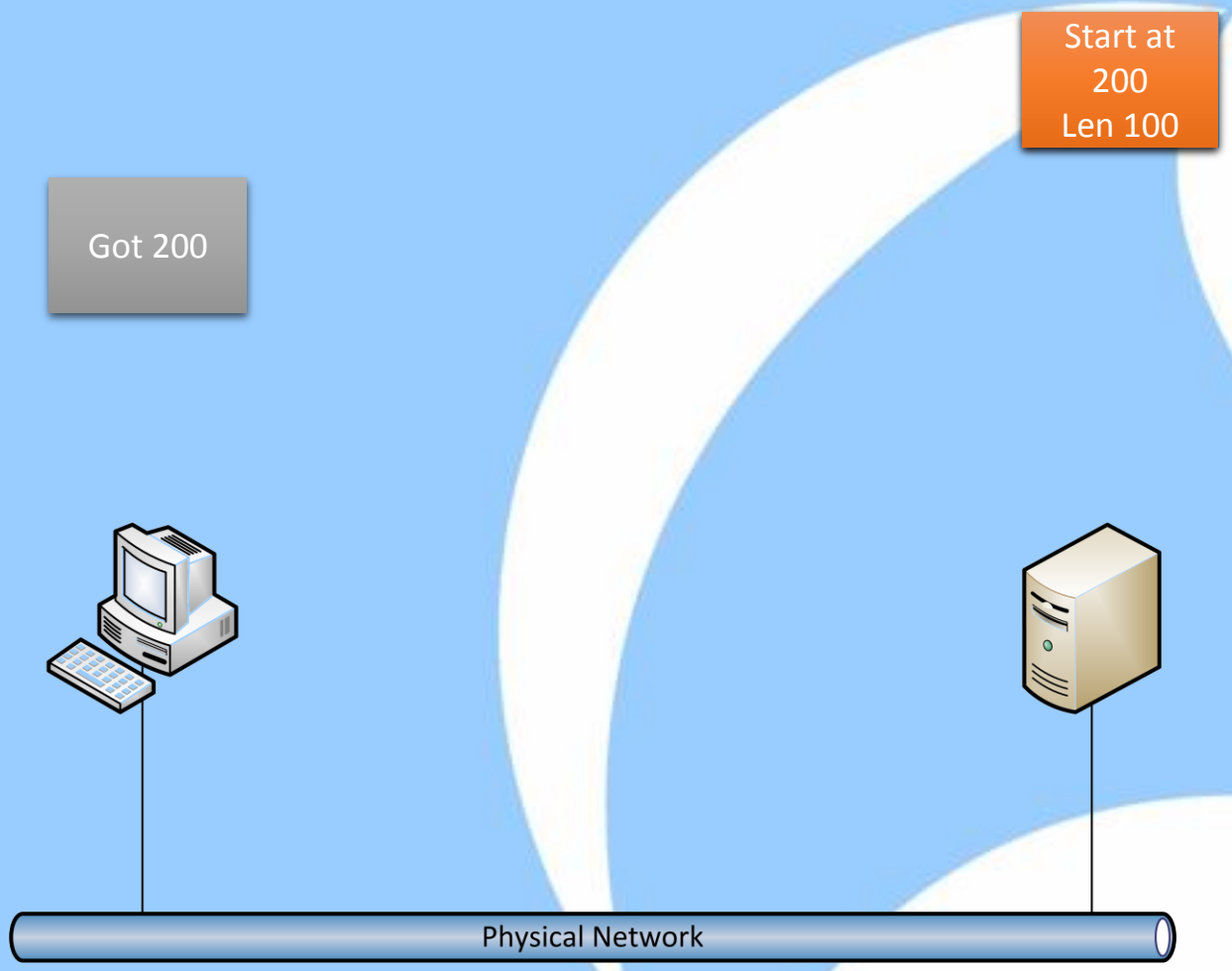
- The TCP Window is a great help for locating congested servers and clients
- If a computer sends very low window sizes, or window sizes of zero, it may be in trouble
  - Hardware apparently not fast enough to cope with **incoming** packets
- Exceptions:
  - Reset Packets -> always has window size of zero
  - Busy servers that do not receive much, e.g. Newstickers often have low window sizes

Packet Loss – When things go wrong

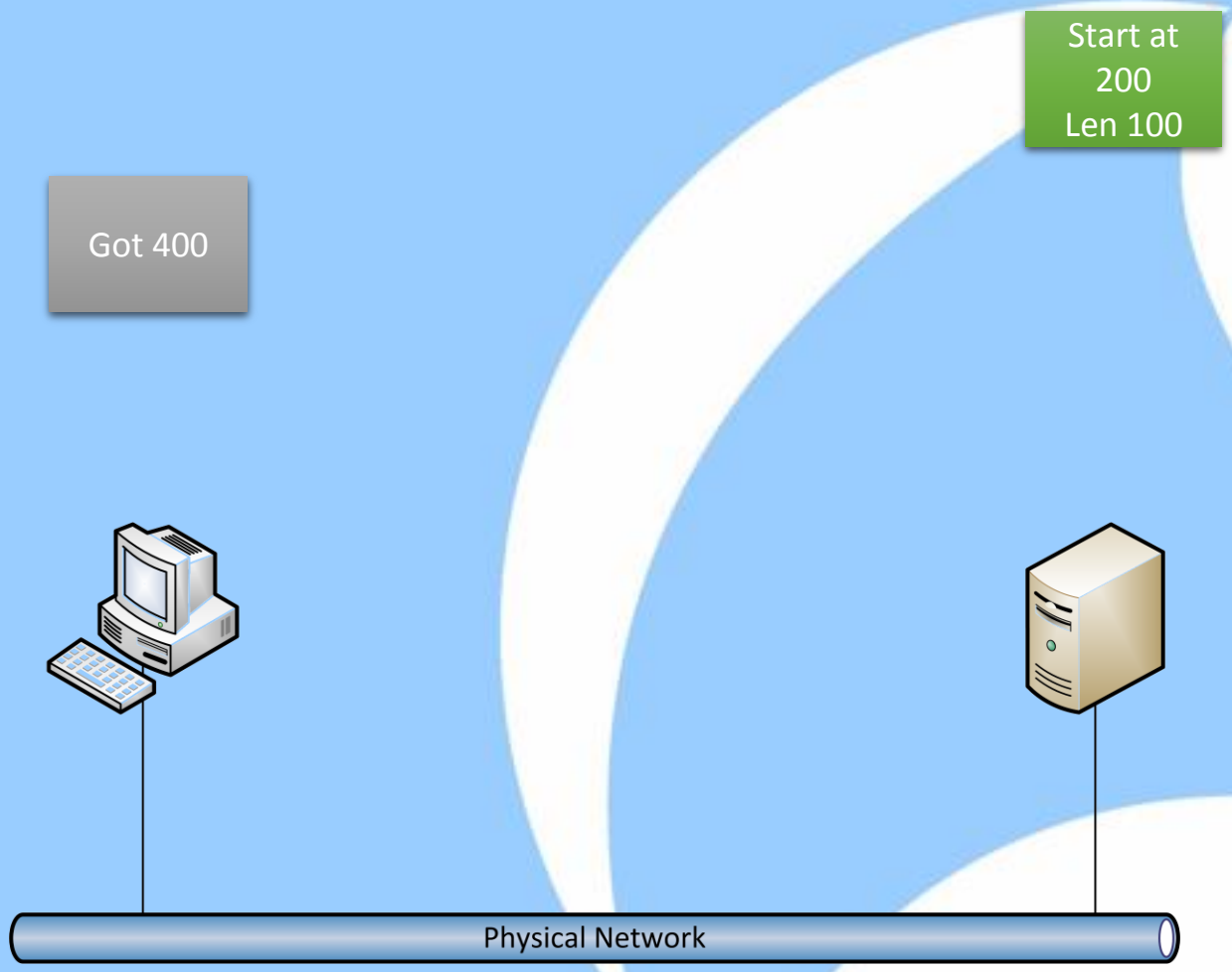
First, everything looks good

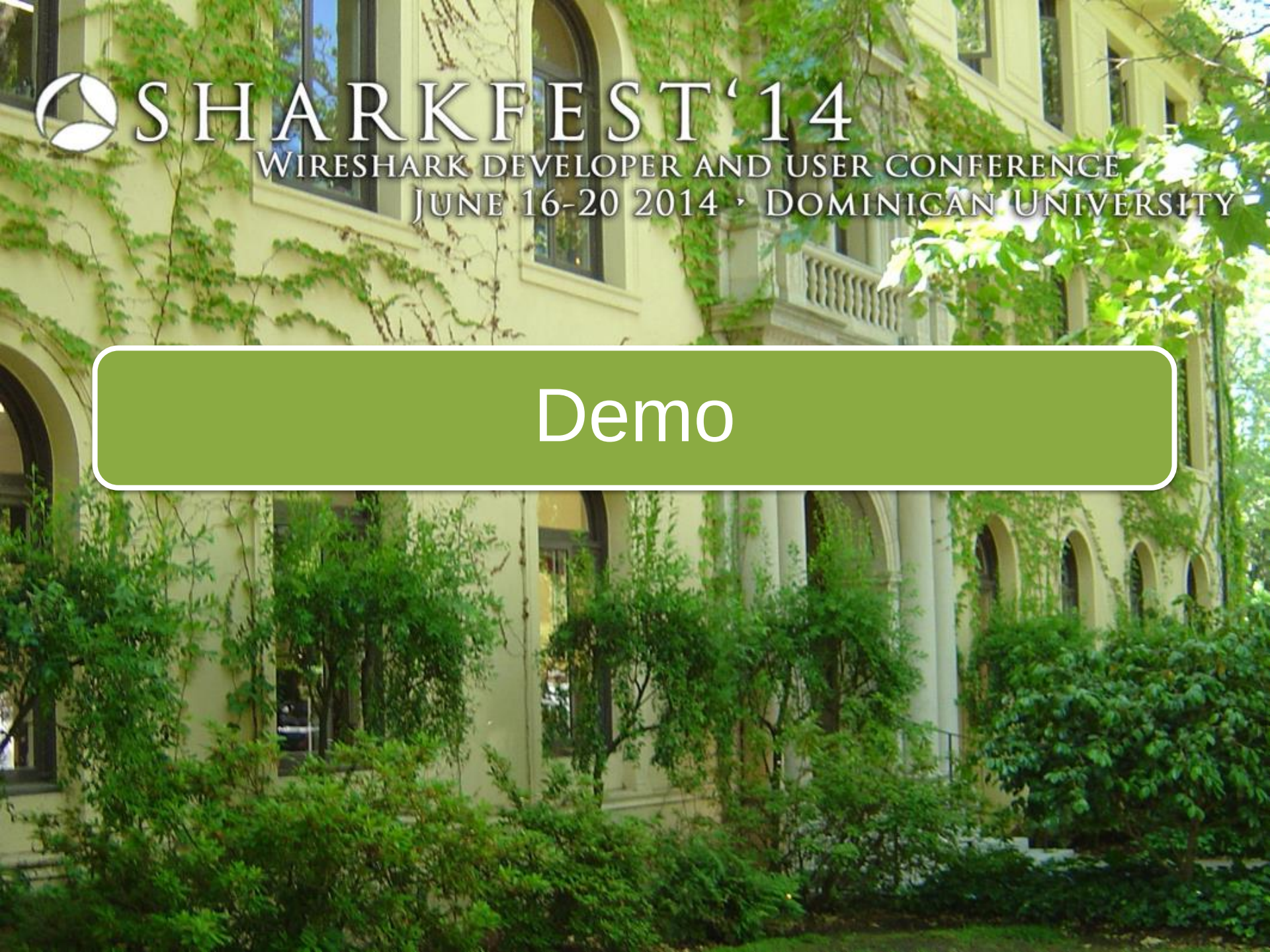


Then something goes wrong…



Retransmission




SHARKFEST'14
WIRESHARK DEVELOPER AND USER CONFERENCE
JUNE 16-20 2014 · DOMINICAN UNIVERSITY

Demo

TCP Retransmissions

- Retransmissions happen in every network
- Different ways to trigger a retransmission:
  - By time out
  - By Triple Duplicate ACK
  - By Selective Acknowledgement (SACK)
- Most important aspect:
  - How much time do they cost?

Thanks! Questions?

eMail: jasper@packet-foo.com
Blog: blog.packet-foo.com
Twitter: @packetjay