Evaluation of Protocol Treatment in 3G Networks

Patrik Arlos, Ravichandra Kommalapati, Markus Fiedler

IEEE Local Computer Networks

2011 Bonn

Evaluation of Protocol Treatment in 3G Networks

Patrik Arlos, Ravichandra Kommalapati, Markus Fiedler
School of Computing
Blekinge Institute of Technology
Sweden
Email: (Patrik.Arlos, Ravichandra.Kommalapati, Markus.Fiedler)@bth.se

***Abstract*—In this work, we present a systematic study of how the traffic of UDP, TCP and ICMP protocols is treated, in three operational Swedish 3G networks. This is done by studying the impact that protocol and packet size have on the one-way-delay (OWD) across the networks. We do this using a special method that allows us to calculate the exact OWD, without having to face the usual clock synchronization problems that are normally associated with OWD calculations.**

From our results we see that all three protocols are treated in the same way by all three operators, when we consider packet sizes that are smaller than 250 bytes and larger than 1200 bytes. We also show that larger packet sizes are given preferential treatment, with both smaller median OWD as well as a smaller standard deviation. It is also clear that, ICMP is given a better performance compared to TCP and UDP.

I. Introduction

The number of mobile broadband and computing users has exploded after the release of the Apple iPhone. Together with this, a vast amount of application developers have started to develop applications that will run on mobile devices, that in turn connect to Internet either via Wi-Fi or 3G networks. When creating network intense applications, i.e. streaming applications, that delivers a good Quality of Experience to the user, the application developer benefits if the network behavior is known. The operators are interested in providing a good user experience, i.e. Quality of Experience. In a limited or congested scenario, they can adapt their service to the users, by treating protocols differently. I.e. as many users perceive networks via browsers, giving TCP a preferential treatment over UDP, may retain the user experience at a high or acceptable level, long enough to cope with the temporary lack of resources. Similarly, operators are well aware that users use ping (ICMP) to measure RTT performance, hence, if they treat ICMP fairly, they may be ranked highly in comparisons to other operators.

In this work, we present a systematic study of how the traffic of UDP, TCP and ICMP protocols is treated, in three operational Swedish 3G networks. This is done by studying the impact that protocol and packet size have on the one-way-delay (OWD) across the networks. We do this using a method that allows us to calculate the exact OWD, without having to face the usual clock synchronization problems that are normally associated with OWD calculations. We use the OWD as comparison metric, as most other metrics (bit rate, throughput, jitter, etc.) are directly or indirectly related to the OWD that individual packet experience. The other metric that is of importance, packet loss, have not been that visible, we've only seen a few random losses.

In [1] the authors investigated how ICMP is treated via the RTT and OWD for both up- and down-link, they did this using random inter-departure times and random packet sizes. In [2] the authors studied OWD performance for UDP when using VoIP traffic and TCP performance via Goodput (Content Size divided by download time) measurements. The difference to the work presented here is that of the measurement method, and used traffic type. Here we use a constant bit rate source. Furthermore, we extend the work in [3], [4] (where we only studied UDP performance) by including TCP and ICMP.

From our results we see that all three protocols are treated in the same way by all three operators, when we consider packet sizes that are smaller than 250 bytes and larger than 1200 bytes. We also show that larger packet sizes are given preferential treatment, with both smaller median OWD as well as a smaller standard deviation. It is also clear that, between 250 and 1200 bytes, ICMP is given a better performance.

The outline of this paper is as follows; in Section II we describe the method; this is then followed by a description of the experimental setup in Section III. In Section IV we present the results and a discussion of these, and Section V contains our conclusions.

II. Method

Our method relies on having access to IP packet at both sender and receiver, based on which we can calculate the one-way delay (OWD). The problem when calculating the OWD is how to handle the clock synchronization. The OWD is, as such, simple to calculate. The OWD of the i^{th} packet, d_i, is calculated as:

$$d_i = T_{\mathrm{b},i} - T_{\mathrm{a},i} \tag{1}$$

where $T_{\mathrm{a},i}$ is the arrival time of the i^{th} packet at location a; correspondingly $T_{\mathrm{b},i}$ is the arrival time of the same packet at location b. In the general case, the time stamps $T_{x,i}$ are obtained from two different clocks. To get an unbiased OWD estimate, these clocks should be synchronized. In [5] the authors investigate the three main synchronization methods NTP, GPS and IEEE1588 used for OWD measurements. When the Network Time Protocol (NTP) [6] is used, the clocks can be synchronized within [10 ms, 20 ms] for WAN scenarios, and around 1 ms in LAN scenarios. If the synchronization needs to be better, then a GPS solution is needed. Together

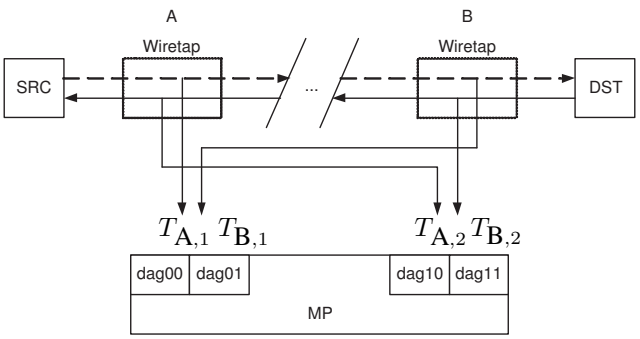

Fig. 1. Wiring method

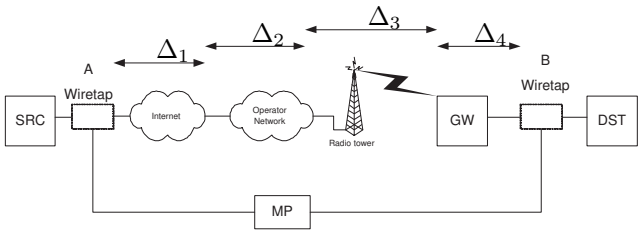

Fig. 2. Setup used in experiments.

with NTP, this allows for a synchronization in the order of 1 μs. The current state of the art is to use Endace [7] DAG cards together with a GPS, then the theoretical synchronization is in the range of [20 ns, 60 ns], depending on type of measurement card. However, according to our own experience, this is difficult to obtain in practice, as we still have two independent clocks. In [8] the author described the internal functioning of the time-keeping in side of the DAG cards, and in [9] the authors describe a method to synchronize clocks across the Internet. Regardless of what method or technique used for synchronization, the OWD estimations can at the worst be twice that of the synchronization level [10].

Our method uses wiretaps and a special wiring in conjunction with DAG cards to obtain the time stamps from the same clock. In Figure 1 a schematic of the wiring is shown. When a packet is sent from SRC to DST it will travel across the upper wire (dashed line). As it passes the first wiretap A, a copy of the packet is made and it is sent to the interface dag00, where it arrives at time $T_{\mathrm{A},1}$. At the same time the original packet makes its way across the network and eventually reaches wiretap B. Here, a copy is sent to interface dag01, where it is received at $T_{\mathrm{B},1}$. Similarly, if a packet is sent from DST to SRC the packets are duplicated by the wiretaps and made available to the dag1x interfaces. The main drawback with this wiring is that we require close proximity between SRC and DST. The actual distance is determined by the technology that carries the traffic from the wiretap to the DAG cards.

Let t_0 be the time when the packet actually passes wiretap A, and t_1 when it passes wiretap B. Then $T_{\mathrm{A},1} = t_0 + L_a/P_s$, where L_a is the cable length between wiretap A and dag0, and P_s is the propagation speed in that cable. Similarly, we define $T_{\mathrm{B},1} = t_1 + L_b/P_s = t_0 + L/P_s + L_b/P_s$, where L_b is the cable distance from wiretap B to dag0, and L is the cable distance between wiretap A and B. The OWD is then obtained as $\Delta = T_{\mathrm{B},1} - T_{\mathrm{A},1} = L/P_s + \frac{L_b - L_a}{P_s}$. So if we select $L_b = L_a$ we cancel the second factor and obtain the desired OWD between the wiretaps.

The method was applied and verified in [3], where it was showed that it could detect change in the transmission time in multiples of 60 ns (using DAG3.5E cards), hence the method is more than accurate enough for the measurements that were conducted in the context of this study.

III. Setup

To evaluate the mobile networks, we used the setup shown in Figure 2. Here SRC is sending traffic to DST. This is done via the Gateway (GW) that uses a Huawei E220 USB modem to connect to the mobile network. In-between the SRC and Internet, we placed wiretap A. The other wiretap B is placed just in front of DST. The SRC and DST (both are AMD 64 X2 5000-2.600 MHz with Linux 2.6 kernels) are connected with 10 Mbps full-duplex Ethernet cards (BroadCom). The GW is Intel Core 2 Duo 32-bit with 4 Gbytes of RAM (Windows Vista Business). The GW was configured for Internet sharing of the mobile network and no firewall was active, but we added port forwarding so that the traffic is forwarded to DST. The DST computer connected directly to the built-in Ethernet card (Broadcom) of the GW. The wiretaps feed into a Distributed Passive Measurement Infrastructure [11] enabled Measurement Point (MP) that stored the packet trace to file. Furthermore, the DAG cards were synchronized using both NTP and GPS.

A. Traffic Generation

To generate data we used three different C++ programs, one for each protocol (UDP, TCP and ICMP), which allow us to control packet sending rate and size. Furthermore, these programs use an application layer header that contains a sequence number. The sequence number starts at zero and is incremented by one for each transmitted message. This allows us to uniquely identify each packet, thus avoiding any ambiguities associated with hashing. Each experiment consisted of 100 packets with a payload size of K bytes, sent one packet per second. Then the payload size is changed, and a new experiment was started around 30 seconds after the previous completed. We start with a 20 byte payload, and increase with 10 bytes until we reach 300 bytes. After this we increment the payload size with 20 bytes, until we reach 1440 bytes payload size, i.e. IP packet size will be 28 bytes larger for UDP and ICMP and 52 bytes (as options were used) for TCP. However, when we investigated the traces, we detected that our ICMP generator actually added one extra byte to the payload, thus making the IP packet 29 bytes larger than the payload size. In total we will evaluate 86 different payload sizes, this will take just above three hours to complete.

To avoid time-of-day patterns, we have tried to execute all experiments related to an operator, protocol and direction at

the same time of day. However, one operator (B) disconnected sessions at midnight, so we intentionally avoided executing the tests during this time. Furthermore, some experiments were repeated due to measurement problems, session termination, etc.

B. Delay Calculation

As we are in control of both sender and receiver, we can easily identify both sending and receiving IP address as well as transport port numbers. We then use the application header for the packet identification. Once identified, we can calculate the OWD. Using the same notation as before, the delay would be calculated as defined in Equation 1. However, due to numerical issues [10], this is not recommended. It is better to use the following equation:

$$d_i = \widetilde{T_{\mathrm{a},i}} - \widetilde{T_{\mathrm{b},i}} \qquad \widetilde{T_{x,i}} = T_{x,i} - \lfloor T_1 \rfloor, \tag{2}$$

where $T_{x,i}$ is the arrival time of the i^{th} packet at location x; and T_1 is the arrival time of the first packet caught in this particular experiment, i.e. $T_1 = \min(T_{\mathrm{a},j}, T_{\mathrm{b},j})$ for $\forall j$. This will avoid having the time stamps truncated by the precision of the analysis tool [10].

C. Delay Components

The OWD that we will calculate has four contributors, see Figure 2. Δ_1 is the delay contribution by the Internet, Δ_2 that of the core network of the operator, Δ_3 that of the radio network, and the last contribution Δ_4 comes from the GW. Out of these four, we cannot measure or estimate Δ_2 and Δ_3 alone, as this means entering the domain of the operator.

We can estimate Δ_1 by using ping and traceroute to the operator's Internet exchange. From our vantage point on the Internet, the operators are between five or six hops away, and between us and them we have the Swedish University Network with optical multi-gigabit links. Hence the impact of this will not be negligible, but it will be quite stable, the average RTT between DST and the operator Internet exchange is 15 ms for all three operators. Hence, as the links are symmetrical, the OWD contribution will be around 7.5 ms. Furthermore, we can ignore the packet size as the links have such high capacity that the serialization delay is negligible.

In order to quantify Δ_4, we designed a experiment, where we replaced the E220 with a D-link DUB-E100 FastEthernet USB adaptor that allows us to connect directly to the wiretap at the source through a CAT5E cable. As we are using a USB NIC, the packets travelling across this NIC will receive the same treatment as those that are sent across the modem.

Figure 3 and 4 shows the minimum, maximum and mean values of the OWD for both up- and downlink for the GW. From the graphs, the delay increases with increasing packet size as expected and it is observed that the highest delay is around 2.5 ms at payload size of 1440 bytes in both uplink and downlink scenarios. Our purpose is to show that the impact of the GW is limited, bounded and no significant difference between up- or downlink directions. Despite the low load, we did detect that some packets were lost when ICMP was used,

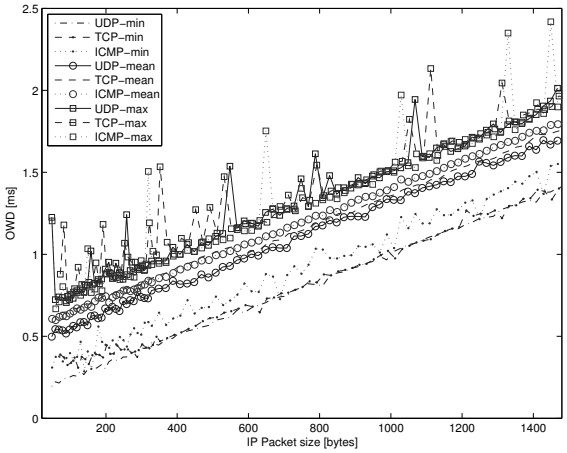

Fig. 3. OWD through GW for different packet sizes, uplink

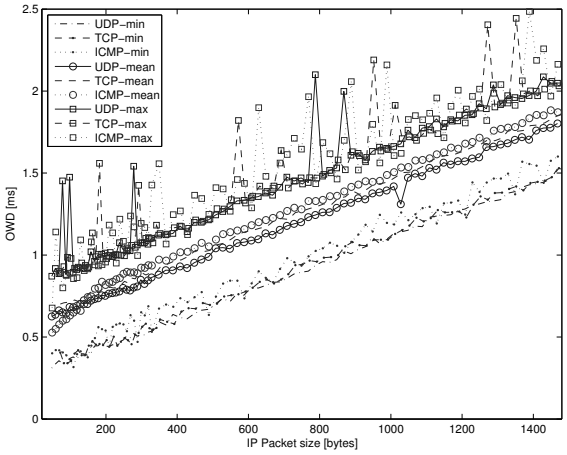

Fig. 4. OWD through GW for different packet sizes, downlink

while both UDP and TCP did not experience any losses. The losses that ICMP suffered occurred only in a few cases and in those it was only one packet that was lost.

In summary, the GW behaves linear in both uplink and downlink scenarios, delivering the packets to the other side within 2.5 ms, in both directions. This could quite easy be modeled, but comparing with the delay caused by the 3G network, the delay caused by the GW is negligible.

IV. Evaluation of Mobile Networks

We conducted our experiments on three major mobile operator networks in Sweden during the last week of December 2010 and during the first and second weeks of January 2011. In these networks, two of them (A and B) share the same radio access (RA), where as third (C) uses a different RA. We used normal business (flat-rate) mobile-broadband subscriptions; hence the operators would treat us as a regular customer. We base our discussion on the minimum, median and standard deviation of the OWD obtained from the experiments. The minimum OWD is interesting, as it is probably as close as we

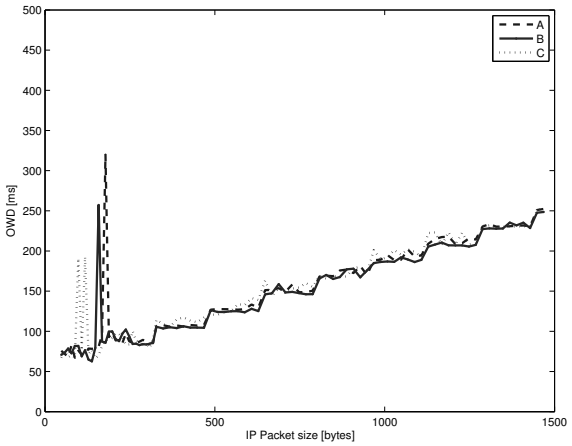

Fig. 5. Minimum OWD with UDP in uplink across different operators.

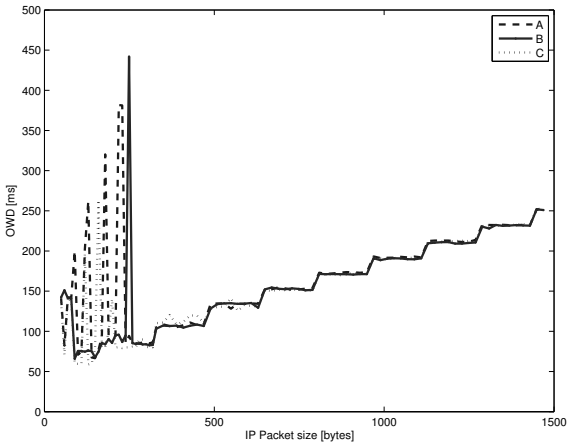

Fig. 7. Minimum OWD with ICMP in uplink across different operators.

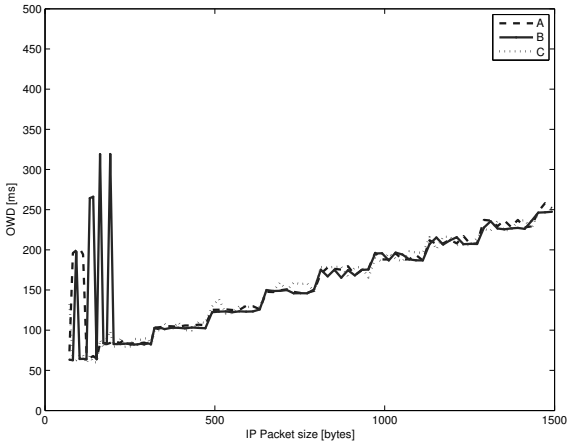

Fig. 6. Minimum OWD with TCP in uplink across different operators.

can get to the optimum treatment that a packet can receive. The median is interesting as it indicates where 50% of the packets experience a delay less or equal to it, while the other 50% experience a larger delay. By studying the standard deviation, we get an indication on how stable the service offered by the operator is.

A. OWD in the uplink

Figure 5 depicts the minimum OWD at different packet sizes with UDP transmission in operator A, B and C. It appears that in all three operators the minimum OWD is approximately the same and it increases with increase in packet size in a staircase pattern. It is also noticed that the steps are 160 bytes wide and the step height is approximately 20 ms in all operators. Similar observations were done by the authors of [3]. There, the authors indicate that the operators initially offered a WCDMA service (for the IP packet size below 252 bytes), and as the packet size increased beyond 252 bytes the operators started to alternate in between WCDMA and HSPA service, and finally changed to HSPA service. If we apply those results here, the operators have started to offer HSPA from the start, but on occasionally WCDMA is assigned.

Turning our attention to TCP, shown in Figure 6. At the initial glance, the figure has the same shape and pattern as that of UDP. We see a somewhat higher amount of WCDMA service for the smaller packet sizes. For a few of the larger packet sizes, operator C offers a WCDMA-like service. As this is seen in the minimum OWD, it means that for a period of 100 s the delay was never smaller than 362 ms for the 612 bytes payload and 533 ms for the 952 bytes case. We compared these results with data that we collected during the spring of 2010, and we did not find the corresponding large minimum OWD. Hence, these peaks are (with a high probability) the results of temporary radio and/or network conditions. Of course, the operator could have changed the network settings, but it's unlikely that they changed them to the worse.

For ICMP, the results are shown in Figure 7. Again, the shape and pattern seen for UDP and TCP is present here. For the smaller packet sizes, WCDMA service seems to be present, but as the packet size increase, HSPA is usually offered at least for a limited time.

B. Comparison of protocols in the uplink

As we have seen operator A and B are quite similar in their OWD behavior, so in the future analysis we will focus on operator A and C only. First we will study the minimum delays, the idea being that if the operator has some fundamental different treatment of the protocols, it will be visible here. The minimum OWD for operator A is shown in Figure 8 and for operator C in Figure 9. For operator A, the difference between the protocols is quite small, indicating that the operator do not treat the protocols differently. The same applies for operator C, but the values are alternating slightly more, causing the staircase pattern to be a bit unclear.

Now, we turn our attention to the median and standard deviation of the OWD. In Figure 10 we see the median OWD for operator A, this is accompanied by Figure 11 that holds the corresponding standard deviation. From these we observe that

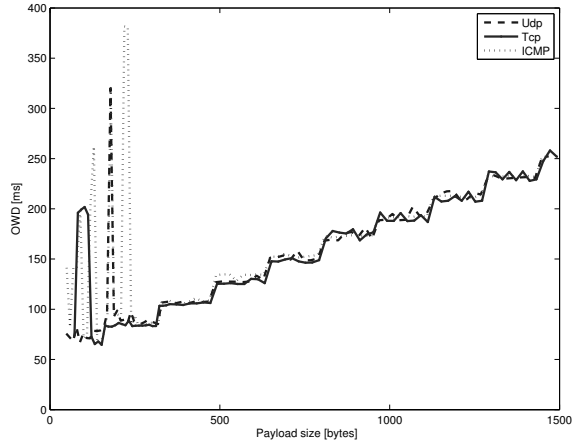

Fig. 8. Minimum OWD with UDP, TCP and ICMP in uplink in operator A.

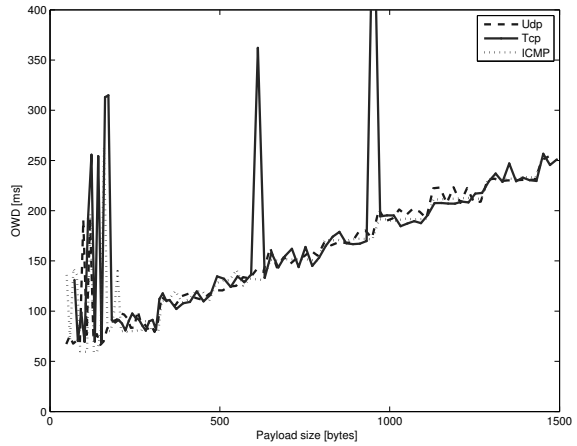

Fig. 9. Minimum OWD with UDP, TCP and ICMP in uplink in operator C.

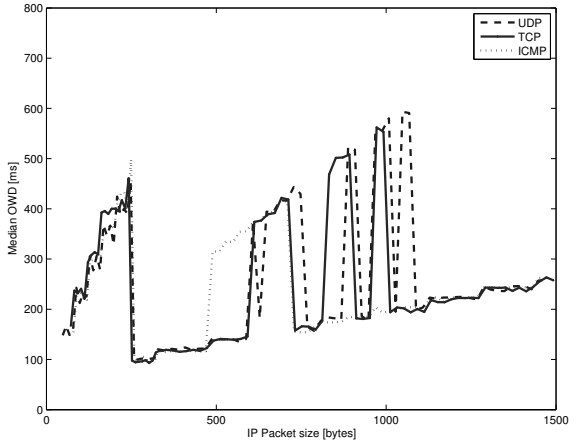

Fig. 10. Median OWD with UDP, TCP and ICMP in uplink in operator A.

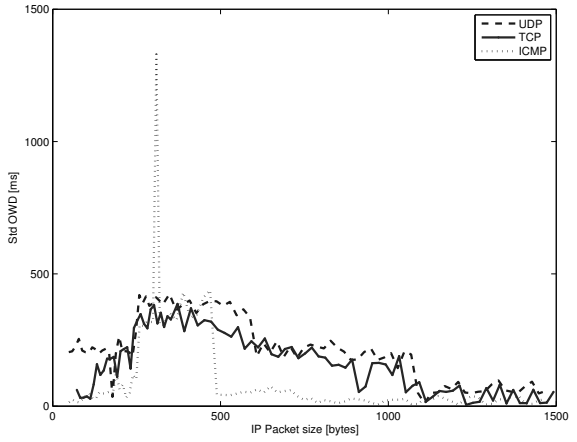

Fig. 11. Standard deviation of the OWD with UDP, TCP and ICMP in uplink in operator A.

operator A prefers to assign WCDMA for IP packets smaller than 250 bytes. Furthermore, we notice that in this region, ICMP has a quite small standard deviation, but it increases with the packet size. Similarly, the standard deviation increases for TCP, while for UDP it is quite large and remains flat.

After 250 bytes, we see (from the standard deviation) that there is a region of high variability, this coincides with that the median value has dropped down to around 100 ms. This high standard deviation indicates that the operator alters service type quite frequently for these packet sizes. The next region starts at 500 bytes. Here ICMP receives an intermediate service, where the performance is in between WCDMA and HSPA service (seen from the median value). Thanks to this new service, the operator reduces its alternation between service types, resulting in a smaller standard deviation, creating a more stable service.

At the same time, we see that UDP and TCP remained in HSPA, and the standard deviations decrease, for the remaining packet sizes. Furthermore, after 600 bytes, TCP and UDP start to alternate between WCDMA and HSPA service, and finally both stabilize in HSPA for packets larger than 1100 bytes. This happened already at 700 bytes for ICMP.

For operator C the same data is presented in Figure 12 and 13. Here the behaviour is quite different from operator A. For the major part of the packet sizes, operator C offers the intermediate service type, up to 700 bytes for ICMP, while TCP uses it until 1100 bytes, and UDP up to 1250 bytes. On the other hand, the standard deviation remains mostly below 200 ms for all protocols. With two exceptions; one for UDP at 308 bytes, and the other for ICMP at 729 bytes, where it coincides with the change to HSPA service. For TCP the standard deviation remains below the median, but it comes quite close to each other when the service changes to HSPA. To investigate why UDP had such an extreme standard deviation, we looked at that particular experiment, and we detected that there was a communication block (outage) that prevented any data to be delivered to the destination for around 6.5 seconds, this caused 7 packets to be delivered in a burst of 200 ms to the destination. We see a similar pattern in the ICMP case, but here the communication problem happens already on the first packet, causing an 8 second delay. After this, the OWD stabilizes around 160 ms, equivalent to HSPA service.

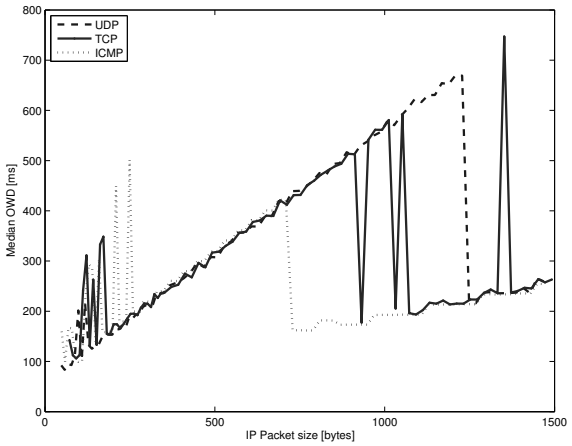

Fig. 12. Median OWD with UDP, TCP and ICMP in uplink in operator C.

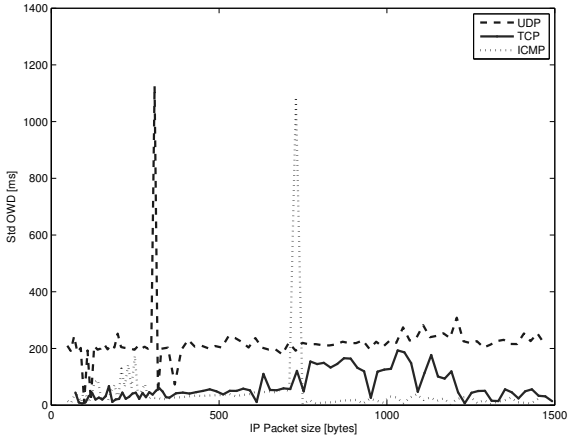

Fig. 13. Standard deviation of the OWD with UDP, TCP and ICMP in uplink in operator C.

C. OWD in the downlink

Figures 14, 15 and 16 show the minimum OWD for the different transport protocols. The minimum delay is quite small, less than 50 ms for all protocols and packet sizes, however, occasionally some large peaks are detected. We investigated these, and again compared them to the data collected during spring of 2010. In that data we did not detect peaks at the corresponding packet sizes, hence these are probably the results of temporary radio/network conditions.

D. Comparison of protocols in the downlink

In Figure 17 and 18 we show the minimum OWD values that are observed in operator A and C with all three protocols in download case.

From these graphs, it is observed that all operators offered to HSPA service beyond 500 bytes. However, for operator A, huge OWDs are observed occasionally at packet sizes beyond 500 bytes. From our previous experience, we consider them as outliers which are to be removed. Hence, the operators do not seem to give any preferential treatment to the protocols, when the network has no, or a light load.

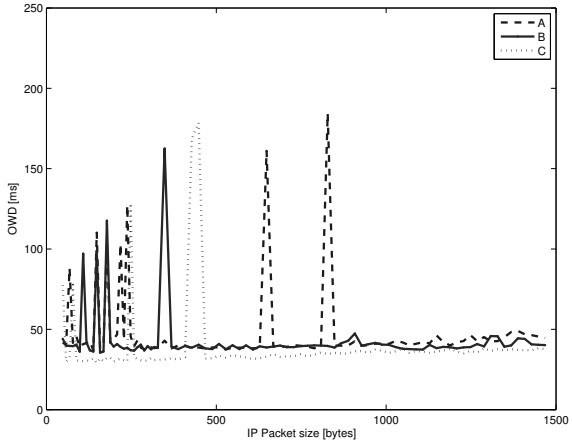

Fig. 14. Minimum OWD with UDP in downlink across different operators.

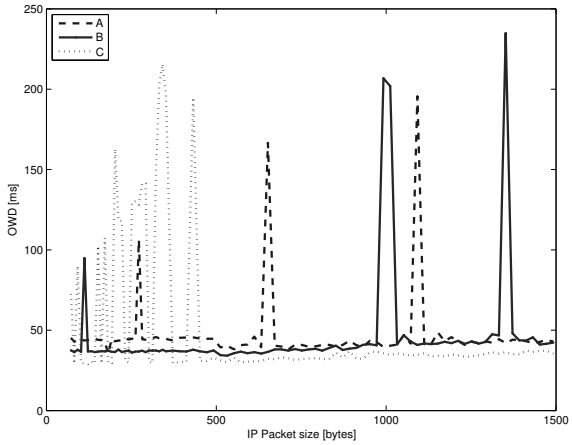

Fig. 15. Minimum OWD with TCP in downlink across different operators.

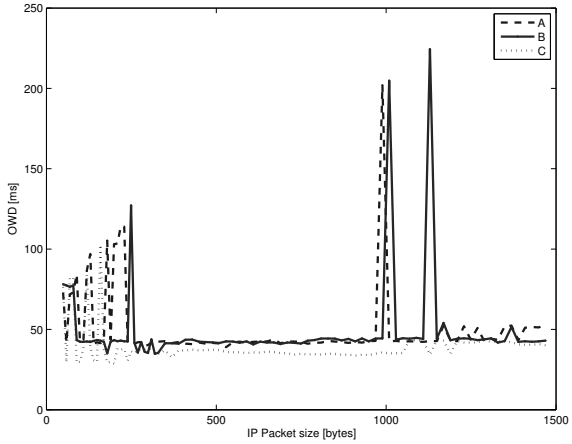

Fig. 16. Minimum OWD with ICMP in downlink across different operators.

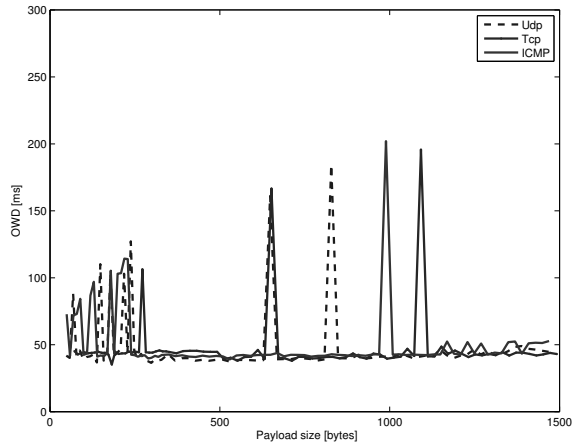

Fig. 17. Minimum OWD with UDP, TCP and ICMP in downlink in operator A.

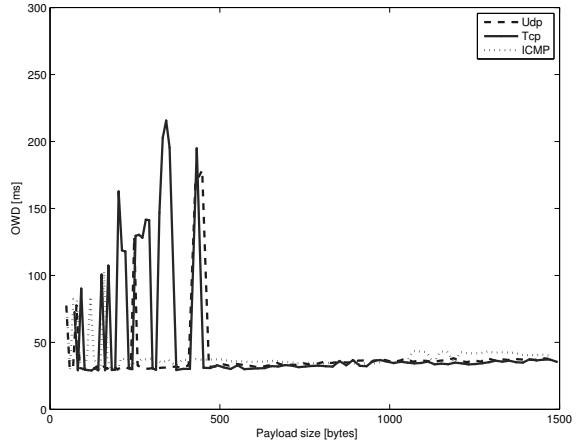

Fig. 18. Minimum OWD with UDP, TCP and ICMP in downlink in operator C.

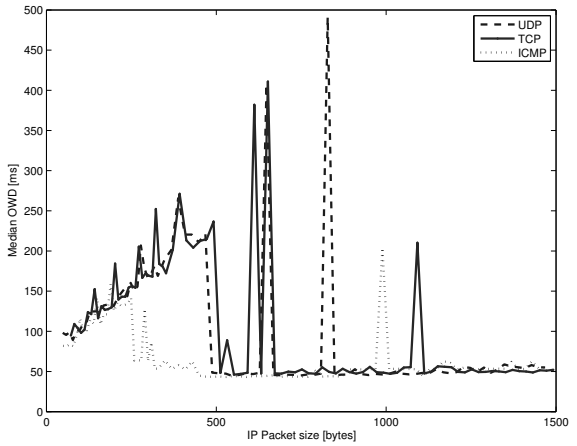

Fig. 19. Median OWD with UDP, TCP and ICMP in downlink in operator A.

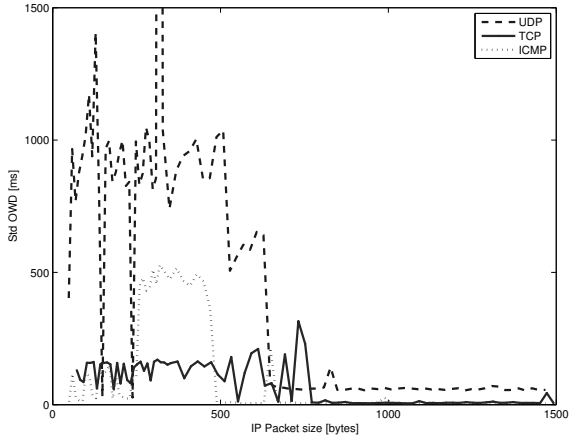

Fig. 20. Standard deviation of the OWD with UDP, TCP and ICMP in downlink in operator A.

When we look at the median values for operator A, shown in Figure 19, we see a clear distinction between the protocols, with ICMP receiving the better treatment, i.e. smaller OWD. Furthermore, we see that TCP and UDP behave in a quite similar way. Looking at the standard deviation, this is not the case. Again, ICMP receives a better treatment (overall lower standard deviation), with the exception between 250 and 500 bytes. Whereas for UDP and TCP, we observe extremely different standard deviation values up to 650 bytes. Especially in UDP the standard deviation is alternating around 900 ms. At 318 bytes, UDP experienced a massive delay of 31 seconds and loosing 24 packets, this results in a standard deviation of 10290 ms. After the 650 bytes threshold, the standard deviation for UDP stabilizes at approximately 60 ms, while for TCP is in the same range as ICMP, around 6 ms.

For operator C, the median values are shown in Figure 21 and the standard deviation in Figure 22. Here it is more difficult to distinguish the protocols. For packet sizes below 500 bytes, we occasionally see behaviors similar to WCDMA. Beyond 500 bytes, the values are more or less identical. For the standard deviation, we again see that UDP receives worse service, with values around 200 to 350 ms, as compared to the median that was below 300 ms for all protocols. Hence, with this standard deviation, any receive would have a serious problem to adapt to the data stream. We also see that ICMP receives the best treatment, with values around 6.5 ms, while TCP seems to alternate between 6.5 ms and 50 ms.

E. Maximum values

As we already mentioned, in some cases the maximum values have a substantial impact on the statistics. Generally the maximum values seems to be unaffected by protocol, direction and operator. The typical maximum values that have been observed are in the range of a 400 ms to 2000 ms. But, then from time to time, we detect delays exceeding 6 s, and the worst OWD we've detected was 32 s in a downlink, and 14 s in a uplink experiment.

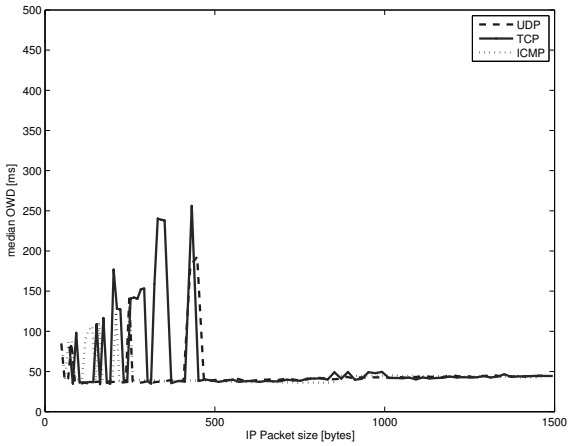

Fig. 21. Median OWD with UDP, TCP and ICMP in downlink in operator C.

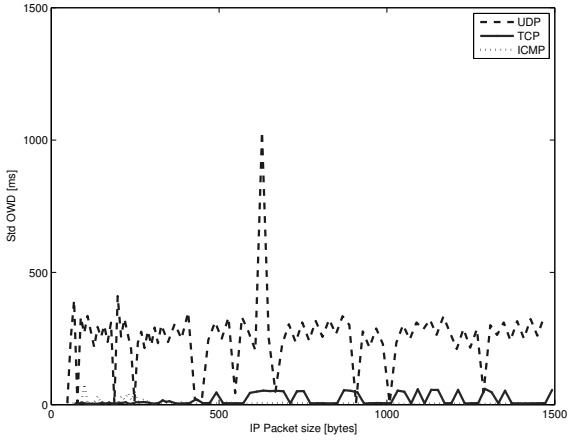

Fig. 22. Standard deviation of the OWD with UDP, TCP and ICMP in downlink in operator C.

V. Conclusions

In this paper we studied how the traffic of UDP, TCP and ICMP protocols are treated by three different 3G networks in Sweden. We based our study on OWD measurements through the networks, for both up and downlink. We used a constant bitrate traffic, where we sent one packet per second, for 100 packets at different payload sizes. We showed that the time-of-day does not have any significant impact on our data.

We found that all operators give preferential treatment to ICMP, with a much smaller standard deviation compared to TCP and UDP, this applies for both up and down-link. We also show that larger packets are given a better treatment. Here the operators have slightly different thresholds, but they are around 1100 to 1250 bytes in the uplink, and 800 bytes for downlink. For packet sizes below 250 bytes, all operators seem to more or less alternate between service types, this affects the standard deviation, but yet ICMP always receives a good treatment, and TCP somewhere in-between, and UDP having a rather large standard deviation, around 200 ms for the uplink, and 350 or 1000 ms for the downlink.

One interesting observation is that the threshold for ICMP seems to happen for all operators at the same packet size, around 700 bytes. The fact that ICMP seems to be treated better, cf. TCP and UDP, is quite opposite to previous findings for fixed networks. Hence, using ICMP for evaluation of network performance, might give you a value that is slightly too good as compared to reality.

For TCP, the two important packet sizes are 52 bytes in the uplink case and 1492 bytes in the downlink. The smaller size is equivalent to an acknowledgement packet, and the larger to a data packet. For these values, TCP receives in general a good service, with reasonably small median and standard deviation values. The operators seem to have prioritized these combinations, as to receive a fast treatment through the network. UDP on the other hand, for most of the packet sizes the standard deviation is quite large, but it decreases as the packet size increase.

References

[1] J. Fabini, W. Karner, L. Wallentin, and T. Baumgartner, "The illusion of being deterministic — application-level considerations on delay in 3G HSPA networks," in *NETWORKING '09: Proceedings of the 8th International IFIP-TC 6 Networking Conference*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 301–312.

[2] M. Jurvansuu, J. Prokkola, M. Hanski, and P. Perala, "HSDPA performance in live networks," in *Communications, 2007. ICC '07. IEEE International Conference on*, june 2007, pp. 467 –471.

[3] P. Arlos and M. Fiedler, "Influence of the packet size on the one-way delay in 3G networks," in *Passive and Active Measurement*, ser. Lecture Notes in Computer Science, A. Krishnamurthy and B. Plattner, Eds. Springer Berlin / Heidelberg, 2010, vol. 6032, pp. 61–70.

[4] ——, "Influence of the packet size on the one-way delay on the downlink in 3G networks," in *Proceedings of the 5th IEEE international conference on Wireless pervasive computing*, ser. ISWPC'10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 573–578.

[5] L. Tomaciello, L. De Vito, and S. Rapuano, "One-way delay measurement: State of art," in *Instrumentation and Measurement Technology Conference, 2006. IMTC 2006. Proceedings of the IEEE*, April 2006, pp. 218–223.

[6] D. Mills, "Network time protocol (version 3) specification, implementation," United States, 1992.

[7] Homepage. Endace.com. [Online]. Available: http://www.endace.com

[8] S. Donnelly, "High precision timeing in passive measurements of data networks," Ph.D. dissertation, The University of Waikato, 2002.

[9] D. Veitch, S. Babu, and A. Pàsztor, "Robust synchronization of software clocks across the internet," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2004, pp. 219–232.

[10] P. Arlos, "On the quality of computer network measurements," Ph.D. dissertation, Blekinge Institute of Technology, 2005.

[11] P. Arlos, M. Fiedler, and A. A. Nilsson, "A distributed passive measurement infrastructure," in *PAM2005*, ser. Lecture Notes in Computer Science, C. Dovrolis, Ed., vol. 3431. Springer Berlin / Heidelberg, 2005, pp. 215–227.

[12] K. Pentikousis, M. Palola, M. Jurvansuu, and P. Perl, "Active goodput measurements from a public 3G/UMTS network," in *In Communications Letters IEEE*, 2005, pp. 802–804.

[13] J. Prokkola, M. Hanski, M. Jurvansuu, and M. Immonen, "Measuring WCDMA and HSDPA delay characteristics with qosmet," in *Communications, 2007. ICC '07. IEEE International Conference on*, june 2007, pp. 492 –498.

[14] J. Prokkola, P. H. J. Perälä, M. Hanski, and E. Piri, "3G/HSPA performance in live networks from the end user perspective," in *Proceedings of the 2009 IEEE international conference on Communications*, ser. ICC'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 1152–1157.