Sharkfest 2013

Work Book

Presentation Material

© Copyright 1995 - 2013 Advance Seven Limited. All rights reserved.

Advance Seven Limited
Melville House
High Street
Dunmow
Essex CM6 1AF
United Kingdowm

Table of Contents

1 TMS HTTP Trace 1

2 TMS Database Trace – Scenario 1 2
Summary Trace 2
Database Performance Analysis 4

3 TMS Database Trace – Scenario 2 6
Summary Trace 6
Response Packet 7
Request Packet 8

4 Correlation Strategies 9
Introduction 9
TCP Port-to-port Mapping 9
Data Content 12
Multi-tier Characterisation 13

1 TMS HTTP Trace

No.	Time	Source	Destination	Src Port	Dst Port	Info
48	11:42:18.580075	192.168.1.70	192.168.1.77	59635	80	GET /TicketView.php?TicketNo=511161 HTTP/1.1
50	11:42:18.749026	192.168.1.77	192.168.1.70	80	59635	[TCP segment of a reassembled PDU]
51	11:42:18.749254	192.168.1.77	192.168.1.70	80	59635	[TCP segment of a reassembled PDU]
52	11:42:18.749395	192.168.1.77	192.168.1.70	80	59635	[TCP segment of a reassembled PDU]
53	11:42:18.749671	192.168.1.77	192.168.1.70	80	59635	HTTP/1.1 200 OK (text/html)
64	11:42:22.924800	192.168.1.70	192.168.1.77	59636	80	GET /TicketView.php?TicketNo=510005 HTTP/1.1
66	11:42:23.071154	192.168.1.77	192.168.1.70	80	59636	[TCP segment of a reassembled PDU]
67	11:42:23.071396	192.168.1.77	192.168.1.70	80	59636	[TCP segment of a reassembled PDU]
68	11:42:23.071536	192.168.1.77	192.168.1.70	80	59636	[TCP segment of a reassembled PDU]
69	11:42:23.071773	192.168.1.77	192.168.1.70	80	59636	HTTP/1.1 200 OK (text/html)
88	11:42:29.073392	192.168.1.70	192.168.1.77	59638	80	GET /TicketView.php?TicketNo=511188 HTTP/1.1
90	11:42:29.228525	192.168.1.77	192.168.1.70	80	59638	[TCP segment of a reassembled PDU]
91	11:42:29.228754	192.168.1.77	192.168.1.70	80	59638	[TCP segment of a reassembled PDU]
92	11:42:29.228894	192.168.1.77	192.168.1.70	80	59638	[TCP segment of a reassembled PDU]
94	11:42:29.232555	192.168.1.77	192.168.1.70	80	59638	[TCP segment of a reassembled PDU]
95	11:42:29.232667	192.168.1.77	192.168.1.70	80	59638	HTTP/1.1 200 OK (text/html)
107	11:42:32.527742	192.168.1.70	192.168.1.77	59639	80	GET /TicketView.php?TicketNo=511152 HTTP/1.1
109	11:42:32.676012	192.168.1.77	192.168.1.70	80	59639	[TCP segment of a reassembled PDU]
110	11:42:32.676241	192.168.1.77	192.168.1.70	80	59639	[TCP segment of a reassembled PDU]
111	11:42:32.676381	192.168.1.77	192.168.1.70	80	59639	[TCP segment of a reassembled PDU]
112	11:42:32.676615	192.168.1.77	192.168.1.70	80	59639	HTTP/1.1 200 OK (text/html)
124	11:42:36.622843	192.168.1.70	192.168.1.77	59640	80	GET /TicketView.php?TicketNo=511129 HTTP/1.1
153	11:42:42.770757	192.168.1.77	192.168.1.70	80	59640	[TCP segment of a reassembled PDU]
154	11:42:42.771037	192.168.1.77	192.168.1.70	80	59640	[TCP segment of a reassembled PDU]
155	11:42:42.772202	192.168.1.77	192.168.1.70	80	59640	[TCP segment of a reassembled PDU]
156	11:42:42.772761	192.168.1.77	192.168.1.70	80	59640	HTTP/1.1 200 OK (text/html)
177	11:42:52.932842	192.168.1.70	192.168.1.77			Echo (ping) request id=0x0001, seq=19439/61259,
178	11:42:52.932902	192.168.1.77	192.168.1.70			Echo (ping) reply id=0x0001, seq=19439/61259,

2 TMS Database Trace – Scenario 1

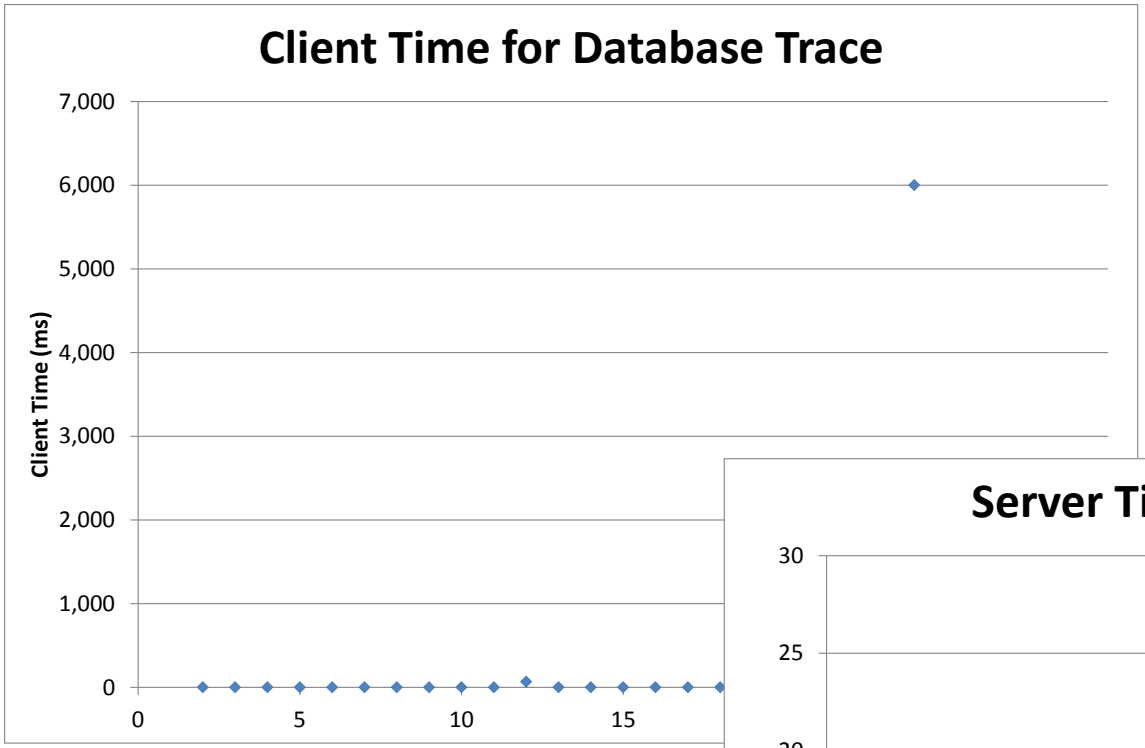
Summary Trace

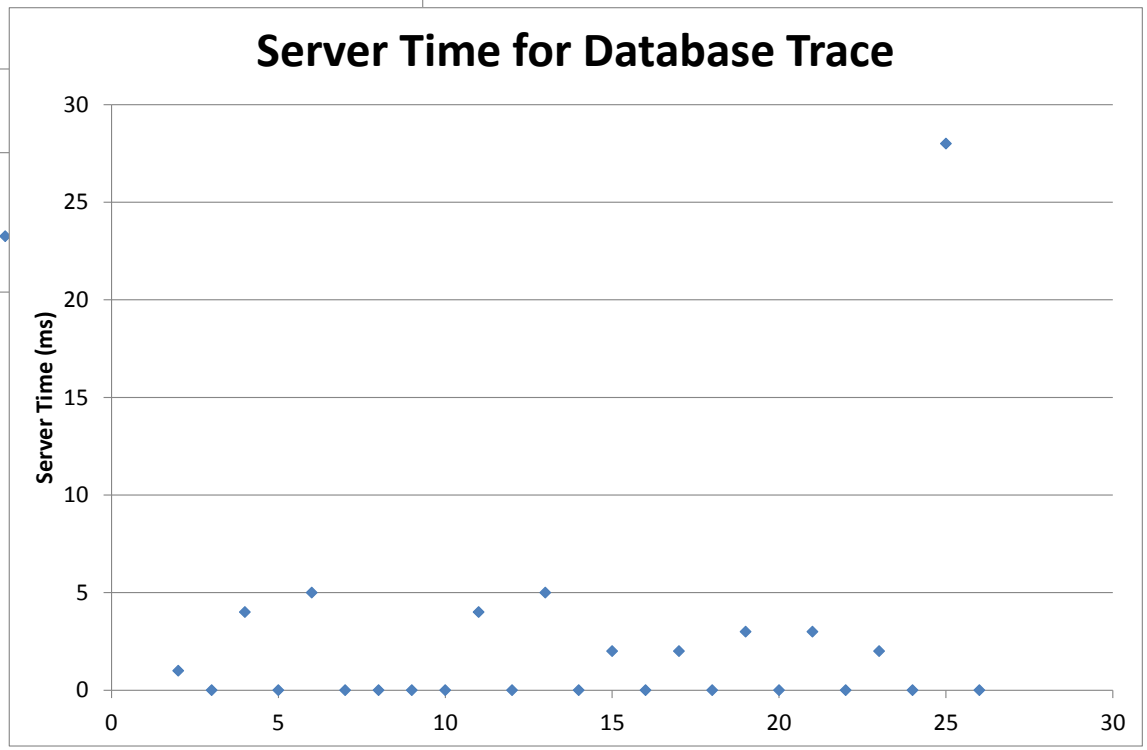
No.	Time	Source	Destination	Src Port	Dst Port	Info
Trace lines deleted						
145	11:42:29.095696	127.0.0.1	127.0.0.1	34573	5432	>
147	11:42:29.096301	127.0.0.1	127.0.0.1	5432	34573	[TCP segment of a reassembled PDU]
149	11:42:29.096406	127.0.0.1	127.0.0.1	34573	5432	>
152	11:42:29.099997	127.0.0.1	127.0.0.1	5432	34573	[TCP segment of a reassembled PDU]
154	11:42:29.172001	127.0.0.1	127.0.0.1	34573	5432	>Q
155	11:42:29.177176	127.0.0.1	127.0.0.1	5432	34573	[TCP segment of a reassembled PDU]
157	11:42:29.178424	127.0.0.1	127.0.0.1	34573	5432	>Q
158	11:42:29.179786	127.0.0.1	127.0.0.1	5432	34573	[TCP segment of a reassembled PDU]
159	11:42:29.180045	127.0.0.1	127.0.0.1	34573	5432	>Q
160	11:42:29.181759	127.0.0.1	127.0.0.1	5432	34573	[TCP segment of a reassembled PDU]
161	11:42:29.182083	127.0.0.1	127.0.0.1	34573	5432	>Q
162	11:42:29.184454	127.0.0.1	127.0.0.1	5432	34573	[TCP segment of a reassembled PDU]
163	11:42:29.184963	127.0.0.1	127.0.0.1	34573	5432	>Q
164	11:42:29.188314	127.0.0.1	127.0.0.1	5432	34573	[TCP segment of a reassembled PDU]
165	11:42:29.188676	127.0.0.1	127.0.0.1	34573	5432	>Q
166	11:42:29.189526	127.0.0.1	127.0.0.1	5432	34573	[TCP segment of a reassembled PDU]
167	11:42:29.190018	127.0.0.1	127.0.0.1	34573	5432	>Q
168	11:42:29.219041	127.0.0.1	127.0.0.1	5432	34573	[TCP segment of a reassembled PDU]
169	11:42:29.224530	127.0.0.1	127.0.0.1	34573	5432	>X
181	11:42:32.528652	127.0.0.1	127.0.0.1	34574	5432	>
183	11:42:32.529778	127.0.0.1	127.0.0.1	5432	34574	[TCP segment of a reassembled PDU]
185	11:42:32.529887	127.0.0.1	127.0.0.1	34574	5432	>
188	11:42:32.534048	127.0.0.1	127.0.0.1	5432	34574	<R/S/S/S/S/S/S/S/S/S/S/K/Z
189	11:42:32.534169	127.0.0.1	127.0.0.1	34574	5432	>Q
190	11:42:32.538803	127.0.0.1	127.0.0.1	5432	34574	<T/D/C/Z
191	11:42:32.538916	127.0.0.1	127.0.0.1	34574	5432	>X
202	11:42:32.548039	127.0.0.1	127.0.0.1	34575	5432	>
204	11:42:32.548625	127.0.0.1	127.0.0.1	5432	34575	[TCP segment of a reassembled PDU]
206	11:42:32.548730	127.0.0.1	127.0.0.1	34575	5432	>
209	11:42:32.552327	127.0.0.1	127.0.0.1	5432	34575	[TCP segment of a reassembled PDU]
211	11:42:32.619972	127.0.0.1	127.0.0.1	34575	5432	>Q
212	11:42:32.625552	127.0.0.1	127.0.0.1	5432	34575	[TCP segment of a reassembled PDU]

```
214 11:42:32.626767 127.0.0.1       127.0.0.1       34575    5432     >Q
215 11:42:32.628191 127.0.0.1       127.0.0.1       5432     34575    [TCP segment of a reassembled PDU]
216 11:42:32.628431 127.0.0.1       127.0.0.1       34575    5432     >Q
217 11:42:32.630162 127.0.0.1       127.0.0.1       5432     34575    [TCP segment of a reassembled PDU]
218 11:42:32.630457 127.0.0.1       127.0.0.1       34575    5432     >Q
219 11:42:32.632775 127.0.0.1       127.0.0.1       5432     34575    [TCP segment of a reassembled PDU]
220 11:42:32.633281 127.0.0.1       127.0.0.1       34575    5432     >Q
221 11:42:32.636568 127.0.0.1       127.0.0.1       5432     34575    [TCP segment of a reassembled PDU]
222 11:42:32.636936 127.0.0.1       127.0.0.1       34575    5432     >Q
223 11:42:32.637757 127.0.0.1       127.0.0.1       5432     34575    [TCP segment of a reassembled PDU]
224 11:42:32.638241 127.0.0.1       127.0.0.1       34575    5432     >Q
225 11:42:32.666828 127.0.0.1       127.0.0.1       5432     34575    [TCP segment of a reassembled PDU]
226 11:42:32.672119 127.0.0.1       127.0.0.1       34575    5432     >X
238 11:42:36.623803 127.0.0.1       127.0.0.1       34576    5432     >
240 11:42:36.624993 127.0.0.1       127.0.0.1       5432     34576    [TCP segment of a reassembled PDU]
242 11:42:36.625101 127.0.0.1       127.0.0.1       34576    5432     >
245 11:42:36.628643 127.0.0.1       127.0.0.1       5432     34576    [TCP segment of a reassembled PDU]
246 11:42:36.628872 127.0.0.1       127.0.0.1       34576    5432     >Q
247 11:42:36.634027 127.0.0.1       127.0.0.1       5432     34576    [TCP segment of a reassembled PDU]
248 11:42:36.634141 127.0.0.1       127.0.0.1       34576    5432     >X
259 11:42:36.643659 127.0.0.1       127.0.0.1       34577    5432     >
261 11:42:36.644303 127.0.0.1       127.0.0.1       5432     34577    [TCP segment of a reassembled PDU]
263 11:42:36.644409 127.0.0.1       127.0.0.1       34577    5432     >
266 11:42:36.648025 127.0.0.1       127.0.0.1       5432     34577    [TCP segment of a reassembled PDU]
268 11:42:36.715297 127.0.0.1       127.0.0.1       34577    5432     >Q
269 11:42:36.720471 127.0.0.1       127.0.0.1       5432     34577    [TCP segment of a reassembled PDU]
271 11:42:36.721194 127.0.0.1       127.0.0.1       34577    5432     >Q
272 11:42:36.723085 127.0.0.1       127.0.0.1       5432     34577    [TCP segment of a reassembled PDU]
273 11:42:36.723338 127.0.0.1       127.0.0.1       34577    5432     >Q
274 11:42:36.725111 127.0.0.1       127.0.0.1       5432     34577    [TCP segment of a reassembled PDU]
275 11:42:36.725416 127.0.0.1       127.0.0.1       34577    5432     >Q
276 11:42:36.727759 127.0.0.1       127.0.0.1       5432     34577    [TCP segment of a reassembled PDU]
277 11:42:36.728204 127.0.0.1       127.0.0.1       34577    5432     >Q
278 11:42:36.730527 127.0.0.1       127.0.0.1       5432     34577    [TCP segment of a reassembled PDU]
279 11:42:36.731211 127.0.0.1       127.0.0.1       34577    5432     >Q
280 11:42:36.732775 127.0.0.1       127.0.0.1       5432     34577    [TCP segment of a reassembled PDU]
282 11:42:42.733601 127.0.0.1       127.0.0.1       34577    5432     >Q
288 11:42:42.762467 127.0.0.1       127.0.0.1       5432     34577    [TCP segment of a reassembled PDU]
290 11:42:42.767720 127.0.0.1       127.0.0.1       34577    5432     >X
```

Database Performance Analysis

No.	Time	Client Time (ms)	Req Spread (ms)	Rsp Spread (ms)	Server Time (ms)	Source	Destination	Client IP	Client Port	Src Port	Dst Port	Info
238	11:42:36.624					127.0.0.1	127.0.0.1	127.0.0.1	34576	34576	5432	>
240	11:42:36.625				1	127.0.0.1	127.0.0.1	127.0.0.1	34576	5432	34576	[TCP segment of a reassembled PDU]
242	11:42:36.625	0				127.0.0.1	127.0.0.1	127.0.0.1	34576	34576	5432	>
245	11:42:36.629				4	127.0.0.1	127.0.0.1	127.0.0.1	34576	5432	34576	[TCP segment of a reassembled PDU]
246	11:42:36.629	0				127.0.0.1	127.0.0.1	127.0.0.1	34576	34576	5432	>Q
247	11:42:36.634				5	127.0.0.1	127.0.0.1	127.0.0.1	34576	5432	34576	[TCP segment of a reassembled PDU]
248	11:42:36.634	0				127.0.0.1	127.0.0.1	127.0.0.1	34576	34576	5432	>X
259	11:42:36.644					127.0.0.1	127.0.0.1	127.0.0.1	34577	34577	5432	>
261	11:42:36.644				0	127.0.0.1	127.0.0.1	127.0.0.1	34577	5432	34577	[TCP segment of a reassembled PDU]
263	11:42:36.644	0				127.0.0.1	127.0.0.1	127.0.0.1	34577	34577	5432	>
266	11:42:36.648				4	127.0.0.1	127.0.0.1	127.0.0.1	34577	5432	34577	[TCP segment of a reassembled PDU]
268	11:42:36.715	67				127.0.0.1	127.0.0.1	127.0.0.1	34577	34577	5432	>Q
269	11:42:36.720				5	127.0.0.1	127.0.0.1	127.0.0.1	34577	5432	34577	[TCP segment of a reassembled PDU]
271	11:42:36.721	1				127.0.0.1	127.0.0.1	127.0.0.1	34577	34577	5432	>Q
272	11:42:36.723				2	127.0.0.1	127.0.0.1	127.0.0.1	34577	5432	34577	[TCP segment of a reassembled PDU]
273	11:42:36.723	0				127.0.0.1	127.0.0.1	127.0.0.1	34577	34577	5432	>Q
274	11:42:36.725				2	127.0.0.1	127.0.0.1	127.0.0.1	34577	5432	34577	[TCP segment of a reassembled PDU]
275	11:42:36.725	0				127.0.0.1	127.0.0.1	127.0.0.1	34577	34577	5432	>Q
276	11:42:36.728				3	127.0.0.1	127.0.0.1	127.0.0.1	34577	5432	34577	[TCP segment of a reassembled PDU]
277	11:42:36.728	0				127.0.0.1	127.0.0.1	127.0.0.1	34577	34577	5432	>Q
278	11:42:36.731				3	127.0.0.1	127.0.0.1	127.0.0.1	34577	5432	34577	[TCP segment of a reassembled PDU]
279	11:42:36.731	0				127.0.0.1	127.0.0.1	127.0.0.1	34577	34577	5432	>Q
280	11:42:36.733				2	127.0.0.1	127.0.0.1	127.0.0.1	34577	5432	34577	[TCP segment of a reassembled PDU]
282	11:42:42.734	6,001				127.0.0.1	127.0.0.1	127.0.0.1	34577	34577	5432	>Q
288	11:42:42.762				28	127.0.0.1	127.0.0.1	127.0.0.1	34577	5432	34577	[TCP segment of a reassembled PDU]
290	11:42:42.768	6				127.0.0.1	127.0.0.1	127.0.0.1	34577	34577	5432	>X
		6,075	0	0	59							


Client Time for Database Trace
Client Time (ms)
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
0
5
10
15


Server Time for Database Trace
Server Time (ms)
30
25
20
15
10
5
0
0
5
10
15
20
25
30

3 TMS Database Trace – Scenario 2

Summary Trace

No.	Time	Source	Destination	Src Port	Dst Port	Info
Trace lines deleted						
105	11:42:36.623803	127.0.0.1	127.0.0.1	34576	5432	>
106	11:42:36.624993	127.0.0.1	127.0.0.1	5432	34576	[TCP segment of a reassembled PDU]
107	11:42:36.625101	127.0.0.1	127.0.0.1	34576	5432	>
108	11:42:36.628643	127.0.0.1	127.0.0.1	5432	34576	[TCP segment of a reassembled PDU]
109	11:42:36.628872	127.0.0.1	127.0.0.1	34576	5432	>Q
110	11:42:36.634027	127.0.0.1	127.0.0.1	5432	34576	[TCP segment of a reassembled PDU]
111	11:42:36.634141	127.0.0.1	127.0.0.1	34576	5432	>X
112	11:42:36.643659	127.0.0.1	127.0.0.1	34577	5432	>
113	11:42:36.644303	127.0.0.1	127.0.0.1	5432	34577	[TCP segment of a reassembled PDU]
114	11:42:36.644409	127.0.0.1	127.0.0.1	34577	5432	>
115	11:42:36.648025	127.0.0.1	127.0.0.1	5432	34577	[TCP segment of a reassembled PDU]
116	11:42:36.715297	127.0.0.1	127.0.0.1	34577	5432	>Q
117	11:42:36.720471	127.0.0.1	127.0.0.1	5432	34577	[TCP segment of a reassembled PDU]
118	11:42:36.721194	127.0.0.1	127.0.0.1	34577	5432	>Q
119	11:42:36.723085	127.0.0.1	127.0.0.1	5432	34577	[TCP segment of a reassembled PDU]
120	11:42:36.723338	127.0.0.1	127.0.0.1	34577	5432	>Q
121	11:42:36.725111	127.0.0.1	127.0.0.1	5432	34577	[TCP segment of a reassembled PDU]
122	11:42:36.725416	127.0.0.1	127.0.0.1	34577	5432	>Q
123	11:42:36.727759	127.0.0.1	127.0.0.1	5432	34577	[TCP segment of a reassembled PDU]
124	11:42:36.728204	127.0.0.1	127.0.0.1	34577	5432	>Q
125	11:42:36.730527	127.0.0.1	127.0.0.1	5432	34577	[TCP segment of a reassembled PDU]
126	11:42:36.731211	127.0.0.1	127.0.0.1	34577	5432	>Q
127	11:42:36.732775	127.0.0.1	127.0.0.1	5432	34577	[TCP segment of a reassembled PDU]
128	11:42:36.733601	127.0.0.1	127.0.0.1	34577	5432	>Q
129	11:42:42.762467	127.0.0.1	127.0.0.1	5432	34577	[TCP segment of a reassembled PDU]
130	11:42:42.767720	127.0.0.1	127.0.0.1	34577	5432	>X

Response Packet

```
Frame 129: 251 bytes on wire (2008 bits), 251 bytes captured (2008 bits)
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_0
Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1
Transmission Control Protocol, Src Port: postgresql (5432), Dst Port: 34
```

```
00  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
10  00 ed 2e 62 40 00 40 06  0d a7 7f 00 00 01 7f 00   ...b@.@. ........
20  00 01 15 38 87 11 90 85  d5 04 85 55 db 6d 80 18   ...8.... ...U.m..
30  02 11 fe e1 00 00 01 01  08 0a 05 34 b6 40 05 34   ........ ...4.@.4
40  b6 23 54 00 00 00 a6 00  06 74 69 63 6b 65 74 5f   .#T..... .ticket_
50  6e 6f 00 00 00 a8 81 00  01 00 00 00 17 00 04 ff   no...... ........
60  ff ff ff 00 00 75 73 65  72 69 64 00 00 00 a8 81   .....use rid.....
70  00 02 00 00 04 13 ff ff  00 00 00 24 00 00 64 74   ........ ...$..dt
80  5f 75 70 64 61 74 65 00  00 00 a8 81 00 03 00 00   _update. ........
90  04 5a 00 08 ff ff ff ff  00 00 74 61 67 5f 6c 69   .Z...... ..tag_li
a0  6e 65 00 00 00 a8 81 00  04 00 00 04 13 ff ff 00   ne...... ........
b0  00 00 34 00 00 69 6e 66  6f 00 00 00 a8 81 00 05   ..4..inf o.......
c0  00 00 04 13 ff ff 00 00  28 04 00 00 61 63 74 69   ........ (...acti
d0  76 69 74 79 69 64 00 00  00 a8 81 00 06 00 00 00   vityid.. ........
e0  17 00 04 ff ff ff ff 00  00 43 00 00 00 0b 53 45   ........ .C....SE
f0  4c 45 43 54 00 5a 00 00  00 05 49                  LECT.Z.. ..I
```

```
156 11:42:42.772761  192.168.1.77    192.168.1.70    80      59640   HTTP/1.1 200 OK
```

```
\r\n
        <tr> \r\n
          <th width="110">Date/Time Queued</th>\r\n
          <th width="90">Ticket#</th>\r\n
          <th width="240">Description</th>\r\n
          <th width="80">Pri</th>\r\n
          <th width="80">Status</th>\r\n
        </tr>\r\n
\r\n
        <tr> \r\n
          <td>2012-08-03 07:40</td>\r\n
          <td>511129</td>\r\n
          <td>PSG Create - Communications</td>\r\n
          <td>3</td>\r\n
          <td class="alarm">Alarm</td>\r\n
        </tr>\r\n
\r\n
```

Request Packet

```
Frame 128: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits)
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_0
Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 34577 (34577), Dst Port: postgr
```

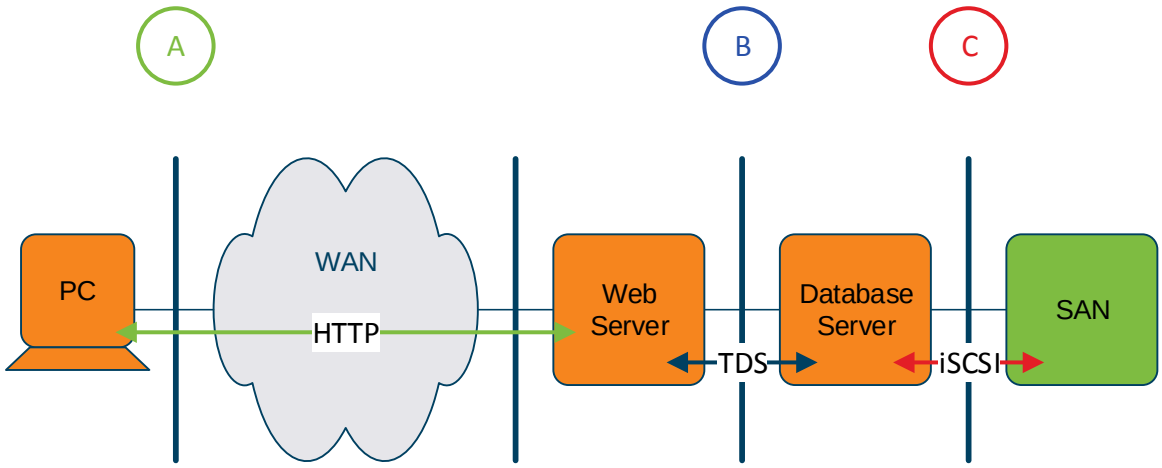
```
00  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
10  00 80 47 5d 40 00 40 06  f5 18 7f 00 00 01 7f 00   ..G]@.@. ........
20  00 01 87 11 15 38 85 55  db 21 90 85 d5 04 80 18   .....8.U .!......
30  03 02 fe 74 00 00 01 01  08 0a 05 34 b6 23 05 34   ...t.... ...4.#.4
40  9e b2 51 00 00 00 4b 53  45 4c 45 43 54 20 20 2a   ..Q...KS ELECT  *
50  20 46 52 4f 4d 20 20 61  63 74 69 76 69 74 79 20    FROM  a ctivity
60  57 48 45 52 45 20 20 74  69 63 6b 65 74 5f 6e 6f   WHERE  t icket_no
70  20 3d 20 35 31 31 31 32  39 20 4f 52 44 45 52 20    = 51112 9 ORDER
80  42 59 20 64 74 5f 75 70  64 61 74 65 20 00         BY dt_up date .
```

```
Ethernet II, Src: Netgear_a3:5b:a9 (00:09:5b:a3:5b:a9), Dst: IntelCor_73:a5:11 (00:13:ce:73:
Internet Protocol Version 4, Src: 192.168.1.70 (192.168.1.70), Dst: 192.168.1.77 (192.168.1.
Transmission Control Protocol, Src Port: 59640 (59640), Dst Port: http (80), Seq: 3292362848
Hypertext Transfer Protocol
⊞ GET /TicketView.php?TicketNo=511129 HTTP/1.1\r\n
  Host: 192.168.1.77\r\n
  User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:20.0) Gecko/20100101 Firefox/20.0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
  Accept-Language: en-US,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate\r\n
  Referer: http://192.168.1.77/QueueSummary.php?Qid=A7GPJO\r\n
⊞ Authorization: Basic cGF1bG9mOkExd2lhcDB5\r\n
```

4 Correlation Strategies

Introduction

As we track the execution of a user transaction through a system we have to deal with multiple protocols.



In the system above we have HTTP flowing from Internet Explorer to the web server. The web server may need to make calls to the database, and so perhaps we will have to use the Microsoft SQL protocol TDS. The database may then access its storage via iSCSI. There isn't a one-to-one match between the packets flowing at each interface (A, B and C), and so we need strategies to match packets at one tier to packets at another.

There are three strategies available:

- TCP port-to-port mapping
- Data content
- Comparison with a sample transaction

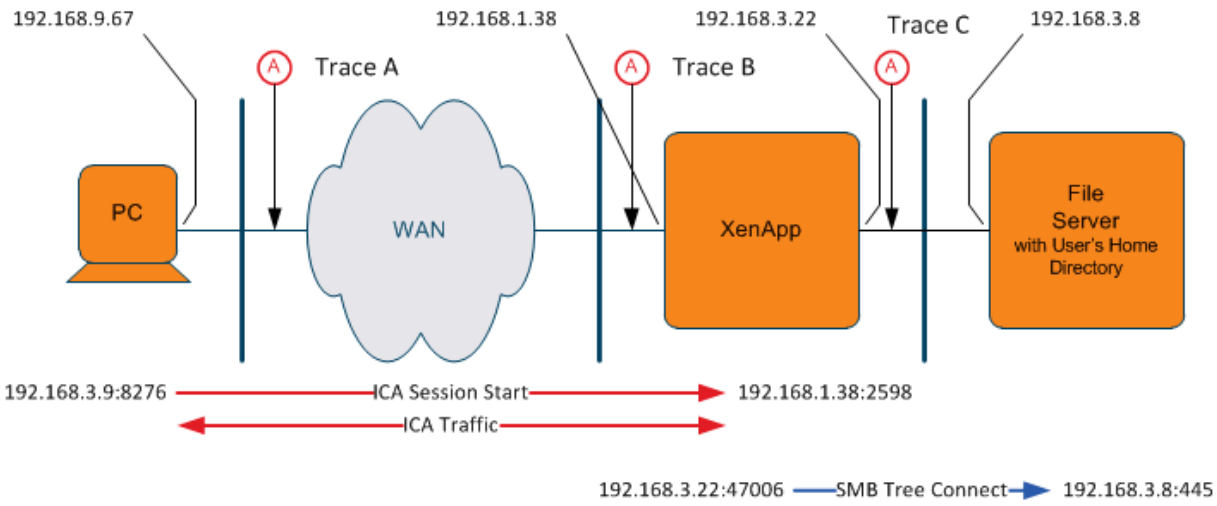
TCP Port-to-port Mapping

In this case we rely on the situation where a TCP connection from one client to a server has a constant relationship with a connection between another client and another server.

NB: In examples showing packets, time increases down the page.

Citrix XenApp Example

Consider the situation where a user's desktop is delivered from a Citrix XenApp server. In this case the user PC will maintain one (or three) constant connections to the Citrix server to carry the ICA presentation protocol. Once the user completes the login, there will be TCP connections to network file servers. For the duration of the login, there is a direct and constant relationship between ICA connections and the file server connections.



Therefore if we capture the creation of these connections at login time it's relatively easy to identify these mappings.

- PC to the XenApp server – simply filter to select ICA traffic from and to the PC and note the time of day that the PC started the connection
- XenApp to file server – filter the appropriate trace to show TCP SYN packets to the file server (IP address and port number) and find sessions that start shortly after the ICA connection is established

Once a XenApp to file server session has been identified, cross-check by using the 'follow TCP stream' Wireshark feature to select just that stream. Add to the stream filter:

- smb2.cmd == 0x03 - to view all SMB 2 TreeConnect requests with share information
- smb2.cmd == 0x05 - to view all SMB 2 Create requests with file name information
- smb.cmd == 0x32 – to view all SMB 1 Query Path Info packets with path information
- smb.cmd == 0xa2 – to view all SMB 1 NT Create AndX requests with file name information

Do the paths and file names seem appropriate for the user? Perhaps they actually include the userid. An identifiable marker can help. For example, ask the user to make a copy of a file and then rename the copy to something distinctive, such as 'marker101.docx'. You can then use this marker to check the TCP session.

In the above example we see that the SMB 2 connection defined by the quadruplet 192.168.3.22:47006:192.168.3.8:445 'belongs' to the ICA session 192.168.3.9:8276:192.168.1.38:2598

A further method is to;

- use the command netstat –b to see all connections and the process (PID) that owns them, then
- use Task Manager to identify the User Name for the PID

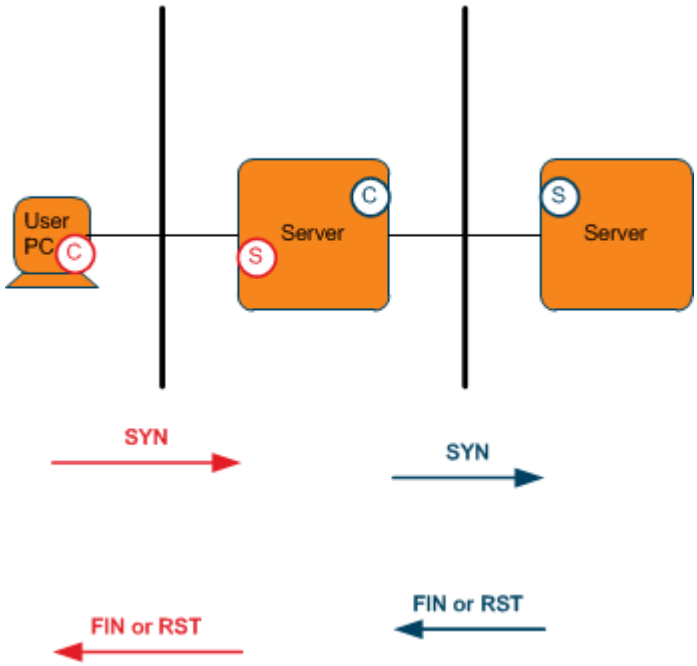
Of course this has to be done at the time, and it won't work with SMB connections since they are owned by the system.

The final way is to run Process Monitor (procmon) in conjunction with the network trace, and then review the .pml file later. Procmon shows the TCP Send and Receive details including userid. Again SMB is a problem here as the owning PID will always be 4 (system), but by looking at the file operations and paths specified it's relatively easy to work out who owns what.

Redirected file operations are a problem though since they all appear to flow on one TCP connection. In this case, content matching is needed.

Web Server to Application Server Example

Although the connections from a browser to a web server are often short-lived, there will probably be a connection from the web server to an application server (or other service) for the duration of the browser to web server connection.



The same principle as in the XenApp example applies; we determine the timestamp for the start and end of the browser to web server connection. We then look for TCP connections between the web server and the application server that start and end at around the same time.

Again, we can cross-check the findings by checking the data being carried on each connection and looking for distinctions. Because these connections are typically short-lived, it's not possible to use a marker for correlation.

Summary

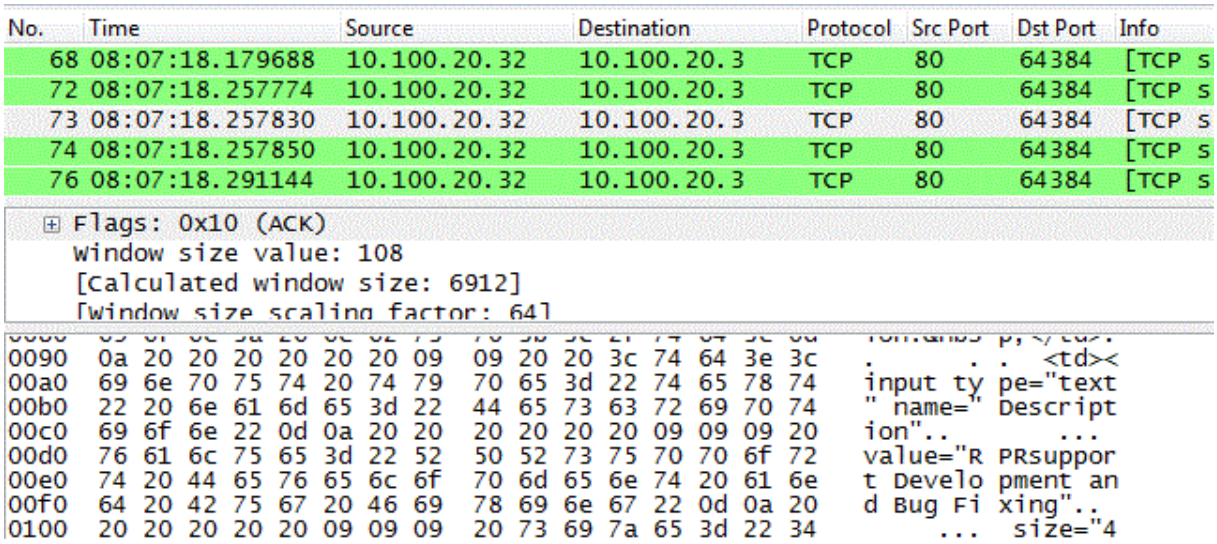
We have covered two examples to illustrate this general principle. The above technique will also work with other scenarios such as:

- User to Outlook on Citrix or terminal services to Exchange
- User to web or application server to database connections (although most now use connection pooling)
- User to a proxy or gateway server to web or application server

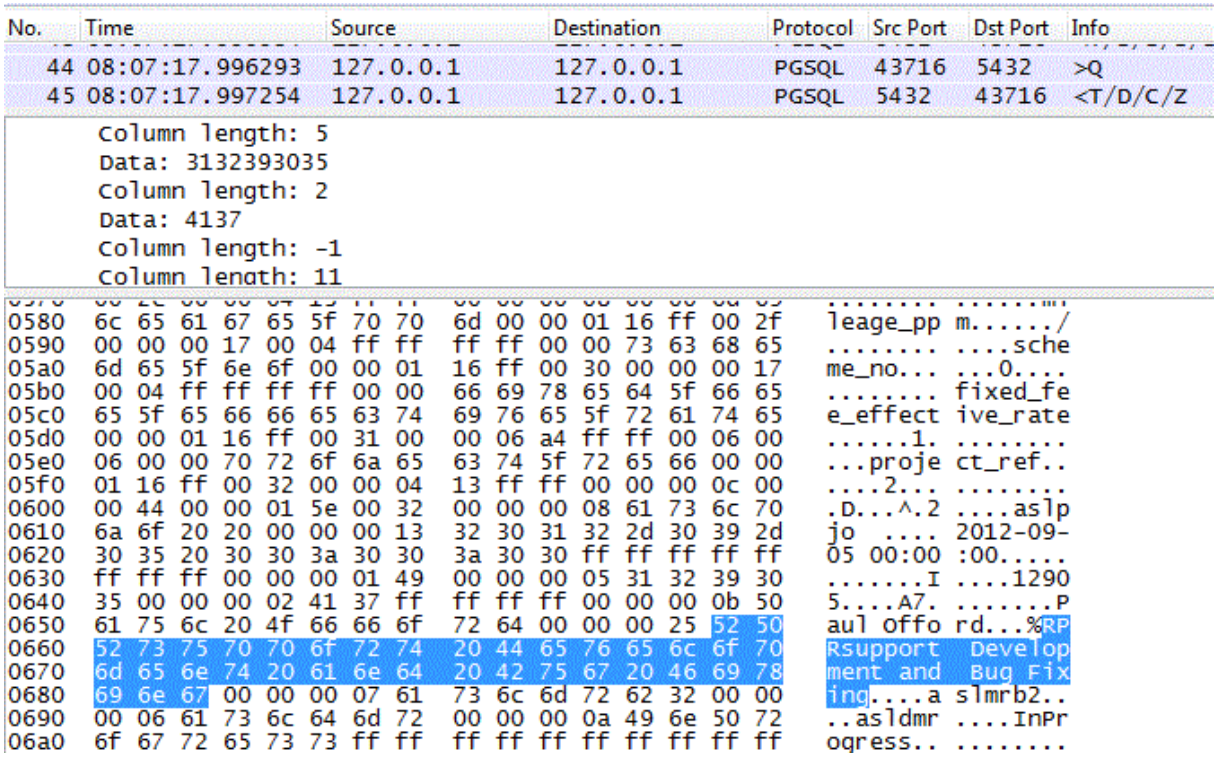
If the traces you capture are reasonably well synchronised, and you can't spot the patterns we have covered here then you'll need to consider a different approach to correlation.

Data Content

The principle here is very simple. If we have a multi-tiered system, application data that is delivered to the user was probably retrieved from another server such as a database. Therefore, if we know what data was delivered to the user we can search for that data at other tiers.



The above trace shows HTML data being sent from a web server to the user's browser. A screenshot of the browser shows a variable description value of 'RPRsupport Development and Bug Fixing'. This can be clearly seen in the trace data.



A corresponding trace was captured for traffic to and from the system's database. A search for the description string in the database trace, within the timeframe of the web transaction, quickly identifies this data being retrieved from the database.

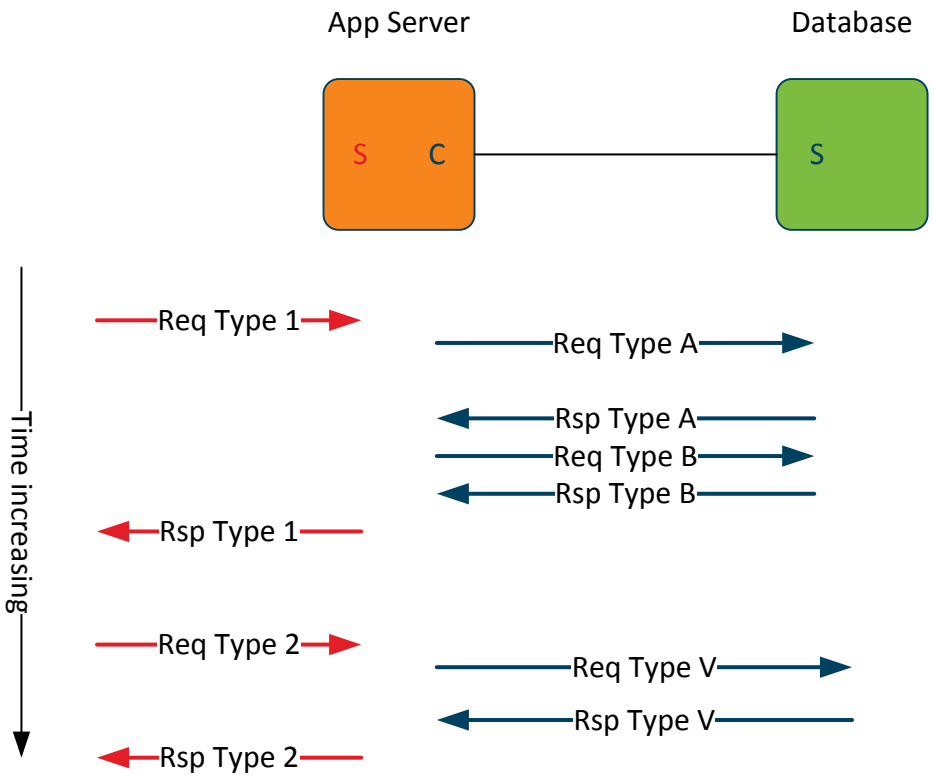
If we have a video recording of the problem occurring we can use data submitted by the user and data rendered on the user's screen when the transaction completes. However, if we don't have a video recording it can be difficult to get information about what precisely was entered by the user. In this case we must rely on the data return to the user; this is why a screenshot is so powerful in diagnosing a problem.

Applications that are presented to the user via a web browser are perhaps the simplest to study. Remember to consider that data from the browser to the first server could be via a GET query string, POST data or cookie data.

When using an Office application like Word, be sure to record information such as the name and path of the file being worked on, so that it can then be found easily in the network trace (in Wireshark simply use 'search string').

Multi-tier Characterisation

By repeating the failing or slow transaction (even if it doesn't fail or go slow) we can capture a baseline example showing the interactions at all of the tiers. This characterisation must be done during a maintenance period where the transaction is the only one executed on the system at that time.



In the example above we conduct a controlled test that shows that when a Type 1 transaction hits the app server, we see that the code on this server generates two database transactions; Types A and B. When a Type 2 transaction is processed by the application server, a single Type V database transaction is needed. We need to be conscious of the fact that the mapping of transactions in this way may be affected by application transaction parameter values, and so it's a good idea to work with the developer (if possible) and test a range of parameter values.

Gathering the baseline information in a test environment is not a good idea since it's quite likely that it will look different to a sample taken in production, even though the differences may be subtle. Ideally we need to capture the baseline using the same input data, same userid and same user PC as was used when the problem occurred.