Proceedings of the International Multiconference on
Computer Science and Information Technology pp. 837–843
ISBN 978-83-60810-14-9
ISSN 1896-7094

Network-level properties of modern anonymity systems

Marta Rybczyńska
Warsaw University of Technology

Email: mrybczyn@elka.pw.edu.pl

***Abstract*—This paper shows the network-level view of the behaviour of two popular and deployed anonymity systems; Tor and JAP (AN.ON). The analysis uses the fact that both of them depend on TCP (Transmission Control Protocol) and shows cases when network conditions may affect the systems' operations. The main topics are: on-off traffic, difference in available bandwidth between peers, and usage of the window mechanism in the TCP. The analysis is divided into two parts: from both the sender's and receiver's point of view. We present the results of experiments made on live systems, showing that the analysed effects may be encountered in practice.**

I. Introduction

THIS PAPER studies network-level effects on performance and protection given by modern anonymity systems. As the anonymity layer is built on top of the existing network protocols, the interactions between the two may affect the operations of the anonymity system.

We focus on the effects the TCP protocol has on the two most popular and already deployed designs; Tor and JAP. Although a significant amount of effort has been put into researching possible attacks against the protection mechanisms used by those systems, it is not clear which features of the network protocol allow the system to gain better protection, and which should be avoided.

It has been shown that attacks by traffic analysis against Tor are possible, when the attacker modifies the traffic from the sender's side [21]. We check if this works equally well against both Tor and JAP, and look into the limits of such attacks, introduced by the network conditions. We do not limit the analysis to the sender's side only, but also investigate the way the receiver can shape the traffic using the TCP flow control. The analysis and interpretation of results concentrate on the security-related issues, but we also look into possible performance implications when they are clear of issues.

The remainder of this paper is structured as follows: Section 2 reviews the related work and gives the background on Tor and JAP. In Section 3 we show our methodology and setup. The analysis and results are presented in Section 4. We conclude the paper in Section 5.

II. Background

Modern anonymity systems can be divided into two types. The first type introduces a noticeable delay, even up to days, in exchange for good anonymity of the user messages. Such systems are designed for services like e-mail. Two important examples are Mixminion [6] and Mixmaster [13]. The second group can be used for real-time or nearly real-time applications, like web browsing or remote access. Such systems are designed using client-server or peer-to-peer model. Important examples include Tor [7], JAP (AN.ON) [2], Freedom [3] (all three client-server), and the new I2P system [8] (peer-to-peer).

A. Tor

Tor is a low-delay, high-bandwidth anonymity system developed for TCP traffic, like web browsing [7]. It includes several extra features, like hidden servers.

The current implementation uses TCP connections between the nodes. It neither supports cover traffic, nor mixing. Congestion control and simple traffic shaping exist. The flows are routed independently through the network. However, multiple requests transmitted between the same two intermediate nodes may be multiplexed and form one connection.

When it comes to the network layer, Tor uses TCP/IP implementation provided by the host operating system. All the processing is done at the user level. From the networking point of view, Tor is a cascade of proxy servers with traffic aggregation.

B. JAP (AN.ON)

JAP (Java Anon Proxy) is an anonymity system developed at the University of Dresden [10] [11].

It uses a different approach than Tor. Instead of onion routing, it uses a modified version of Chaum mixes. JAP does not form one single network. Instead, users connect to so called cascades. In a cascade, mixes are connected by single connections only, which are used to transfer all the traffic. Additionally, proxy cache servers may be added after the last mix. In practice, JAP uses static cascades. All flows share the same path and are transmitted in a single TCP connection between subsequent nodes (mixes).

From the networking point of view, JAP is, just like Tor, a cascade of proxy servers also implemented as external applications.

C. Related work

The influence of traffic on the operations of the anonymity systems has been studied mainly in the context of traffic analysis attacks. That class of attacks uses flow properties to match flows before and after the anonymity system.

In the literature, a statistical approach is often used, as in [5]. Murdoch and Danezis have shown an attack against Tor that marks traffic flows by introducing delays, caused by flooding a node with traffic from an attacker–owned, corrupted node [14]. On the other hand, Wiangsripanawan et al. state that the attack would not work against systems using a different design, like that of Tarzan [19]. Zhu et al. have used a different approach and studied flow correlation attacks for a mix–based system [22]. They propose dummy traffic as a defence.

The problem of timing attacks have been studied in a number of papers [12] [16] [20], showing that numerous properties of the traffic can be observed at the other end of the anonymity system. Shmatikov et al. propose dummy traffic against such attacks [17]. It has also been shown that single flows can be recognised in the aggregated traffic [23]. Yu et al. introduce a flow marking technique where they mark traffic by the changing transfer rate, and detect the introduced pseudo-noise code later [21].

Research on the performance of current designs and reasons for that performance has been much less intensive. Wendolsky et al. measured the delay introduced by Tor and AN.ON (JAP), and report it to be in seconds, rather than milliseconds [18].

III. Setup and methodology

Our test environment consisted of two computers, controlled by us, and connected to the Internet. One of the machines had a Web server installed, the other one worked as client. The client also used Tor and JAP client software, which was used to connect to the anonymity systems.

During the test we downloaded several files of different sizes, from 40MB to 100KB, from our server to the client, using the two chosen anonymity systems during their normal operations. We did not control any of the nodes, nor interfere with the standard path selection algorithms.

We introduced changes to the traffic sent by the server and received by the client using bandwidth limits by utilising the standard Linux firewall tool iptables/netfilter with additional, custom scripts. We examined the direction from the sender using traffic shaped into bursts with lengths of 60 and 10 seconds, and with a constant limit during experiments performed on the receiver. There were two types of bursts, with the first at 128 kbit/sec and the second at 64 kbit/s. Between bursts, the transmission took place at 32 kbit/sec. Such rates were chosen after initial experiments, showing that JAP allows us to transfer at a maximum rate of 128 kbit/sec. All the traffic was captured for later analysis.

We used unmodified client software and only services available to the general public. It turned out that while huge a majority of Tor connections offered acceptable performance (there were, however, sporadic connection resets and stops of the data flow), JAP showed the performance needed for the experiments only on one cascade; Desden-Dresden. On the other cascades, unexpected connection drops and/or very low bandwidth prevented even short file transfers.

As we did not control the anonymity systems or the behaviour of other users, we could not achieve full stability of the conditions for our measurements. We addressed that issue at different levels. Our transfers were relatively long (from several seconds up to minutes), what allowed us to limit the impact of short disturbances, which were observed in the traffic. Low transfer rates allowed us to perform transfers without interfering with other limits there may be on the path, and without putting significant stress on the anonymity system nodes. Finally, we performed the experiments during the same hours and days of the week.

The later analysis was performed using the recorded traces. We processed the traces to get the number of packets in a one second timeframe. Then we calculated the cross-correlation between both ends. We used the following equation:

$$r(d) = \frac{\sum_i [(x(i) - mx) * (y(i-d) - my)]}{\sqrt{\sum_i (x(i) - mx)^2} \sqrt{\sum_i (y(i-d) - my)^2}} \quad (1)$$

where d is delay, r is cross–correlation, $x(i)$ and $y(i)$ are the signals, and $i = 0, 1, ..N-1$. mx and my are the mean values of the x and y signals, respectively. Cross-correlation is often used as a metric in anonymity system evaluation ([22], [12] or [21]), because it gives an attacker a relatively simple and efficient tool to match the sender with the receiver. We used that metric for the same reasons.

IV. Analysis and results

The analysis is based on TCP properties and on the properties of TCP connection cascades. Similar tests may be performed for any other TCP-based anonymity system.

Figure 1 shows one of our Tor transfers with no modifications from our side, in which a number of interesting facts arise. The first one is the disturbance at approx. 400 seconds from the transfer start, which is clearly visible in both the server and client traffic. Such disturbances can cause attacks to fail and were also observed in the other traces.

It is also worth noting that the number of packets on both pictures differs. The server sends at approx. 20 packets/sec, while the client receives at 40 packets/sec. That fact is easy to explain after realizing that the first picture shows plain a HTTP response, when the second shows the same response, but for packed and encrypted in Tor messages. That is why those pictures, and the ones shown later, should not be compared directly.

Figure 2 shows similar graphs for JAP traffic, with visible bursts on the server side (similar ones appear in all of our transfers) which looks like the expected results of the backward attack. The client traffic does not run at the maximum rate, however, as there are also bursts visible from that side.

A. Sender side

A number of different modifications to the traffic may be made on the sender side. For instance, the server may limit bandwidth for a single flow. There may also be bursts of traffic as the server switches from one flow to another. Additionally, there may be losses, which cause TCP retransmissions. We

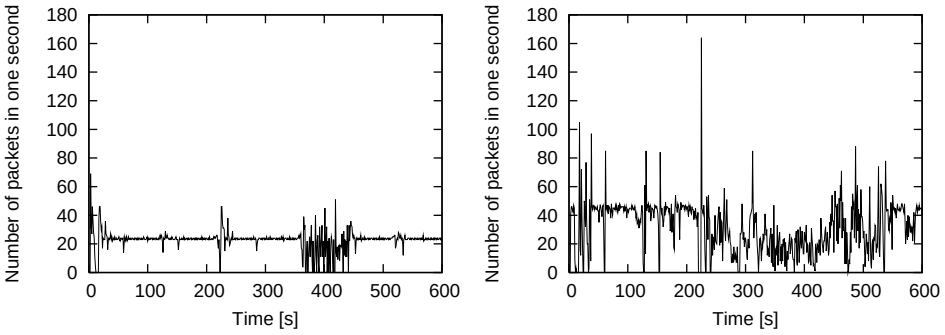

Fig. 1. Traffic flow sent from the server (left) and received by the client (right) during a transfer using Tor, without modifications to the traffic.

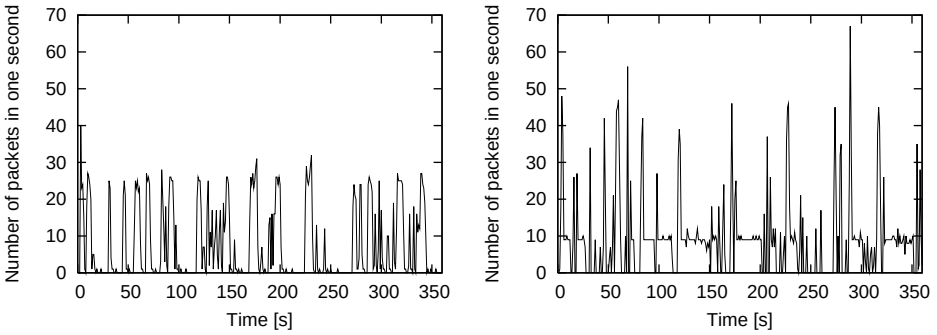

Fig. 2. Traffic flow sent from the server (left) and received by the client (right) during a transfer using JAP, without modifications to the traffic.

wanted to check if and how echos of such events would be visible on the receiver side.

Such echos are likely because of one of the basic features of the TCP protocol [9] [15], namely the fact that the data reaches the receiver application in the same order it was sent, even if it might have reached the node in a different order. It means that if one message is delayed, the data from the later ones would not reach the receiver before the delayed message is received correctly. Given the fact that the path used by the anonymity system consists of several TCP connections, the delays will cumulate.[1] That analysis is correct for both Tor and JAP, as they both use the TCP for transfers between the nodes and between the user and the nodes.

Such properties were already discussed in the context of traffic analysis attacks, as in [14] or [12]. We are not aware, however, of effectiveness tests under real-world conditions.

Figures 3 and 4 show modulated traffic before and after it is transferred, using Tor and JAP, respectively. In both cases the anonymity system did not change the delays and transfer rates to the extent that would make it unrecognisable.

In the tests we introduced a known traffic pattern into the anonymity systems and observed the resulting flow on the receiver side. In the first test, our server was transmitting at 128kbit/sec for 60 seconds, then at 32kbit/sec for 120 seconds, then at 64kbit/sec for 60 seconds and then, finally, at 32kbit/sec again for another 120 seconds. The relatively slow transmission rates were used to limit the possibility of reaching any other limits there might be on the path. The results are presented in Tab. I. Tor traffic obviously shows high correlation. In the case of JAP, the correlation is not that easily seen. However, after calculating cross-correlation between the flows using the equation (1), the values for the corresponding flows are the highest.

We have also calculated the length of the burst before and after the anonymity system. The results are presented in Table II. In both cases, bursts after the anonymity system are longer

[1]That is independent from packet reordering. The packets will be placed back in the correct order on every anonymity system node during TCP data reassembly, before passing them to the application (anonymity software, in this case).

TABLE I
CROSS–CORRELATION BETWEEN THE USER AND SERVER SIDE TRAFFIC.

System	Test	Best correlation
None	Normal transfer	0.97
Tor	Long bursts	0.82
	Short bursts	0.23
JAP	Long bursts	0.54
	Short bursts	0.59

TABLE II
CHANGE OF BURST LENGTH AFTER PASSING THROUGH ANONYMITY SYSTEMS (ORIGINAL LENGTH: 60 SAMPLES).

System	Burst length (before)		Burst length (after)	
	avg.	std. dev.	avg.	std. dev.
Tor	50.0	4.1	59.4	9.3
JAP	47.9	7.0	52.7	10.1

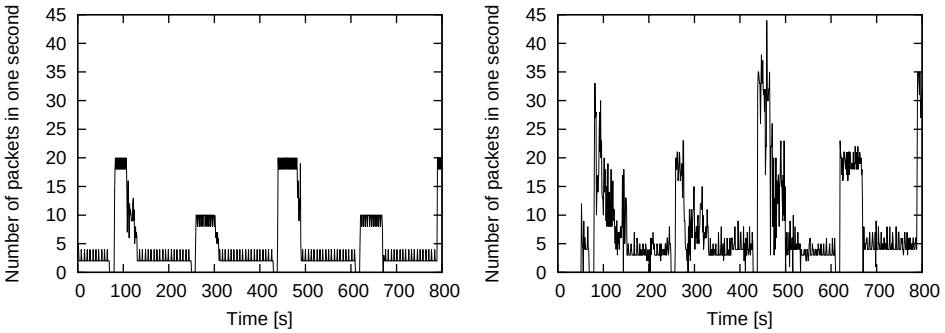

Fig. 3. Traffic flow sent from the server (left) and received by the client (right) during a transfer using Tor, sender–side traffic shaping.

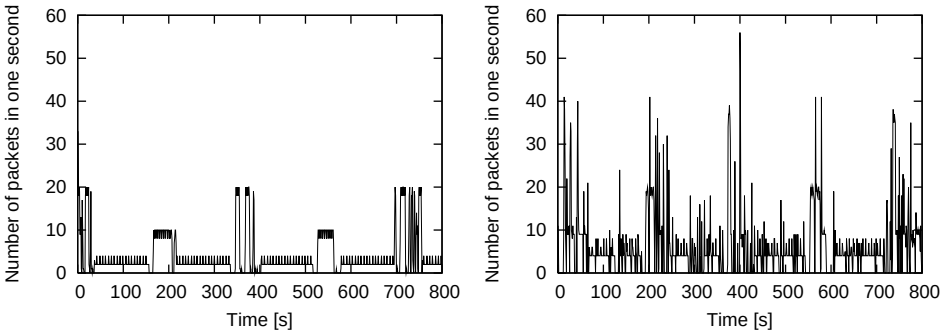

Fig. 4. Traffic flow sent from the server (left) and received by the client (right) during a transfer using JAP, sender–side traffic shaping.

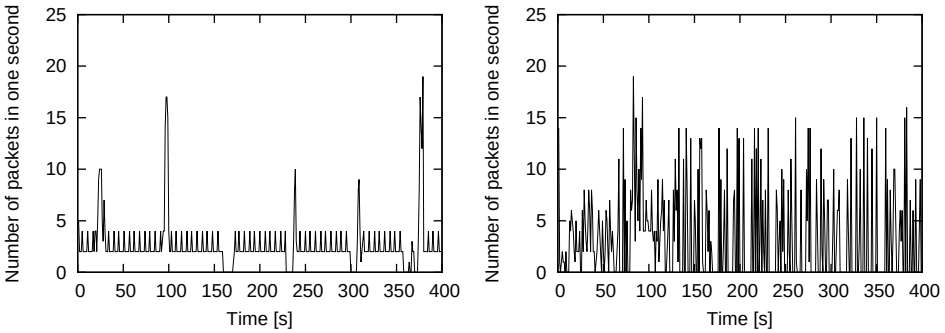

Fig. 5. Traffic flow sent from the server (left) and received by the client (right) during a transfer using Tor, server-side traffic shaping with short bursts.

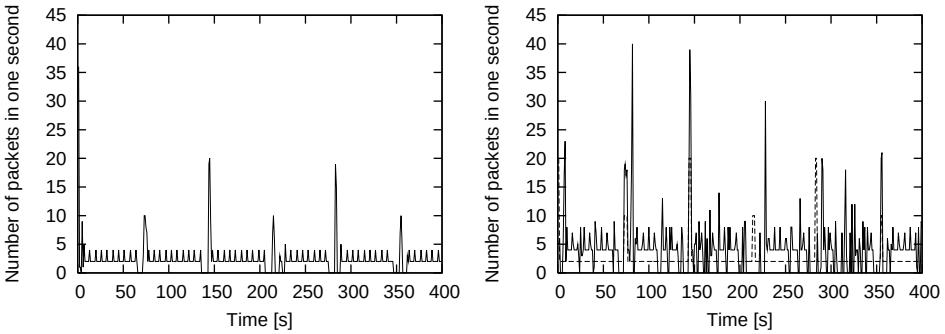

Fig. 6. Traffic flow sent from the server (left) and received by the client (right) during a transfer using JAP, server-side traffic shaping with short bursts.

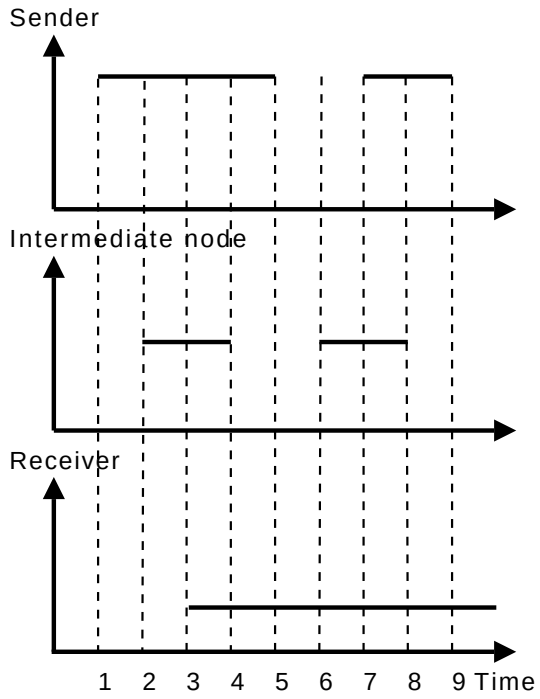

Fig. 7. Application-level view of the traffic in TCP-based anonymity systems as a result of flow control, in the case of differences in available bandwidth, from the sender (S) to the receiver (R) using the intermediate node (IM). Main events: (1) S starts transmitting to IM, (2) IM starts transmitting to R, (3) R starts getting data, (4) IM's transmit buffer full, stops receiving from S, (5) S' transmit buffer full, stops sending, (6) IM re-starts sending to R and receiving from S, (7) S re-starts sending to IM, (8) IM's transmit buffer full again, (9) S' transmit buffer full again.

than before, but the standard deviation also increases.

As the test was successful, we performed another one, using shorter, 10 second bursts. There were two types of bursts: at 128 and 64 kbit/sec. The traffic was then formed at the server, as shown in Figures 6 and 5. It can be noticed that the bursts are actually shorter than 10 seconds, which is an effect of the rate switching.

This time, the Tor traffic shown in Figure 5 did not show similarities to the original. That was also confirmed by low cross-correlation.

The JAP traffic shown in Figure 6, on the other hand, shows some similarities to the original. The cross-correlation test gave values at nearly the same levels as in the previous case of longer bursts.

B. Receiver side

Conditions on the receiver side can also influence the whole path because of the flow control mechanism in the TCP. The TCP has a window mechanism that allows notifications to the other node, that the transmission should be slowed down or stopped. A window is the number of octets that may be in the network without an acknowledgement. The current value is transmitted as a field in the TCP header [1] [4].

Dividing one TCP connection into multiple ones breaks that end-to-end flow control. The result of this may be seen if the sender transmits at a higher rate than the receiver can receive, and if the rate offered by the anonymity system is enough to handle the traffic without introducing noticeable additional delays or losses.

The situation is depicted in Figure 7. The transmission on the first (receiver) link will take place at the maximum rate (or close to that value). It will be lower, from the beginning, than the transmission rate of the server (events 1-3). That will cause the buffers in the nodes to fill.

When the sending buffer on the first node becomes full, it lowers the window size in the connection with the second node. After some time, the sending buffer on that node will become full. That will stop receiving from the next node (event 4). Finally, window lowering will occur at the last link (between the sender and last node), where it can be observed (event 5). With the maximum rates still at the same level, the transmission from the server will start forming bursts, as the transfers will occur at the maximum rate, but only for short periods of time, when the next node allows it (events 6-9).

Depending on the network configuration, delays, TCP implementation used by the nodes, and their settings, the bursts may be observed sooner or later. Also, the time until the window drops to zero for the first time will vary. In the worst case that includes sending and receiving TCP buffers on all the anonymity system nodes along the path and the buffers in the anonymity software. The bursts may appear earlier if the anonymity protocol has its own window mechanism, for instance.

Long and short bursts: We have checked if the effect described above can be observed in practice. Figures 8 and 9 show the results for Tor and JAP, respectively.

In the case of Tor there are bursts visible which are just as we expected (compare the with normal traffic from Figure 1). The time from the transfer start to the moment when the burst appears is much lower than expected, however. That was the reason for another test. The aim was to check more Tor paths and see how long a time was needed for the bursts to appear. We show the results later in this section.

JAP also shows bursts. The client receives at the maximum rate. The bursts do not differ sufficiently from the ones in Figure 2. Even the burst lengths are similar. A different pattern did not emerge even during the longest 12 minutes transfer.

The fact that the test did not work against JAP during our test time, is one of the issues worth noticing. The root cause remains a problem to be considered in the future work. The bursts may finally appear after all the buffers on the path are filled, which may take hours in the case of the slow transfer rates we used.

Our observations show that the transfer rates, even without any modifications, were lower for JAP than Tor. Also, Figure 6 shows no bursts on the server side, that may suggest an external bandwidth limit.

Filling the buffers of Tor: We performed additional experiments on Tor, as the amount of data needed to start bursts was much lower than expected. The results we got clearly show that the buffers on the whole path were not full when

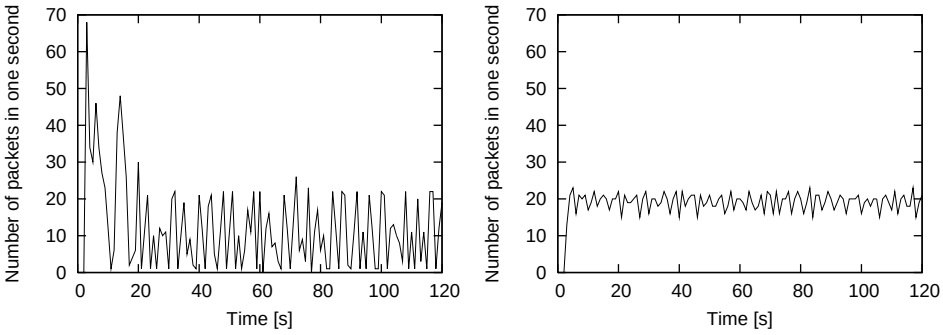

Fig. 8. Traffic flow sent from the server (left) and received by the client (right) during a transfer using Tor with client bandwidth limit at 96kbit/s.

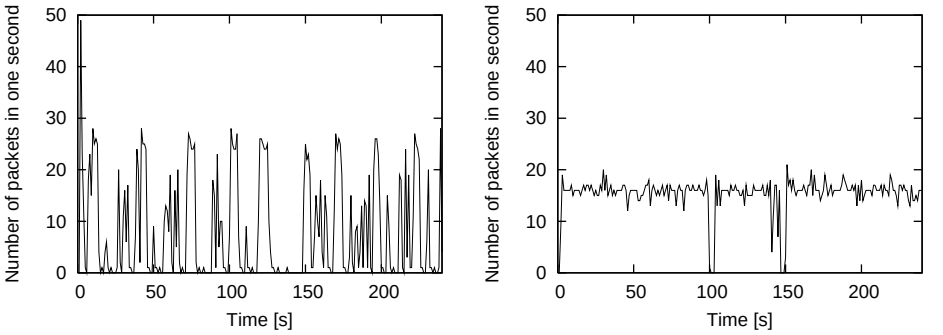

Fig. 9. Traffic flow sent from the the server (left) and received by the client (right) during a transfer using JAP with client bandwidth limit at 96kbit/s.

TABLE III
DELAY TO THE FIRST ZERO WINDOW MESSAGE.

Amount of bytes transferred	Number of occurrences	Percentage
less than 100,000	4	15.4%
100,000 to 200,000	2	7.7%
200,000 to 300,000	0	0.0%
300,000 to 400,000	3	11.5%
400,000 to 500,000	6	23.1%
500,000 to 600,000	2	7.7%
600,000 to max	2	7.7%
no zero window	7	27.0%
Total	26	100.0%

the traffic pattern started appearing. There should have been a similar test performed for JAP, but only one cascade seemed usable for our research.

Table III shows the results of tests performed using a 630 KB file against different paths through Tor. Every measurement was performed after a Tor restart, and resulted in a different path being used. We show the number of bytes transferred from the connection start to the first packet with zero window size (precisely: a window size lower than 1400 bytes).

We found that there are big differences between the paths. The situation of the window size dropping to zero may appear after less than 100,000 bytes have been transferred, but may not appear in the whole transfer. We looked into the traces in more detail and found a number of interesting facts. There were differences in the initial window. Also, the window size reached close to the beginning of the connection differs between the traces. A large window size appeared in the transmissions with a non-zero window, or in those with a zero window close to the transfer end.

That leads us to the conclusion that the differences are caused by the conditions on the path, including the configuration and operating system of the anonymity system nodes (especially the one closest to the server). It seems that the 'bursty' traffic is not caused by the buffers' overruns on the whole path, but rather as an effect of Tor's internal flow control.

It is also worth noting that if the zero window is reached, the bursts appear regularly for the rest of the transfer.

As the initial window size differs between the nodes, so does the amount of data that may be in transit simultaneously. That may lead to differences in response time, for applications that have such needs.

Artifacts: We observed a number of artifacts in the traffic. Examples include: oscillation at 220 sec in Figure 1a, disturbance between 350 and 450 sec in Figure 1 with its' echo in Figure 1, bursts on higher bandwidth transmissions in Figure 4 or oscillations in Figure 9.

While searching for possible explanations, we analysed the source code of the two tested systems. There was no direct answer. However, we have found out that, as data from many flows travel by a single TCP connection between the internal nodes, an retransmission or any other problem will affect performance on multiple streams at the same time. It means that the disturbances may be an effect of interactions with other flows passing through the anonymity systems.

V. Conclusions

We have shown a number of network-level effects that affect performance of currently deployed, TCP-based anonymity systems. We have also presented results showing that the effects may be visible in practice.

Our research shows that describing the network properties of an anonymity system by using only delay is not enough, and there are more factors to consider. We have also shown how the internal design affects flow characteristics. The results can be interpreted in two ways. The first shows possible performance issues, and the second one possible attacks (as the traffic may be marked from both sides of the connection).

We have shown that both systems do not change the traffic characteristics to the degree that would allow them to hide fast rate changes. Bursts of a length of 60 seconds were recognisable after passing by both systems. Additionally, that was also true for JAP and 10 second bursts. This limits the space for attacks that would watermark the traffic by bandwidth changes, like in [21]. Also, both systems tend to output longer bursts than the input bursts.

It should be noted that receiver behaviour (instead of only sender behaviour) should also be taken into account when modelling anonymity systems using network protocols with flow control like the TCP. When the end-to-end property is broken, characteristic flow bursts may appear on the sender side. How it may be used by an attacker remains a problem for future work to examine. It clearly affects performance, however, and introduces delay when a fast reply is needed. Adding to, or improving application level flow control in anonymity systems, could be a solution to this problem. Such mechanism should be designed to find a compromise between performance and the time of reaction to events from the other peers.

That may also mean that different settings or implementation differences of the network protocols could have a noticeable impact on the anonymity system's performance.

References

[1] Alleman, M., Paxson V., and Stevens, W, *TCP Congestion Control*, RFC 2581, April 1999.

[2] Berthold, O. Federrath, H., Köpsell, S, *Web MIXes: A system for anonymous and unobservable Internet access*, Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, 115–129, July 2000, Springer-Verlag, LNCS 2009.

[3] Boucher, P., Shostack, A., Goldberg, I., *Freedom Systems 2.0 Architecture*, white paper, Zero Knowledge Systems, Inc., December 2000.

[4] Clark, D., *Window Acknowledgement Strategy in TCP*, RFC 813, July 1982.

[5] Danezis, G. *Statistical Disclosure Attacks: Traffic Confirmation in Open Environments*, Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003), pp. 421–426, May 2003.

[6] Danezis, G., Dingledine, R., Mathewson, N., *Mixminion: Design of a Type III Anonymous Remailer Protocol*, Proceedings of the 2003 IEEE Symposium on Security and Privacy, May 2003.

[7] Dingledine, R., Mathewson, N., Syverson, P. *Tor: The Second-Generation Onion Router*, Proceedings of the 13th USENIX Security Symposium, August 2004.

[8] I2P site, URL: http://www.i2p.net.

[9] Jacobson, V., Braden, B., Borman, D., "TCP Extensions for High Performance", RFC 1323, May 1992.

[10] "Technischer Hintergrund von JAP", technical paper, URL: http://anon.inf.tu-dresden.de/develop/JAPTechBgPaper.pdf

[11] Köpsell, S., "AnonDienst—Design und Implementierung", technical paper, http://anon.inf.tu-dresden.de/develop/Dokument.pdf

[12] Levine, B. N., Reiter, M. K., Wang, C., Wright, M. K., *Timing Attacks in Low-Latency Mix-Based Systems*, Proceedings of Financial Cryptography (FC '04), February 2004, LNCS 3110, Springer-Verlag.

[13] Möller, U., Cottrell, L., Palfrader, P., Sassaman, L. "Mixmaster Protocol—Version 2", July 2003.

[14] Murdoch, S. J., Danezis, G., *Low-Cost Traffic Analysis of Tor*, Proceedings of the 2005 IEEE Symposium on Security and Privacy, May 2005, IEEE CS.

[15] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[16] Serjantov, A., Sewell, P. *Passive Attack Analysis for Connection-Based Anonymity Systems*, Proceedings of ESORICS 2003, October 2003.

[17] Shmatikov, V., Wang, M.-H., *Timing Analysis in Low-Latency Mix Networks: Attacks and Defences*, Proceedings of ESORICS 2006, September 2006.

[18] Wendolsky, R., Herrmann, D., Federrath, H., *Performance Comparision of low-latency Anonymisation Services from User Perspective*, Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007), June 2007, Springer.

[19] Wiangsripanawan, R., Susilo, W., Safavi-Naini, R., *Design principles for low latency anonymous network systems secure against timing attacks*, Proceedings of the fifth Australasian symposium on ACSW frontiers (ACSW '07), 2007.

[20] Tóth, G., Hornák, Z., *Measuring Anonymity in a Non-adaptive, Real-time System*, Proceedings of Privacy Enhancing Technologies workshop (PET 2004), LNCS 3424, Springer-Verlag.

[21] Yu, W., Fu, X., Graham, S., Xuan, D., Zhao, W., *DSSS-Based Flow Marking Technique for Invisible Traceback*, SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy, pp. 18–32, 2007, IEEE Computer Society.

[22] Zhu, Y., Fu, X., Graham, B., Bettati, R., Zhao, W., *On Flow Correlation Attacks and Countermeasures in Mix Networks*, Proceedings of Privacy Enhancing Technologies workshop (PET 2004), LNCS 3424.

[23] Zhu, Y., Bettati, R., *Unmixing Mix Traffic*, Proceedings of Privacy Enhancing Technologies workshop (PET 2005), May 2005.