How (not) to Build a Transport Layer for Anonymity Overlays

Florian Tschorsch Björn Scheuermann

Institute of Computer Science 4
University of Bonn, Germany
{tschorsch, scheuermann}@cs.uni-bonn.de

ABSTRACT

Internet anonymity systems, like for instance Tor, are in widespread use today. Technically they are realized as overlays, i. e., they add another instance of routing, forwarding, and transport functionality on top of the Internet protocol stack. This has important (and often subtle) implications, as overlay and underlay may interact. Far too often, existing designs neglect this. Consequently, they suffer from performance issues that are hard to track down and fix. The existing body of work in this area often takes a quite narrow view, tweaking the design in order to improve one specific aspect. The behavior of the interacting underlay and overlay transport layers is complex, though, and often cause unexpected—and unexplored—side effects. Therefore, we show that so far considered combinations of overlay and underlay protocols cannot deliver good throughput, latency, and fairness at the same time, and we establish guidelines for a future, better suited transport layer design.

1. INTRODUCTION

Overlay protocol designers, including those of anonymity overlays, tend to perceive overlay links as an equivalent of dedicated point-to-point links, just like the ones forming the basis of the Internet. Another instance of data forwarding and transport functionality is then added on the overlay level. There, it is tempting to take up concepts from their Internet counterparts. An unreflected re-use is treacherous, though: in various contexts, it has already been shown (and sometimes painfully felt) that end-to-end Internet connections may not be mistaken as point-to-point links.

One well-known example is the *TCP meltdown* effect: if a TCP connection is sent through a TCP-based VPN tunnel, the stacked TCP implementations will start interacting. This can cost 55 % or more throughput performance [11]. There are more such effects and constraints which are often and easily overlooked.

The anonymity overlays deployed and used today, including Tor [5], JAP [4], or I2P [10], suffer from severe performance problems, mostly due to various transport layer effects [6]. Therefore, it is highly necessary to think about anonymity overlays from a network performance perspective: while there is a significant body of work on cryptographic aspects, there are surprisingly little insights into how to design the overlay in such a way that it makes efficient and proper use of network resources. Previous work typically focused on isolated aspects, and the proposed mechanisms more often than not cause undesired deteriorations in other parts of the system. Means from related areas such as VPNs and Mobile IP tend to move the logic to the network edge to keep the core simple and clean. In anonymity overlays, this is generally not desirable, though, as end-to-end mechanisms always increase the danger of information leakage. Therefore, nodes in anonymity overlays typically maintain per-connection status—and transport layer designs can make use of that fact.

This paper intends to provide an integral perspective on performance aspects of transport in anonymity overlays. In particular, we are interested in the interrelations between underlay transport, transmission scheduling, and overlay transport, with respect to throughput, latency, and fairness. Our key contribution is that we show that anonymity overlays cannot in general deliver satisfactory performance with any so far considered combination of overlay and underlay protocols. We will argue that an overlay-aware joint congestion control mechanism for multiple sources is necessary. Such a mechanism would make use of the abovementioned per-connection knowledge of anonymity overlay nodes in order to implement suitable rate control.

The remainder of the paper is structured as follows. We first discuss related work in Sec. 2. An overview of the design space follows in Sec. 3. We then discuss the effects of different design decisions on throughput and packet loss frequency in Sec. 4, on data transmission latency in Sec. 5, and on fairness aspects in Sec. 6. In Sec. 7, we conclude our findings and derive lessons on the choice of transport layer mechanisms.

2. RELATED WORK

In recent years, a number of transport layer modifications for the Tor anonymity overlay have been proposed [3, 12, 19, 25]. They focus on different weaknesses in the current design [6]. Even though each proposal improves individual aspects, they often cause undesirable side effects. In the remainder of this paper, we will take up and discuss some of the ideas. A descriptive comparison of feasibility is provided in [15]. Here, our focus is on a performance-oriented perspective and on establishing guidelines for well-suited designs. Approaches which improve the performance by classifying and prioritizing traffic [2, 22] or modified path selection [1] are orthogonal to the problem considered here.

Also beyond specific improvements of Tor, there is existing literature on transport aspects of overlays. As an example, [9] on congestion control in Gnutella is noteworthy. The focus is on per-hop TCP connections; as we will see, the design space for anonymity systems is much larger.

BitTorrent recently introduced a transport protocol named μTP [21]. μTP implements congestion control based on the "less-than best-effort" principle [20]. It aims to be "over-friendly" to TCP, i. e., to give way for TCP connections when sharing a bottleneck. This is reasonable, since BitTorrent intends to make use of remain-

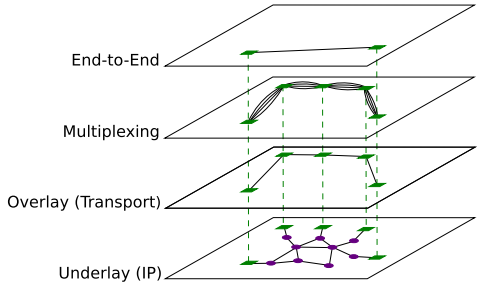

Figure 1: Layers in the transport architecture design space.

ing bandwidth in the background. But it is not desirable for interactive applications like web browsing in anonymity overlays.

For a setting in which a mix network is used to re-order IP packets, Fu et al. [8] argue that reordering and batching will lead to more duplicate ACKs. As a countermeasure they propose to modify the *dupthresh* parameter in the TCP implementation. In [18], Tor's end-to-end throughput was examined, showing dependencies on the path length. In both [8] and [18], Floyd and Fall's TCP model [7] was used. In some aspects of our analysis, we build upon Padhye et al.'s TCP model [17], which is more sophisticated and considers TCP implementation parameters. This will provide valuable hints for the overlay transport layer design.

3. DESIGN SPACE

In anonymization overlays like Tor [5], unicast connections are forwarded over a sequence of overlay links. These overlay links are transport layer connections through the Internet. We illustrate this in the bottom two layers of Fig. 1. In analogy to the terminology of circuit-switched networks (and also resembling Tor's terminology), we call the end-to-end unicast tunnels *circuits*.

The most obvious design choice is the transport protocol used for hop-by-hop communication, i. e., for the overlay links. Overlay links may, in principle, be based on TCP, UDP, or any other protocol implemented on its endpoints. In practice, only TCP and UDP are widely available. Both can be extended to provide security and message integrity by SSL/TLS or DTLS, respectively. To avoid the necessity of implementing a transport protocol in the operating system kernel, alternative transport layers are sometimes realized in user space and tunneled over UDP. Another reason for the same measure can be to not use overly many transport layer sockets. Realizing alternative transport layers in this way is possible since UDP—apart from port addressing—does not change the service model of IP. An example is BitTorrent's μTP [21]; Reardon et al. propose an architecture for Tor which tunnels application-layer TCP connections over UDP [19].

A single overlay link can be traversed by a varying number of circuits. The most commonly used architecture—and arguably the one closest to the "idea" of an overlay—is that all circuits traversing a given overlay link share one common transport layer connection. However, it is also conceivable to establish multiple parallel transport layer connections, as in [19]. As shown in Fig. 1, this degree of freedom can be perceived as another layer in the design space, which we call the *multiplexing layer*: here, circuits are multiplexed into one or more transport layer connections.

One of the most central aspects in the choice of the transport protocol is an appropriate congestion control mechanism. The choice for a transport protocol on each overlay link implies a choice for a congestion control mechanism used on that link. There is a broad spectrum of options: TCP alone comes in a variety of different flavors. Further possibilities range from an application-specific mechanism (as in μTP) or no congestion control at all in case of UDP.

The latter need not necessarily be a bad idea, if appropriate congestion control is performed on higher layers, e. g., on the circuit level. Indeed, it is noteworthy that per-link congestion control alone does not suffice anyway: consider a circuit which first traverses a high-bandwidth link, and then subsequently a much tighter one. If there were no feedback, data would pile up before the bottleneck, and would either need to be dropped or lead to excessive queues. Therefore, regardless whether a design implements congestion control on individual overlay links, circuit-level congestion feedback is always necessary. In Tor, for example, TCP is used on each link, complemented by a circuit-level end-to-end sliding window.

Note that none of these concepts limits whether the traffic that is carried *through* a circuit is UDP-like datagram or TCP-like bytestream traffic. Given proper encapsulation, any overlay design can carry both. The only noteworthy difference is that pure UDP-like traffic allows for additional degrees of freedom, since reliability and order guarantees need not be provided then.

In all cases, one needs to be aware of the implications on anonymity. In general, end-to-end approaches come at a higher risk of leaking information, as headers and parameters are forwarded in an unmodified way along the whole circuit. This is also the reason why active queue management techniques (like RED or ECN) do not appear to be a wise choice within an anonymity overlay. On a general level, hop-by-hop feedback seems more appropriate, because it does not directly reveal parameter choices or other information to further away nodes, and because it can make use of per-circuit knowledge that anyway exists in intermediate nodes.

4. THROUGHPUT AND LOSS

4.1 Analytical throughput characterization

The achieved throughput is perhaps what first comes to mind when the aim is to "improve the performance" of an anonymity overlay. Not surprisingly, many of the existing works focus on improving throughput performance by more or less subtle modifications. Their effects are typically demonstrated either by measurements on real hard- and software, or by simulations. Both can only cover specific, narrow settings, though.

More general insights can be obtained by taking analytical characterizations of transport protocol performance into account. The TCP performance model by Padhye et al. [17] is particularly noteworthy. TCP is by far the most important transport protocol, and virtually all anonymity overlay designs make use of TCP in one way or another. TCP performs congestion control based on packet loss: when transmitted segments do not arrive at the receiver, this is taken as an indication of network congestion. Assume that the number b of packets that are typically acknowledged in a single ACK, the retransmission timeout (RTO) T_0, and the round trip time (RTT) of the network path are given. If we furthermore assume that there is no hard limit on the window size (i. e., TCP is free to make full use of the available bandwidth), Padhye et al. give the following formula for TCP Reno, which sets the packet loss frequency p observed by the TCP connection and the achieved throughput B into relation:

$$B \approx \frac{1}{\mathrm{RTT}\cdot\sqrt{\frac{2bp}{3}} + T_0 \cdot \min\left\{1, 3\sqrt{\frac{3bp}{8}}\right\} \cdot p \cdot (1+32p^2)}. \quad (1)$$

Typical parameter values are $b = 2$ (cf. delayed ACKs) and $T_0 = 0.2\,\mathrm{s}$ (cf. initial RTO).

In a nutshell, a higher loss rate p will increase the denominator of (1), and thus—all other parameters unchanged—corresponds to a lower achieved throughput. Note that the available bandwidth of the used network links does *not* appear in the formula: if the available bandwidth is exceeded, packet drops will occur (i. e., p will increase), which in turn implies a lower throughput. Padhye et al.'s model describes the *steady state*, in which bandwidth and observed packet loss have reached an equilibrium.

This also underlines that throughput and loss are closely intertwined, so that they cannot be treated independently. While some of the more modern, post-Reno TCP variants (NewReno, Vegas, CUBIC, etc.) vary this theme in different regards, this is still and generally at the heart of how TCP congestion control works. In fact, it is likewise the case for other congestion controlled transport protocols on the Internet: if they aim to be *TCP friendly*, they must react at least as strongly to packet loss as TCP, so as not to unfairly "steal" bandwidth from competing TCP connections.

Latency-based congestion control like, for instance, in TCP Vegas or μTP takes changes in the RTT into account in order to react early to congestion and to thus reduce the number of packet drops. Yet, this comes at the cost of a high sensitivity to changes in the RTT, as they are to be expected on long, bandwidth-scarce overlay paths. For TCP Vegas, a discussion can be found in [13]. It also means that such protocols react earlier than classical TCP variants if they compete for bandwidth at a common bottleneck—and that they therefore "lose" in such a competition.

4.2 Multiplexing on TCP overlay links

Perhaps the most important implication of the interrelation between throughput and packet loss becomes clear if we look at the multiplexing layer. Which differences should we expect if we either use (a) one single connection between each pair of onion routers shared by all circuits traversing the overlay link, or (b) multiple independent TCP connections, each carrying a single circuit?

For simplicity, assume that no other connections are present on an overlay link's underlay path and that n circuits are currently active on the overlay link. Then, in case of (a), one TCP connection with bandwidth B will carry all circuits. For (b), there are n connections, each with bandwidth B/n. We saw above that a lower bandwidth corresponds to a higher packet loss frequency. Therefore, if $n > 1$ connections share a bottleneck, each of them will *necessarily* experience a higher packet loss rate than one single connection across the same TCP bottleneck.

We show this effect in Fig. 2, where we "invert" Padhye et al.'s model: while (1) cannot be solved for p in closed form, we can use Newton's method to determine the value of p for which the predicted throughput reaches a given level. In the figure, we vary the number n of circuits, which are either (as, e. g., in Tor) multiplexed over a single connection or (as proposed, e. g., in [19]) use separate, parallel connections. Here, we chose a total bandwidth of 1 MBit/s and an RTT of 200 ms. As expected, the loss rate drastically increases for more and more parallel TCP connections.

Note that this is an inherent property of TCP congestion control. It does not depend on where exactly the congestion control mechanism is implemented, be it at the transport layer or in userspace tunneled over packet-based transport.

Of course, TCP is a reliable transport protocol, therefore losses will be repaired by retransmissions. Nevertheless, packet loss is a problem, because it causes additional delay: until the loss is repaired, no subsequent data can be forwarded to the application. We will soon look at this in more detail. For now, we conclude that a higher number of TCP connections on an overlay link very significantly increases the number of packet loss events.

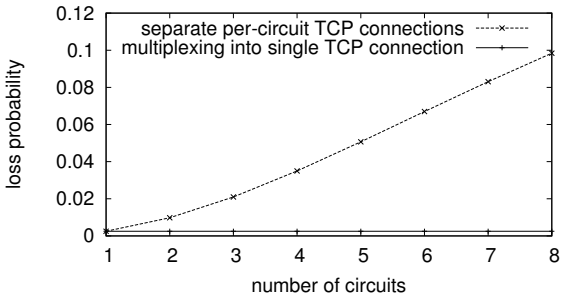

Figure 2: Loss probability according to Padhye model.

4.3 Stacking TCP connections

Not all TCP-related effects are covered by the Padhye model. One well-known problem, TCP meltdown, has already been mentioned before: when TCP connections are tunneled through a TCP-based VPN, then the "link" which they traverse (namely the VPN TCP connection) has entirely different characteristics than a typical Internet link. The resulting severe throughput loss is discussed in [11]. For anonymization overlays, this means that any design where multiple TCP connections are "stacked" must be avoided.

4.4 Transitional effects

While the Padhye model only covers the steady state, transitional effects also need to be considered. This becomes clear when we contrast the alternatives of using TCP on the end-to-end level in combination with datagram transport on individual overlay links on one hand, and per-hop TCP connections on the other hand. The former is, for instance, used in [12] and [25].

There are two effects that should be taken into account: first, we observe that the former approach implies separate TCP connections per end-to-end connection. Therefore, if multiple circuits share one overlay link, there necessarily are multiple parallel TCP connections. Consequently, similar observations as above hold: the bandwidth per TCP connection will be lower, and a higher frequency of packet losses is to be expected. This is particularly problematic in a setting with long TCP connections spanning the whole path of a circuit, because packet retransmissions for repairing these losses will take place end-to-end through the overlay. Due to the longer RTT, they will take longer to be detected and to be repaired.

Second, the high round trip time of an end-to-end path through an anonymization overlay has implications on TCP dynamics before the steady state is reached: classical TCP variants will take much longer to initially ramp up the bandwidth. In order to get an idea of the time span for this ramp-up, we performed network simulations with ns-3 [16], an advanced packet-level network simulator. We implemented an overlay in ns-3, which we modeled closely after Tor. We used TCP NewReno, which is today (beside CUBIC) the most common TCP implementation. The scenario is intentionally kept very simple, in order to clearly show the key reasons for the performance problems that will, of course, likewise be present in more complex setups. Circuits with fixed-window end-to-end congestion control traverse a sequence of three intermediate overlay nodes (= onion routers). All up- and downstream connections of a given overlay node share a common link to the network core. The round trip time between any pair of nodes through the underlay is set to 80 ms. All links are configured to 10 MBit/s, except for one bottleneck link (between the intermediate overlay nodes closest to the client) which is restricted to 1 MBit/s.

Ideally, the throughput achieved by the circuit would be 1 MBit/s.

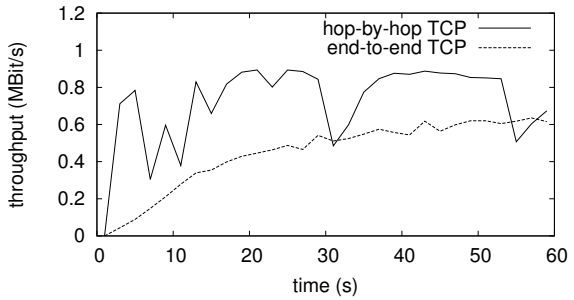

Figure 3: Circuit throughput in ns-3 simulations.

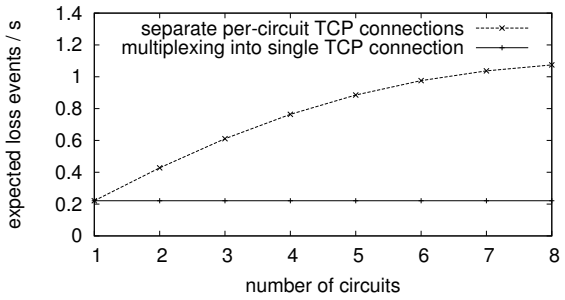

Figure 4: Loss event frequency per circuit.

In practice, protocol overheads and the effects of congestion control will not allow to always fully utilize the bandwidth.

In Fig. 3, we show the throughput of a single circuit over time, measured end-to-end on the application layer. The *hop-by-hop TCP* line shows the result with separate TCP connections along each overlay hop. *End-to-end TCP* denotes one single TCP connection along the whole circuit, forwarded over datagram transport. As can be seen, the latter takes time in the order of one minute to reach a throughput level that is comparable to what hop-by-hop TCP can deliver almost instantaneously.

TCP CUBIC ramps up based on real-time clock ticks and thus overcomes the RTT dependency. However, as shown in [14], CUBIC generally—that is, also for short RTTs—converges to the long-term bandwidth rather slowly. Therefore, our conclusion from these results is that end-to-end TCP (or TCP-like) congestion control reacts far too slowly to constitute a viable design alternative.

5. DELAY

Because anonymization networks forward data multiples times over potentially long Internet links, higher latency compared to a direct connection is inevitable. The overlay nodes are located all around the world, so that the sum of delays on the traversed overlay links can easily be hundreds of milliseconds. The question is: how much *additional* delay is caused by the transport layer design?

Additional delays occur when data is queued for processing or forwarding. An interesting observation is that if multiple TCP connections are used in parallel, then queuing delays on the IP layer will typically be higher. This can be understood by again looking at Padhye's TCP model: a higher number of parallel TCP connections through a common bottleneck will, as discussed before, result in a higher packet loss rate. That is, queue overflows—the source of packet loss—happen more frequently. Consequently, higher packet loss rates also correspond to queues that are, on average, longer.

We performed ns-3 simulations similar to those described above, and increased the number of circuits. We found that the queue lengths are indeed 10% higher than for a single TCP connection, with correspondingly higher queuing delays.

For reliable transport protocols like TCP, packet loss is another source of delays: lost packets need to be retransmitted. As observed by Reardon et al. [19], this has severe implications if multiple circuits are multiplexed over one transport layer connection: TCP delivers one single reliable, in-order bytestream. It does not distinguish between data from different circuits. Consequently, a missing segment with data from one circuit will also temporarily stall any other circuit on the same connection.

Reardon et al.'s remedy is to use separate TCP connections. As seen above, though, this comes at the cost of an increased loss rate for each individual connection. To assess the impact, we build upon our results from Sec. 4.2 above and determine the frequency at which loss events would affect any individual circuit (assuming TCP segments of 1500 byte). The results are shown in Fig. 4. The additional delays in fact *overcompensate* the gain that results from separating circuits into independent connections: each circuits suffers *much more often* from segment recovery delays.

Apart from network layer queues and the need to wait for retransmissions, application layer queues in the overlay implementation can also cause significant delays. For example, as we have shown in earlier work [23], significant latency in Tor is caused by undesired interactions of two separate rate limiting mechanisms.

6. FAIRNESS

The question of fairness refers to the sharing of bandwidth: do multiple independent, competing connections get equal (fair) shares of the available bandwidth? Relevant is both fairness *within* the anonymization overlay (i. e., between circuits) and fairness between the anonymization overlay and other applications.

6.1 Fairness within the overlay

Concurrent TCP connections tend to share resources fairly: connections traversing the same bottleneck will experience the same loss probability. In the Internet as an underlay, TCP connections with different RTTs will share bottlenecks, so that perfect fairness cannot be taken for granted anyway. In fact, since higher RTTs at equal loss rates result in lower throughput for most TCP variants, high-delay connections "suffer" twice. As mentioned before, TCP CUBIC overcomes this dependency, but in turn converges much slower to a fair bandwidth share.

Even if *overlay* links share the bandwidth fairly, this does not imply a fair sharing between *circuits*. For instance, assume that two circuits are multiplexed over one transport layer connection. Another connection uses equal bandwidth, but carries only one circuit. Then bandwidth is in fact *not* fairly shared between circuits. We showed such unfairness effects for Tor in [24].

Solving these problems is hard, because it would require *unequal*, "unfair" sharing of bandwidth between TCP connections in order to prefer connections with a higher number of circuits. The gross unfairness between circuits could be remedied by not multiplexing multiple circuits into one connection. Yet, the severe drawbacks of doing so have been discussed above. In order to achieve fairness with multiplexing, it would be necessary to adjust the relative fairness level across competing TCP connections, based on the number of active circuits that they currently carry. This, however, is not possible with current TCP algorithms.

6.2 Fairness towards other applications

Fairness should also be considered towards other applications. This is easy to see from a straightforward gedankenexperiment: assume an anonymization overlay using n parallel connections across

a network link (e. g., the Internet link of an overlay node), and a separate, independent application which also has a TCP connection open on that link. For comparable RTTs at least, TCP connections share bandwidth fairly. Thus, since there are $n+1$ connections in total, n of which belong to the anonymization overlay, this overlay will get a total share of $n/(n+1)$ of the bandwidth, while the other application gets only $1/(n+1)$.

A higher number of (TCP-based or TCP-friendly) transport layer connections will thus, if considered in sum, be more aggressive. Proposals which increase the number of TCP connections therefore are at risk to unfairly disadvantage other applications running in parallel to the anonymization overlay.

Unequal sharing of the bandwidth need not even necessarily be wrong or undesired—but it should be independent from the number of active circuits and overlay links (i. e., n). Ideally, it should be configurable by the user. This can even be considered an additional incentive for donating bandwidth: overlay node operators might be more willing to allocate additional bandwidth if they are able to configure how aggressively this bandwidth is "claimed" by the overlay if other applications also have demand.

7. LESSONS LEARNED

In summary, what have we learned about the choice of transport protocols when building anonymity overlays? Can we narrow down the design space?

First, we conclude that pure end-to-end mechanisms without reliability and congestion control on individual overlay links are not a favorable design choice. The high total latency along an entire path through an anonymity overlay will necessarily lead to very long reaction times for repairing packet loss and to slow convergence towards a balanced bandwidth share. The high number of connections causes unfairness towards other applications. This is particularly undesirable since the degree of unfairness heavily depends on the current load situation in the overlay; it is thus outside the sphere of influence of the node operator.

Separate per-hop, per-circuit connections are likewise not a convincing solution: the aggressive behavior of a large number of bundled TCP connections results in (again, non-controllable) unfairness towards other applications and in frequent packet loss. The latter, in turn, causes frequent delays while waiting for retransmissions. Multiplexing circuits into shared connections, on the other hand, raises fairness issues that cannot be fully resolved by changes in the application layer alone.

From these insights it becomes clear that a congestion control algorithm that considers only the individual circuit level is not sufficient—and an algorithm which operates only on overlay links likewise is not. Consequently, it is necessary to take into account *both* the circuit level and the information which circuits share an underlay path. We therefore envision an overlay-aware algorithm which locally performs *joint* congestion control for all circuits traversing an overlay node. Such a scheme would observe packet loss and/or RTT variations along *all* outgoing overlay links, and would continuously adjust the assigned bandwidths. It would therefore make use of knowledge about individual circuits, and thus of the specifics of anonymity overlays. Such a scheme could well be implemented in the application/multiplexing layer over UDP transport, without modifications to the overlay nodes' operating system.

Of course, as discussed in Sec. 3, this needs to be combined with some form of end-to-end congestion feedback to avoid excessive queuing. Again, it seems wise to make use of the fact that intermediate overlay nodes already maintain per-circuit status information: nodes in an anonymization overlay can (and should) actively contribute to per-circuit congestion control by appropriate feedback and queue management. We believe that a hop-by-hop, backpressure-based congestion control scheme is a promising direction. First steps in this direction have actually already been taken by experimentally applying the N23 scheme to Tor [3].

Acknowledgments

The authors thank the DFG for funding this work.

8. REFERENCES

[1] M. Akhoondi, C. Yu, and H. V. Madhyastha. LASTor: A low-latency AS-aware Tor client. In *SP '12*, May 2012.

[2] M. AlSabah, K. Bauer, and I. Goldberg. Enhancing Tor's performance using real-time traffic classification. Technical Report CACR 2012-12, University of Waterloo, Canada, 2012.

[3] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. Voelker. DefenestraTor: Throwing out windows in Tor. In *PETS '11*, July 2011.

[4] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In *DIAU '00*, July 2000.

[5] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security '04*, Aug. 2004.

[6] R. Dingledine and S. J. Murdoch. Performance improvements on Tor or, why Tor is slow and what we're going to do about it, Mar. 2009. www.torproject.org/press/presskit/2009-03-11-performance.pdf.

[7] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Trans. Netw.*, 7, August 1999.

[8] X. Fu, W. Yu, S. Jiang, S. Graham, and Y. Guan. TCP performance in flow-based mix networks: Modeling and analysis. *IEEE Trans. Parallel Distrib. Syst.*, 20, May 2009.

[9] Q. He and M. Ammar. Congestion control and message loss in Gnutella networks. In *MMCN '04*, Jan. 2004.

[10] I2P: Invisible Internet Project. www.i2p2.de.

[11] S. Khanvilkar and A. Khokhar. Virtual private networks: An overview with performance evaluation. *IEEE Communications Magazine*, 42(10):146–154, Oct. 2004.

[12] C. Kiraly, G. Bianchi, and R. Lo Cigno. Solving performance issues in anonymization overlays with a L3 approach. Technical Report DISI-08-041, Ver. 1.1, Univ. degli Studi di Trento, Sept. 2008.

[13] R. J. La, J. Walrand, and V. Anantharam. Issues in TCP Vegas. Technical Report M99/3, Univ. of California, Berkeley, Jan. 1999.

[14] D. J. Leith, R. Shorten, and G. McCullagh. Experimental evaluation of Cubic-TCP. In *PFLDnet '08*, Mar. 2008.

[15] S. J. Murdoch. Comparison of Tor datagram designs. Technical report, Nov. 2011. www.cl.cam.ac.uk/~sjm217/papers/tor11datagramcomparison.pdf.

[16] ns-3 network simulator. www.nsnam.org.

[17] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *SIGCOMM '98*, pages 303–314, Aug. 1998.

[18] R. Pries, W. Yu, S. Graham, and X. Fu. On performance bottleneck of anonymous communication networks. In *IPDPS '08*, Apr. 2008.

[19] J. Reardon and I. Goldberg. Improving tor using a TCP-over-DTLS tunnel. In *USENIX Security '09*, Aug. 2009.

[20] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind. IETF-draft: Low extra delay background transport, Oct. 2011. www.tools.ietf.org/pdf/draft-ietf-ledbat-congestion-09.

[21] L. Strigeus, G. Hazel, S. Shalunov, A. Norberg, and B. Cohen. BEP 29: μTorrent transport protocol. www.bittorrent.org/beps/bep_0029.html.

[22] C. Tang and I. Goldberg. An improved algorithm for Tor circuit scheduling. In *CCS '10*, pages 329–339, Oct. 2010.

[23] F. Tschorsch and B. Scheuermann. Tor proposal 182: Credit bucket. https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/182-creditbucket.txt.

[24] F. Tschorsch and B. Scheuermann. Tor is unfair – and what to do about it. In *LCN '11*, pages 432–440, Oct. 2011.

[25] C. Viecco. UDP-OR: A fair onion transport design. In *HotPETS '08*, July 2008.