Duration	Client	Situation and Outcome
5 days	Airline	New automated bag-drop units (ABDs) were spending 7 seconds (and often much longer) keeping the bag stationary for unexplained reasons. There were three vendors involved, each with their own software running inside the ABD. Initial packet captures taken at the local switch provided data for a single ABD and showed three distinct phases during the time in question. Some “log” traffic generated by one of the vendors was used to identify these phases. The middle phase (of 3.5 seconds but often longer) was easily identified as a single HTTP POST to an application server located in an airline data centre. Each vendor was then asked to explain what their code was doing during the other two phases. Given this narrower time to investigate, they were able to find inefficiencies that could be improved. Packet captures taken at the web/application server showed that 98% of the duration of the ABD’s HTTP POST was due to 13 back-end API calls to another server in the USA - which used HTTPS. The round trip time for each of these was more than 210 ms. Further analysis found that 6 round trips were due to a HTTP “feature” that happens to be enabled by default in Microsoft’s .Net. There were, in fact, only 7 real API calls. Simply adding a line in a configuration file eliminated these 6 wasted round trips, saving 1.2 seconds for every bag drop. Further, it eliminated 7 or 8 round trips (1.4 to 1.7 seconds) from other kiosk and human operated check-in functions too. The variability of the timings of the POSTs was due to packet losses in the network between Australia and the USA.
6 days	Large Service Provider	Performance of file transfers from Sydney to China, over an Internet VPN, was very poor. Analysis of packet captures taken from both ends showed that the IPsec packets were going out-of-order enough to trigger the TCP “Congestion Avoidance” mechanism – which resulted in very slow data transfers. It was thought that “The Internet” must be the problem, but over an elapsed period of a few weeks of questioning the network people and getting captures in different locations along the path, it was eventually found that the OOO packets were created in the internal Sydney network by an overloaded IPS device. That device was configured to not “inspect” this particular IPsec traffic flow, resulting in performance moving into line with that expected – given the 50 Mbps bandwidth limit at the China end.
2 days	University	Performance load testing of a newly purchased web application produced “failed” transactions at the point of full load. The error message was “Bad Request”. Analysis identified that the load generator was issuing good requests but the application (in an Apache Tomcat server) was not correctly handling database connection pooling. The failure occurred only when the number of concurrent TCP/IP connections reached the supposed maximum. The fault lay squarely with the application developer and no faults were found in any other components (load balancers, network, etc.) A summary report and supporting artefacts were created to present to the vendor, including details of ~300,000 transactions, database connections, exact point of failure, transaction content details, etc.)

Duration	Client	Situation and Outcome
5 days	State Government	A new application performed very badly when it was tested under the assumed maximum load. Packet captures were taken on a user PC while various application functions were exercised. Simultaneously, packet captures were taken on the Oracle database server and on both sides of an intermediate Check Point firewall. This was done for both no-load and heavy-load situations. Measurements of the packet transit times between each of the capture points showed that the performance degradation was entirely due to the firewall. Transit times through the firewall increased from the no-load sub-millisecond to an average 30 ms and maximum 80 ms (each way) when under load. The firewall was upgraded in order to cope with the expected loads. Also, a little known Check Point and Oracle performance issue was discovered and remediated.
4 days	Federal Government	Performance of reports from a SAS system (with Teradata backend database) was extremely variable. Packet captures were taken on a user PC while various reports were run. The analysis revealed that the SAS system was retrieving large numbers of rows of data (up to 200,000) from the backend database – but was doing it just one row at a time. The analysis also provided a clean bill of health for the WAN (which was initially strongly suspected by some of the support people).
4 days	Airline	Performance of web browsing for international users at two sites was very poor. Stopwatch measurements taken when a PC used the official proxy servers (versus going directly to the Internet without a proxy) showed a significant performance difference. Naturally, the initial thoughts were that the proxy servers (provided by an external vendor) were at fault. The support staff were proposing to change proxy vendors. However, analysis of packet captures taken on the user’s PC during the test runs showed that the degradation was actually caused by the way the browser handled TCP/IP connections to the proxies. Changing proxies would not have achieved any improvement, unless changed to a different “style” such as transparent proxies.
7 days	State Government	A newly purchased application was not performing as well as desired or expected. Packet captures were taken on a user PC while various application functions were exercised. The exact transaction/network/server behaviours for each function were measured and plotted. The major cause of poor performance was the way the application accessed the Oracle database. Data was requested in very small chunks, resulting in tens of thousands of unnecessary network round-trips. Spreadsheets listing the exact SQL statements involved with each function (each with the exact breakdown of client/network/server timings) was provided to the vendor. They were able to modify their application code to dramatically improve efficiency, significantly improving performance.

Duration	Client	Situation and Outcome
1 day	State Government	A newly purchased application was still not performing as well as expected. It was found that 75% of the time taken for each user function was actually inside the user PC (that is, related to the internal PC application). Spreadsheets and charts plotting the exact network activities involved with each function (with the exact breakdown of client/network/server timings per application transaction) were provided to the vendor. Any improvements would require application code modifications. The network and server infrastructure were found "not guilty" of any performance problems.
9 days	Large Service Provider	Credit card transactions flowing via MQ between mainframes at a major bank and a service provider were experiencing frequent delays of up to a second. Although only 1% of transactions were affected, the total delays added up to 12% (7 minutes per hour of no activity). The provider staff and their customer had spent months trying to identify the source of the problem – with no success. Packet captures taken with NetScouts at various points in the network path revealed that something on the customer's internal network was delaying packets by 10ms. Due to a combination of circumstances, including out-of-order packets, this was causing packet drops and large retransmission delays. Several possible fixes and workarounds were recommended. A change to the mainframe TCP settings removed the gaps, improved performance – but more importantly, gave the bank more confidence in the system moving into the heavy Christmas period.
3 days	Telco	Sydney users of an internal SAP application (with server in Asia) experienced occasional failed logons. This did not happen to users outside Australia. The problem had existed for several weeks. Analysis of packet captures taken at a user PC in Sydney revealed that a WAN accelerator at the Sydney end of the international WAN was improperly responding to HTTP Authentication requests. Three main alternatives were recommended.
3 days	Network Provider	User logons to a new application deployed to users were normally 10-15 seconds but were often taking over 2 minutes (and regularly failing). Three different vendors were involved in providing the application, server and associated network components and were all "pointing the finger" at each other. Packet captures in 3 locations (user PC, vendor hosted server in different city and WAN provider router) revealed that the "trigger" event was occasional packet losses in the customer's own network (which was easily fixed). Arguably, the real cause was the low bandwidth WAN link, coupled with "bad" behaviour in the customer's own Cisco ASA firewall as well as behaviour in the application vendor's F5 load balancer. None of these behaviours were "bad" when taken individually – but became so when combined.

Duration	Client	Situation and Outcome
3 days	Large Service Provider	The provider operated the outgoing Internet proxy environment for a large bank. The bank wanted to implement “per user” authentication mechanism and installed a Bluecoat “AAA” server coupled to a dedicated AD server. However, whenever the AAA function was enabled, Internet web-pages would take up to 30 seconds to load. The analysis of 11 GB of traffic revealed that the AAA-to-AD connection was only handling 2 transactions at once, even though many thousands of requests were arriving from the proxies every second. This resulted in an internal AAA queue length of over 30 seconds. This hinted at a process/thread limit in either the AAA or AD server – and it was subsequently found that the AAA configuration file “Bluecoat.ini” contained the setting “MaxThreads = 2”. Changing this to “5” put performance back to normal This saved the unnecessary $250,000 expense that was budgeted for a bigger/faster proxy server farm.
Past 8 years	Various	The general situation is a “tricky”, intermittent problem that numerous vendors and teams are involved in (and none can find any problem in their own area). The problem drags on for weeks or months until eventually I’m called in to identify the exact problem behaviours within days or sometimes just hours.
Example Reports		LinkedIn Posts / Blogs: https://www.linkedin.com/pulse/sometimes-we-need-packet-captures-more-than-two-locations-storey https://www.linkedin.com/pulse/my-solution-sharkfest-2015-megalodon-challenge-philip-storey YouTube Channel: https://www.youtube.com/c/NetworkDetective