Chasing application performance with Wireshark

Analyzing Database Applications with Wireshark

Matthias Kaiser
ExperTeach GmbH,
Germany
Hello!

I am **Matthias Kaiser**

I am here because I love packet analysis with Wireshark and I love to present

You can find me at Twitter: [@wiresharky](https://twitter.com/wiresharky)

You can contact me via Email: [matthias.kaiser@experteach.de](mailto:matthias.kaiser@experteach.de)
About Myself

- Sniffer University Instructor at Network General /NAI
- Freelancer with own analysis courses
- Trainer and Consultant at ExperTeach
  - Wireshark Training and more
  - Consulting Services for Packet Analysis
- Motto:
  - „Every trace hides a story. Uncover and tell it.“
Files and Downloads

- Presentation covers Real-Life Cases
- Trace Files and Wireshark Profiles:
  - https://tinyurl.com/8nmc59c2
- Trace Files have been anonymized and sanitized with TraceWrangler, © by Jasper Bongertz
Agenda

- Database applications
- Before we start...
- Sample Database Flow
- Case Study 1
- Case Study 3
- Application metrics for Wireshark
- Lessons learned
- Q&A
Database Applications

- Applications drive processes .... Everywhere
- Database applications are all over the place.
  - E-Commerce
  - ERP, like Warehousing or Finance and HR
  - Automation
  - ..... 
- All applications will be IP-based
- Software Defined Networking
  - Controllers will tell servers and network, what to do.
- So, we better understand, how applications work ... in order to analyze them with Wireshark
Before we start ... looking at packets

Have a plan

- Set your goals for the analysis.
- Describe your problem.
- Find out who is affected?
  - Locations, Users, entire PCs, just applications
  - Check the severity of your problem
- Identify the application(s)
- Find out when the problem occurs
  - Permanent
  - Sporadic / intermittent
- Do not just capture some traffic!
- Do not just look at trace files!
- And please ... stop guessing!
Before we start... II

Capture
- What are the traffic flows for your application?
- Capture Location: Where do I see interesting traffic?
- Define the user activity to be analysed.
  - Permanent problem: Pick one typical action
  - Intermittent problem: Long-Term analysis

Analyze
- Prepare your Wireshark (Profiles)
- Filter your trace file
  - IP addresses, Ports
- Identify traffic for User actions
- Know the key metrics for the application?

And then
- Do the analysis
Sample Database Flow

- **Front end Process**
  - HTTP(S) or specific TCP

- **Back end Process**
  - Many Requests - Responses (Application Turns)
  - Small amount of data

- **Back End sensitive to**
  - Round Trip Time
  - Number of Turns
  - Application Response Time of Database Server
  - Delays at App-Server
Traffic Flows and Client Server Architecture

- Client-Server architecture
  - Fat Clients
  - Terminal Server
  - Virtual Clients (VDI)
  - Cloud environment

- Traffic flows
  - Client - Servers - DB-Srv

- Which Users are affected?

- Capture location
  - Client Session
  - Application
Case Study 1
# Real-Life Case #1

- **Case:** Weighing process of palettes (steel)
- **Problem:** High transaction times for database transaction → Permanent
- **User activity:** Weighing process (repeatable)
- **Trace Files:**
  - 1-Site1-before.pcapng
  - 1-Site2-reference.pcapng
- **Wireshark Profile:** App-Analysis-I
- **Suspect:** Network
- **Questions:** Where is the problem?
Real-Life Case #1 - network map

- Network map for Case #1
- Traces taken at App-Servers.
- Front-End and Back-End Flows visible
Real-Life Case #1 - Analysis

● Front End - Site 1:
  ○ Transaction Time: 40.6 s
  ○ High ACK-Times at App-Srv: app. 150 - 200 ms
  ○ TCP Retransmissions

● Front End - Site 2:
  ○ Transaction Time: 15.2 s
  ○ High ACK-Times at App-Srv: app. 100 - 150 ms
  ○ TCP Retransmissions
### Real-Life Case #1 - Analysis

**Filter on just application data**
- `tcp.len > 1`
- Sort by Delta Time
- Large Delta times can be easily spotted.

**Sort by High Delta Times**
- **From App-SRV:**
  - 10 * High Delta Times
  - Total: 32.6 seconds
- **From Database-SRV:**
  - 2 * High SRT
  - Total: 4.55 seconds

<table>
<thead>
<tr>
<th>No.</th>
<th>Delta Time (ms)</th>
<th>Rel. Time (ms)</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>0.000000</td>
<td>Client-Site1</td>
<td>App-SRV-Site-1</td>
<td>TCP</td>
<td>1934813 + 2048 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2795672</td>
<td>2795672</td>
<td>App-SRV-Site-1</td>
<td>App-SRV-Site-1</td>
<td>TCP</td>
<td>1534152 + 3231 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.017785</td>
<td>8.813457</td>
<td>DB-Server-1</td>
<td>App-SRV-Site-1</td>
<td>TCP</td>
<td>6333231 + 4152 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3.154290</td>
<td>5967747</td>
<td>App-SRV-Site-1</td>
<td>DB-Server-1</td>
<td>TCP</td>
<td>1774152 + 3231 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.015088</td>
<td>982835</td>
<td>DB-Server-1</td>
<td>App-SRV-Site-1</td>
<td>TCP</td>
<td>2213231 + 4152 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>6.952157</td>
<td>15.336928</td>
<td>App-SRV-Site-1</td>
<td>DB-Server-1</td>
<td>TCP</td>
<td>1534152 + 3231 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>169</td>
<td>3.919706</td>
<td>33.623444</td>
<td>App-SRV-Site-1</td>
<td>DB-Server-1</td>
<td>TCP</td>
<td>3864152 + 3231 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>3.534441</td>
<td>18.886262</td>
<td>App-SRV-Site-1</td>
<td>DB-Server-1</td>
<td>TCP</td>
<td>1774152 + 3231 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>3.370261</td>
<td>23.765958</td>
<td>App-SRV-Site-1</td>
<td>DB-Server-1</td>
<td>TCP</td>
<td>1534153 + 3231 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3.154290</td>
<td>5.967747</td>
<td>App-SRV-Site-1</td>
<td>DB-Server-1</td>
<td>TCP</td>
<td>1774152 + 3231 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>3.145083</td>
<td>26.919965</td>
<td>App-SRV-Site-1</td>
<td>DB-Server-1</td>
<td>TCP</td>
<td>1744153 + 3231 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>184</td>
<td>2.898743</td>
<td>40.180704</td>
<td>App-SRV-Site-1</td>
<td>DB-Server-1</td>
<td>TCP</td>
<td>1744153 + 3231 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2.795672</td>
<td>2.795672</td>
<td>App-SRV-Site-1</td>
<td>DB-Server-1</td>
<td>TCP</td>
<td>1534152 + 3231 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>163</td>
<td>2.422311</td>
<td>29.687232</td>
<td>DB-Server-1</td>
<td>App-SRV-Site-1</td>
<td>TCP</td>
<td>15143231 + 4152 [ACK] Seq=</td>
<td></td>
</tr>
<tr>
<td>171</td>
<td>2.132969</td>
<td>35.765413</td>
<td>DB-Server-1</td>
<td>App-SRV-Site-1</td>
<td>TCP</td>
<td>15143231 + 4152 [ACK] Seq=</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>1.437712</td>
<td>8.384771</td>
<td>Client-Site1</td>
<td>App-SRV-Site-1</td>
<td>TCP</td>
<td>2214813 + 2048 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>1.388123</td>
<td>20.325455</td>
<td>App-SRV-Site-1</td>
<td>DB-Server-1</td>
<td>TCP</td>
<td>4734152 + 3231 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>173</td>
<td>1.255924</td>
<td>37.012337</td>
<td>App-SRV-Site-1</td>
<td>DB-Server-1</td>
<td>TCP</td>
<td>1534153 + 3231 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>0.441056</td>
<td>6.947059</td>
<td>App-SRV-Site-1</td>
<td>DB-Server-1</td>
<td>TCP</td>
<td>1162048 + 4813 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>0.232304</td>
<td>6.419953</td>
<td>DB-Server-1</td>
<td>App-SRV-Site-1</td>
<td>TCP</td>
<td>2213231 + 4152 [PSH, ACK]</td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>0.224049</td>
<td>37.252566</td>
<td>DB-Server-1</td>
<td>App-SRV-Site-1</td>
<td>TCP</td>
<td>15143231 + 4152 [ACK] Seq=</td>
<td></td>
</tr>
</tbody>
</table>
Real-Life Case #1 - Analysis

- Comparing Site 1 to Site 2
Real-Life Case #1 - Solution

● Results from the analysis
  ○ App-Server seems to take many long breaks!
  ○ App-Server shows high ACK time.
  ○ Also a few retransmissions at both sites.

● Next steps
  ○ Check App-Server health!
  ○ Check App-Server application!
  ○ Take care of retransmissions later
Real-Life Case #1 - Solution

- What we found on the „Application Server“:
  - Application:
    - MS-Access „database“ with 1.2 GBytes in size.
    - Had not been reorganized for months
  - Machine itself:
    - Just 256 MBytes of RAM, high level of disk swapping
    - Machine was heavily overloaded (corresponding with high RTT)
- Fix
  - Reorganize the DB on App-Server → 1-Site1-after.pcapng
  - Add more RAM to the machine → scheduled for later
Real-Life Case #1 - Solution

- **Transaction after fix #1**
  - Transaction time: 8.3 s

- **Still present:**
  - High ACK-times at App-Srv: 200 ms
  - Still overloaded machine (RAM to be added)

- **Retransmissions due to packet loss**
  - Caused by **Duplex Mismatch** between APP-Server and DB-Server
Lessons learned - Case 1

- An overloaded App-Server caused high delays.
  - Filter out TCP-ACKs (TCP.len > 1)
  - Look at large Delta Times
  - Check Flow Graph
- Reorganizing the database helped
- Adding RAM helped as well
- Duplex mismatch between Switch and Router caused packet reordering and retransmission
- Important: Don‘t stop after you identified the first problem.
Case Study 3
Real-Life Case #3

- Case: Accounting software usage after moving Data Center to a new Service Provider
- Customer: Agency for temporary work.
- Problem: High transaction times, higher than baseline → Permanent
- Transaction: Login-Process for user
- Trace File: 3-VDI.pcapng
- Reference: 3-TS.pcapng
- Wireshark Profile: App-Analysis-III
- Suspects: „Check out everything!“
Case #3 - network map

**Normal operation:**
- VDI-Client ↔ App server

**Alternative solution:**
- VDI-Client ↔ Terminal Server
- TS ↔ App Server

- Normal client
  - VDI Client accesses app server directly
  - Problem: Longer transaction times

- Intermediate test
  - VDI-Client via Terminal Server
  - Result: Better transaction times

- Let’s start the analysis.
Case #3 - Capture Preps

- Problem affected all users using the main application
  - Replicable process
  - We selected a typical task where a baseline existed.
    - User Login
  - Traces were taken - data was filtered and isolated
    - 3 client traces showed similar figures
    - So it was really a repeatable process
  - Traces were taken for
    - Pure VDI users
    - Terminal Server test users
Case #3 - Analysis

● Transaction: User Login
  ○ VDI: 31 seconds, 18056 packets
  ○ TS: 10 seconds, 18035 packets
  ○ Baseline: 8 seconds, # of packets unknown

● Questions
  ○ What makes VDI so slow?
  ○ Why is TS much faster?
  ○ Where is the bottleneck?
  ○ How can we improve the performance?
Case #3 - Analysis

- **VDI-Client**
  - Filter out ACKs, then sort by Delta Times
  - Note the three high Delta Times, all from VDI Client
Case #3 - Analysis

- **VDI-Client**
  - Large gaps on client side
  - Longest is 12 seconds
Case #3 - Analysis

- Terminal Server
  - Lower values for Large Delta Times, all from Terminal Server
Case #3 - Analysis

- Terminal Server
  - Gaps on client side
  - Longest is 1.4 seconds
Case #3 - Resolution - Who’s to blame?

- Who is responsible for most of the increase in transaction time?
  - VDI-Client, Database Server, Terminal Server?

- Answer: It must be the ...
  - VDI-Client
  - → Check for reasons for the „high think times“ on VDI Client

- Action:
  - Improve the hardware of VDI host and the VDI software platform

- Findings
  - Improved the transaction time, but still worse than baseline.

- Is there still room for improvement?
  - Database Server?
  - Firewall?
Case #3 - Extras: Summarizing the Response Times

- Isolated Client and Server times
  - Client Idle Times:
    - VDI: 21.15 s
    - TS: 6.53 s
  - Server Response Times:
    - VDI: 9.94 s
    - TS: 3.41 s
Case #3 - Extras: How this was done

- VDI Client Metrics
- Total Time spent: 31 seconds
  - At VDI-Client: 21.15 s
  - Network + Server + Application: 9.94 s
Case #3 - Extras: How this was done!

- Terminal Server Metrics
- Total Time spent:
  - At Terminal Server: 6.53 s
  - Network + Server + Application: 3.41 s
  → which is 6.55 s less than SRT at VDI
Case#3 - Effect of Round Trip Time

Normal operation: VDI-Client – DB-Server

- One Round Trip:
  - Difference: 0.777 ms
  - ~ 8560 turns
  - Difference: 6.65 seconds
  - Added by the firewall

Alternative solution: TS – DB-Server

- Let’s remove the firewall!
- Was he right?

Guess what the customer responded, when we told him...
Lessons Learned - Case 3

- VDI Clients have to carry the load of the application!
  - Need enough performance!
  - Applications often show a tendency towards Terminal Services or to VDI
- Firewalls add extra delay
  - But was the firewall the real problem?
- A huge number of application turns has great impact on app performance
  - Small increase on RTT, but huge overall delay
- Would you remove the firewall or have the application rewritten?
Application Metrics for Database Applications

- **Key Metrics**
  - SRT at Front End vs.
  - Duration of database process

- **Additional Metrics**
  - SRTs at Database ($T_{DB}$)
  - App-Server “think“ times ($T_{AP}$)
  - Number of application turns
  - Round Trip Time
  - RTT * Number of Turns = Minimum Transaction Time
Lessons Learned

Database Applications are sensitive to

- High Server (DB) response times -> Slow database
- Long Client wait times -> Slow Application on Client or App-Srv
- Very sensitive to Round Trip Time (RTT)
  - Many application turns should be avoided
  - Programming Techniques: „Row by Row is slow by slow“
Questions from the chat?
Thank You!

- Thank you for listening!
- Please leave your feedback in the feedback portal.
  - [https://forms.gle/vELKPFgDobAMVC8n7](https://forms.gle/vELKPFgDobAMVC8n7)
  - Link also in Chat and published on SharkFest documents.
- For further questions meet me on Discord Server
  - Voice Channel: zoom 1 discussion
  - Starts in 5 minutes after this presentation ends
- Contact me
  - Matthias.Kaiser@experteach.de
  - Twitter: @wiresharky
End of Presentation
Case Study 2
Real-Life Case #2

- **Case:** New application in emergency room of a hospital
- **Real-Life Case #2:** Application freeze for users → Intermittent
- **Trace File:**
  - 2-Before-oneclient.pcapng
  - 2-Before-allclients.pcapng
- **Wireshark Profile:** App-Analysis-II
- **Suspects:** Network
- **Questions:**
  - Is it the network?
  - If not, where is the problem?
Real-Life Case #2 - network map

- Sample Client machines
  - 3 Virtual-Clients at Site 2, VDI clients at HQ.
  - 1 Fat-Client at Site 2
- All four users reported problems
  - „Application freeze“
  - Freezes > 10 seconds
  - App freezes, not the client.
- Intermittent Problem
Real-Life Case #2 - Methodology

- Identify communication pattern

Methodology

- Capture 4 clients simultaneously
  - Traces were captured on fat client and on VDI clients.
- Ask users to note application freeze times
- Try to correlate noted freeze times to packets in the trace files

Analysis

- Check network performance (RTT and TCP errors)
  - RTT: 4ms, No Errors
- Then check Server Response Times
- And check Server Performance
Real-Life Case #2 - Analysis

- **Communication Pattern**
  - HTTP POST →
  - ← HTTP/1.1 200 OK
  - http.time shows Application Response Time

- **Task**
  - Identify high values for SRT for HTTP
  - Correlate with times, when users noted an application freeze
Real-Life Case #2 - Analysis

- First check with one client
  - High values for SRT.
  - High SRT values correlated with application freeze.
  - High SRT values showed random timing

- Next step
  - Long term capture on 4 clients
Setting up the long term capture

- Long term capture with tshark
  - Batch file to start tshark for 1 day
    - `tshark -i 2 -w file.pcapng -B 200 -a duration:86400 -b filesize:200000`
  - Batch file put into Windows Task Scheduler
    - Starting after Login with SYSTEM rights (not interactive)

- First steps
  - Automated trace file processing with tshark and mergecap
    - Merge related files with mergecap
    - Filter by ip address and http packet with tshark
    - `tshark -2 -r infile.pcapng -Y "filter-expr" -w outfile.pcapng"`
Case#2 - Analysis

- Evaluate the file via tshark script

```sh
@call "C:\Program Files\wireshark\tshark.exe -2 -r filtered-b.pcapng -Y "http.time > 2" -T fields -e frame.number -e frame.time -e ip.src -e ip.dst -e http.time > before.txt
```

- Result

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>IP Address</th>
<th>HTTP Time</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>6475</td>
<td>Jan 20, 2015 11:06:35</td>
<td>9343431000</td>
<td>172.20.100.51</td>
<td>172.20.10.130</td>
</tr>
<tr>
<td>6490</td>
<td>Jan 20, 2015 11:06:38</td>
<td>1678150000</td>
<td>172.20.100.51</td>
<td>172.20.10.128</td>
</tr>
<tr>
<td>9246</td>
<td>Jan 20, 2015 11:33:44</td>
<td>9425760000</td>
<td>172.20.100.51</td>
<td>172.20.10.130</td>
</tr>
<tr>
<td>9271</td>
<td>Jan 20, 2015 11:33:47</td>
<td>2097550000</td>
<td>172.20.100.51</td>
<td>172.20.10.128</td>
</tr>
<tr>
<td>9304</td>
<td>Jan 20, 2015 11:33:49</td>
<td>7437620000</td>
<td>172.20.100.51</td>
<td>172.32.10.16</td>
</tr>
<tr>
<td>9387</td>
<td>Jan 20, 2015 11:34:14</td>
<td>1576190000</td>
<td>172.20.100.51</td>
<td>172.32.10.16</td>
</tr>
<tr>
<td>9750</td>
<td>Jan 20, 2015 11:36:17</td>
<td>7144570000</td>
<td>172.20.100.51</td>
<td>172.20.10.130</td>
</tr>
<tr>
<td>9772</td>
<td>Jan 20, 2015 11:36:21</td>
<td>6683720000</td>
<td>172.20.100.51</td>
<td>172.20.10.130</td>
</tr>
<tr>
<td>9783</td>
<td>Jan 20, 2015 11:36:22</td>
<td>5255440000</td>
<td>172.20.100.51</td>
<td>172.32.10.16</td>
</tr>
<tr>
<td>9795</td>
<td>Jan 20, 2015 11:36:26</td>
<td>4909590000</td>
<td>172.20.100.51</td>
<td>172.32.10.16</td>
</tr>
<tr>
<td>9797</td>
<td>Jan 20, 2015 11:36:36</td>
<td>5574600000</td>
<td>172.20.100.51</td>
<td>172.20.10.130</td>
</tr>
<tr>
<td>9816</td>
<td>Jan 20, 2015 11:36:46</td>
<td>7620040000</td>
<td>172.20.100.51</td>
<td>172.20.10.130</td>
</tr>
</tbody>
</table>
Case#2 - Analysis

- Evaluate via i/O graph
  - HTTP peak SRT values
  - MAX(http.time)
Case#2 - Analysis

- Findings:
  - Server Related Problems
- At Server
  - Processes for IIS and MS-SQL went up to 98% CPU utilization every now and then (always together)
- From trace file
  - High response time were only seen when one specific transaction was issued from the client.
- This was reported to the company who wrote this application ...
Case #2 - Solution

- **Surprise:** They listened and found a problem
- **Software Update:** We were asked to check performance again
- **Trace files:** 2-After-oneclient.pcapng
  2-After-allclients.pcapng
- **Wireshark Profile:** App-Analysis-II
Real-Life Case #2 - Solution

- **Result**
  - Most of the high values for SRT were gone.
  - No app freeze noted by users any more.

- **Still open**
  - Response times of 1 s
Case#2 - Solution

- Evaluate via i/O graph
  - HTTP peak SRT values
  - http.time -> Max
Lessons Learned - Case 2

- Simple application Pattern
- No network problems
- High response times at Appserver
  - High load on Database Service
  - Timeout at Webserver
  - Specific application calls hung
- If it is not the network, check on the server side.