UDP-OR: A Fair Onion Transport Design

No Author Given

No Institute Given

Abstract. Low latency anonymity systems face many challenges. One of them is how to fairly allocate network resources among many unknown users and applications. This paper presents a design for a end-to-end inspired transport mechanism for onion routers. The design uses the same solutions and implementations that had made the Internet scale. We compare an implementation of the design with Tor's (the third generation onion router) [1] and show initial benefits of the design.

1 Introduction

Onion routing is a mechanism to achieve low latency anonymous communication for connection oriented applications[1]. Onion routing uses two kinds of nodes to provide anonymity: "onion routers"(OR) and "clients". To use the system a client selects a path through a set of three or more OR's and builds a circuit through them, so that each OR node knows is predecessor and successor but no other part of the path. Data is transmitted on the network via fixed sized *cells*, which are unwrapped(wrapped) by a symmetric key at each node and relayed downstream(upstream) from the client to the server (server to client).

There have been many OR designs in the literature, Tor[2] being the most successfully deployed and used. The design issues presented by this paper aims to overcome some of the problems in performance that impact daily use by users and that allow some attacks. The design is build around the idea that the OR infrastructure does not need to transport connection oriented reliable data streams but that will transport connection oriented best effort message transfer of TCP friendly streams is sufficient for most applications. The removal of the stream requirement allows for designing routers that are closer to IP routers and that require much less state. The transport design presented here builds upon Tor and provides the following improvements:

Fair allocation of network resources. The use of true end-to-end congestion and flow control allows the system to be self regulating in the availability of network resources and use. We use the use exact mechanisms and implementations of TCP that have enabled the Internet to scale from a connections of a few kilobits per second to multi-gigabit connections. This can prevent some attacks such as the circuit clogging attack [3].

Router detection of misbehaving circuits. The simplification of the network behavior to a best effort service allows routers to detect when circuits

[1] This naming scheme follows conversations with Paul Syverson.

are not TCP-friendly (respond to packet loss by lowering their throughput). This allows routers to limit the effects of misbehaving circuits benefiting well behaved circuits.

Client detection of misbehaving routers. The simplification of the network behavior also allows clients to detect dynamically when routers do not behave throughput wise as advertised. This is useful to prevent attackers that behave differently than advertised avoiding this some class of low resource attacks.

Separation of transport mechanism from client interface. We come one step further than Tor and closer to the Freedom Network. This allows the network to provide future functionality without changing the network nodes. Further it allows for other potential uses such as VOIP, streaming video or gaming.

All of these benefits are on the data transport mechanism. For other system related problems such as router discovery and advertisement the design will copy Tor's design. This design is an improvement of Tor rather than a complete redesign.

2 Previous Work

The design presented in this paper builds upon Tor, but uses the end-to-end design principles from which the Internet was designed[4].

2.1 Anonymity Systems and Tor

The theory behind anonymous communications was initially developed in the 1980s. In general, there are two different approaches to achieve anonymity (both originating from David Chaum): the use of mixes(mix networks)[5] and the use of protocols based on the Dining Cryptographers Problem(DC-nets)[6]. Mix networks provide sender anonymity: the identity of the message sender is not linkable to the message. DC networks achieve both sender and receiver anonymity; however, both sender and receiver must belong to a "broadcast domain".

Most low latency anonymity systems, where the delay in message is in the order of a few seconds, are based in the mix network design. Onion routing belongs to this category and here is where we concentrate our efforts. Low latency systems assume a 'weak' adversary model. This adversary model is a non-global observer, which is in contrast of high latency systems such as Mix-minion[7] and Babel [8]. Aslo it is common to assume that attackers with common goals can control a non significant portion of the router infrastructure.

Other approaches to low latency anonymity systems include systems such as the Anonymizer[9] and Ghostsurf [10]. These systems depend on a trusted third party not to disclose the mappings from input to outputs to the users' adversary. Recent developments such as the case of communication wiretapping[11, 12] and revealing of user keys[13, 14] demostrate the weaknesses of systems that depend on a single third party.

Over the years, many low latency have been designed: Tor(onion routing) [1, 2], Tarzan [15], Cebolla [16], Crowds [17], pipenet [18], The Freedom Network [19], Morphmix [20, 21], JAP [22], Hordes [23], Herbivore[24], P5[25], and SAS [26]. All of these systems have their own strengths and weaknesses in the anonymity guarantees, deployability and performance. A high level comparison of these systems can be found on Table 1.

	Transport Protocol	*Client Interface*	*What Transfers*	*Node Topology*	*Cover traffic?*	*User node control?*	*Node Discovery*	*Route Selection*	*Implementation?*
Tor	TCP	Socks	Streams	Core	No	Yes	Central.	Source	Yes
Tarzan	UDP	Net. API	Datagrm.	p2p[a]	Yes	Yes	Distrib.	Source[b]	Yes
Cebolla	UDP	IP(?)	Datagrm.	p2p(?)	No	Yes	Distrib.	Source	No
Crowds	TCP	proxy	Datagrm.	p2p	No	Yes	Central.	Rand.	Yes
Freedom	UDP	N. Stack	Datagrm.	Core	No	No	Central.	Source	Yes
Morphmix	TCP	proxy	Streams	p2p	No	Yes	Distrib.	Rand.	No
PipeNet	TCP(?)	?	Strm(?)	Core	Yes	?	?	?	No
JAP	TCP	proxy	Streams	Core	Yes	No	Central.	Source	Yes
Hordes	TCP[c]	?	Datagrm.	p2p	No	Yes	Central.	Rand.	No
Herbivore	TCP	V.Net	Datagrm.	Hierarch.[d]	Yes	Yes	?	No [e]	Yes
P5	UDP(?)	?	Datagrm.	Hierarch.	Yes	Yes	?	No	No
SAS[f]	UDP(?)	?	Datagrm.	Core	No	Yes	?	Source	No

[a] By p2p we mean that each node connects directly to only a subset of all the nodes
[b] With "mimic" constraints.
[c] also uses multicast
[d] Forming a ring of cliques.
[e] Uses a point to point protocol
[f] Has no implementation or complete specification so many of the details are not defined.

Table 1. Comparison of Multihop low latency anonymity Systems

For our design, the UDP based systems are important to mention: Freedom, Cebolla, P5 and the Tor UDP transport proposal by Liberatore[27]. Of particular importance is Freedom, not only because it transmits encapsulated datagrams (similar to the proposal described here) but also because it was for a while a successful commercial privacy preserving network.

2.2 Network Performance and Fairness

Efficient use of communication resources is not new. The design of the TCP/IP protocol had into its goals inspired by how to effectively use and share communication networks. The design presented here uses the same TCP found in deployed operating systems as the mechanism to ensure end-to-end congestion avoidance and flow control. The goal is to use the evolution of the protocol (TCP Tahoe, TCP Reno[28], TCP Vegas[29]) and of the extensions for high bandwidth delay networks[30] that are already implemented.

Of particular importance to guarantee the fairness of the bandwidth in this design is the work of Sally Floyd and Kevin Fall [31, 32]. The algorithms described in their work are the ones that will be used by the ORs to detect non-TCP-friendly circuits. Non-TCP-friendly circuits are considered misbehaving in this design as they can potentially try to use an unfair amount of bandwidth in the network. Once a circuit has been detected as misbehaving the OR can employ different mechanisms to limit its effects to the network. In the current implementation routers only detect misbehaving circuits, but is is planned that this will be assigned only a small percentile of the available bandwidth.

The design also mandates for mechanisms to help TCP optimally perform. In particular all OR will implement Random Early Drop (RED). Random Early drop, introduced by Floyd and Jacobson[33], as is a mechanism that helps congestion avoidance by avoiding tail drop queues. The basic rationale of the design is that by stochastical packet drops once a router queue has reached a certain threshold signals TCP connections to start lowering the throughput before congestion takes place. Finally it is desired that queue management should be done by the OR and not by the Internet Service Provider (ISP). This is due to the fact that in many home connections the ISP implements large queues, this tail drop queues affect TCP so that it does not perform at is optimum point when many TCP connections share a connection as is the case of our design.

3 The Transport Design

The design of the transport follows the following objective: *'connection oriented best effort transport of TCP friendly message flows'*. However, introducing a best effort network leads to question where should the side effects of such network (packet loss, reordering) would be handled and by whom. We will address the design objectives, philosophy, tradeoffs and implementation in this section.

3.1 Objectives and Non-Objectives

Objectives

1. *Use a SOCKS interface for applications.* In order to maintain compatibility with network applications we will use the same SOCKS interface that users will be able to use their current applications without modification. This is also to allow users familiar with Tor to use our system with minimum configuration changes.

2. *Separation of Service applications from onion routing infrastructure.* It is common to design anonymity systems where the service provided is integrated into the code of the anonymity system. One of the core concepts of the design is to separate the socks server from the or code so that: (*i*) the number of lines of code is reduced, (*ii*) users can decide to use other applications for service, (*iii*) other services can be easily added to the system, and (*iv*)performance problems can be easier to diagnose.
3. *Minimization of State in for the intermediate nodes.* One of the problems associated with scalability is the constant need for more resources and the need to keep state for callback information. The design will push the need to check for transition changes to the client nodes. The onion router will not have any callback functions. Also, elimination of network buffers is desired allow better scalability of the network.
4. *End-to-End Congestion control, flow control and integrity checking.* The core assumption of the Design is that by allowing end-to-end control of the communication channel, the fairness of all the flows will be achieved. Further, we want to make sure that a single packet drop does not affect all other circuits in an OR connection. This lack of throughput independence allows several low resource attacks[34, 35] to exist.
5. *In core dection and mitigation of misbehaving flows.* While Congestion control has been moved to the network edges (Client application and exit node), it is necessary for the intermediate routers to validate the well behavior of the circuits, in order to ensure fair allocation of resources..

Non-objectives There are several non-objectives of our design, in particular we want to mention:

1. *Steganography of the transport.* The design will not care about the network signatures of the OR design. It will be assumed that an eveasdropper will be able to identify with certainty that a communication pattern and/or content is of onion packets.
2. *Discovery of Servers and server announcement.* The design only cares about transport. Important issues for a real system like discovery and server announcement are not addresses by the design.
3. *Transition phase.* The design does not address any of the transition problems associated with integrating the design with other systems like tor.
4. *File Descriptor Exhaustion.* The design will not address the problem of File descriptor exhaustion problem. While this is a problem acknowledged by several anonymity systems, the design assumes that this problem is not worse that it is with the Tor codebase.
5. *No hidden services.* There is no hidden service provision in the current design. This could be added in the future as the design does not limit how the paths are build in the anonymity network.

3.2 An overview of the Design

In general two types of communication network architectures exist: those that simplify the end nodes,such as the telephone network; and those that simplify the core nodes, such as the Internet. W

The end to end anonymity requirements only require that the information passed through the network is encapsulated to the intermediate routers and that the exit node cannot know who sent the originator of a request. Further we need to do it keeping a socks interface for client applications. The solution this problem is as follows and the core of our design is: *move the socks proxy to the exit node and make the entry node an interface to a virtual network*. Quasi-TCP packets are transported in a best effort non reliable network. The OR network only job is to add or remove onion layers and to certify point to point packet streams. End to end packet loss, reordering and replay detection, should be handled by the end to end TCP session between the socks application and the socks server at the exit node. By quasi-TCP packets we refer to a packet header that contains all the TCP fields that cannot be reconstructed at each end based on the stream identification. For example source and destination ports are not needed as they are implicit by the stream id.

The network connections and transport layer of the design can be seen in Figure 1. This figure shows how unreliable connections are made between the Tor routers using UDP. TCP connections are established between applications and the socks server at the exit node. The interface to the clients will be a virtual network interface.

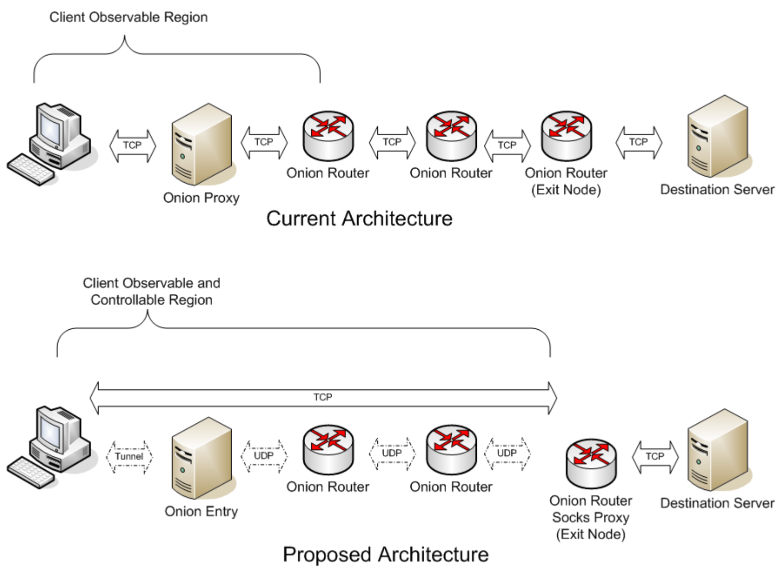

Fig. 1. TCP connections

Advantages The simplification of the OR interface provides many benefits:

1. *Significant reduction in OR requirements* Intermediate Onion routers no longer need to keep network buffers per connection. Only one file descriptor is used.
2. *True End to End congestion Control* Congestion Control has been moved to the Edges. This allows Tor not to bother with it and let TCP handle congestion and flow control. This means less code for TOR while taking maximum advantage of the host's TCP/IP stacks.
3. *Performance can be Measured and Adapted by the end hosts* Recently a technical paper by a set of researches at the University of Colorado demonstrated that low resource attacks are possible against Tor[35]. This attack is possible due to the fact that a Tor node selection strategy tries to balance bandwidth among the nodes. The defense against this attack requires measuring the effective bandwidth over time of tor nodes. Estimating your bandwidth based on your 'regular' user experience with Tor's current architecture has a big problem: a user cannot tell weather the bandwidth problems are due to network congestion within the tor network or due to performance problems between the exit node and the final connection destination. Tor performance could still be measured directly with some accuracy, but having all nodes do this kind of tests most likely would cause the network to saturate and thus make the results not reliable. However, if there is only one TCP connection between the application and the socks server at the other edge of the network, a client can not only adapt to congestion in the network, but can also estimate the networks true available bandwidth based on the congestion window behavior if their connection to the Internet has more bandwidth than the bandwidth provided by the Tor network.
4. *DoS resistance* Related to scalability is the issue of DoS resistance. Since most nodes in the Tor Network are limited to 1024 connections, a malicious attacker can DoS the Tor network by taking over all the connections of the servers with more bandwidth. An attacker just needs to establish the connections and keep them alive.
5. *Firewall Piercing* The use of UDP and would allow to use techniques for 'Firewall' piercing on NAT's. Which would potentially increment the number of available nodes.
6. *Timing attack resistance* One of the attacks on Tor's anonymity [34] is possible due to the fact that Tor transforms network congestion into latency. By transforming network congestion into packet drops such attacks become more complex as the attacker needs to model the interaction not only of the tor node but of the unknown end node. There are still side effects in the network as the system shares resources, there will be more opportunities for research into attacks that detect bandwidth changes.

3.3 Design Tradeoffs

Addressing the Challenges The Tor Designers have published a document detailing some of the challenges for such networks[36]. In particular they address the problem of why not transporting packets but streams These are some of their concerns and why the designers believe the solution presented here is acceptable.

1. *IP packets reveal OS characteristics* Only the exit node will be able to determine the OS of the sender via its TCP fingerprint. The destination node would not be able to determine this. However as clients are already trusting the exit node for the contents of our traffic the extra risk can be considered minimal. Additionally, even the exit node chooses to change some of the QoS determined by OS, this would be possible to detect by the affected clients.
2. *Application level streams still need scrubbing* Interoperability with application specific proxies would not change. Only the destination host of privoxy would need to be changed.
3. *Certain protocols sill leak information* We would still be using the SOCKS interface. There would be no more leaking that what currently Tor leaks.
4. *The crypto has not been validated* The current cryptographic system has been presented to several cryptographers, an none has seen mayor faults. This is not sufficient, we require a real analysis of the cryptographic concepts used here. Further in the current implementation, we have
5. *We'll still need to tune network parameters* While TCP does most of the network tunning for the design, other parts need tunning. In particular the selection of the RED values is crucial to achieve good balance of new connections vs old connection in slow links (less than 256 kbps). We also need to address the link encryption flaws.
6. *Exit policies for arbitrary IP packets means building a secure IDS* The system will only expose valid TCP streams to external systems. For the exit socks server we might need to ensure valid TCP data, or leave that checking the OS.
7. *The Tor-internal name spaces would need to be redesigned* This is the only point where our architecture does not work. A potential solution is to have a different entry port for hidden services and force the clients to choose the appropriate server.

All solutions come with trade-offs, here is a list of the new problems introduced by the architecture.

1. *Need 'Root' privileges at Exit Nodes and Probably at entry nodes* Most OR implementations do not need system privileges to run an onion router. The new design, would require administrative privileges on the exit node or some permission modifications only available to the administrative user (The current prof-of-concept code also needs administrative privileges on the entry node, but this is a coding limitation not an architectural one).
2. *Fascist Firewalls* Currently Tor uses TCP and can be configured to listen on ports 80 or 443. There two ports are the well known ports for web traffic and web traffic under SSL, and are usually open on every network. By moving to UDP clients behind firewalls that only allow connections to this ports would be unable to use the Tor network.

3.4 Implementation Details

The transport design follows Tor's philosophy. However, two big changes are present: (*i*) two cell sizes are available and (*iii*) there is no router initiated notification of state changes to clients. The use of multiple cell sizes is required to achieve a good throughput due to the presence small of acknowledgment data traversing the network. The lack of router initiated notification of state change was designed to reduce the amount of state kept by the routers. The effort of state progress during construction of paths lies completely on the clients.

Cells All cells have eight fields, from two in the Tor design. The header is 24 bytes header with the following fields: version(4 bits), length (4bits), circuit id (3 bytes), alignment (4 bytes), initialization vector (8 bytes), checksum (7 bytes), command (4 bits) and command status (4 bits). The version number is 1, the length is the size of the cell in 128 byte increments, The implemented symmetric encryption algorithm is blowfish with a 128 bit key. The checksum is calculated by calculating the sha1 checksum of the payload and then extracting the seven most significant bytes of the checksum the circuit id is the current link circuit id. Once a circuit is established, routers identify packets that the need to process by checking the checksum. If the checksum matches they process the cell otherwise the cell is forwarded to the related reverse circuit if one exists. The alignment field is necessary to align the size of the opaque values (the checksum) with the size of the cells. Figure 2 illustrates the header, where the gray fields are opaque to all but the destination router in a circuit. When using encryption layers, the initialization vector is kept the same.

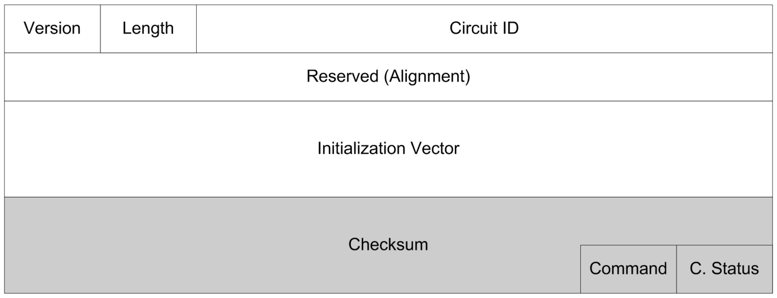

Fig. 2. Transport Headers

Stream cells, carry a stream header and a per protocol header. The stream header has a 2 byte stream id, a 3 bit protocol and options field and a 13 bit length field. For TCP streams, a quasi-TCP header is also appended, this quasi-TCP header contains all the TCP fields that cannot be reconstructed from the stream information(source and destination port) and the payload (checksum).

So that an almost perfect copy of the TCP packet can be reconstructed at the end of the tunnel.

New streams are automatically build at the exit node when the exit node detects that a stream does not exists. If the stream cannot be built, a reverse ICMP message is returned to the circuit origin. Streams can be deleted explicitly by the client nodes or via timeouts at the exit node. Currently timeouts are set at 120 seconds of stream inactivity.

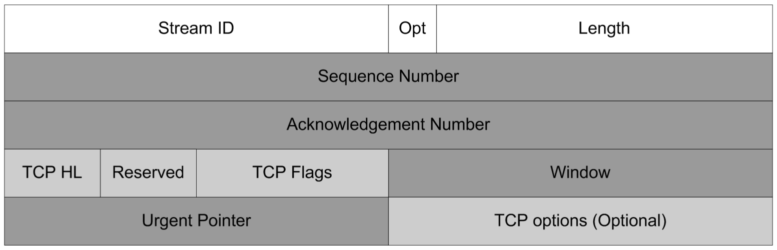

Fig. 3. Transport Headers

Circuits Circuits are built in a 'telescoping' manner, just like Tor's. However, unlike Tor each step of the circuit building process can fail and is up to the clients to verify that the circuit has been built. The same goes to requests to extend a connection to another host. Another difference is that this design does not 'extends' circuits in the same way as Tor does. The process of extending a circuit is a two step process. First, a request to another host is issued (this builds a path to the requested router). Second, a new circuit hand-shake is done from the client node to the destination router.

Circuit destruction can be explicitly requested by the client or exit nodes or can be automatically done via timeouts. Currently circuit creation by clients is done by an independent thread that monitors circuit creation and state changes. Available OR systems are found in a file that comes with the distribution.

Link Encryption For link encryption the authors decided to use a homegrown datagram cryptography layer. The reason of why to do this instead of other solutions like DTLS[37] was due to the desire to use a connectionless api, unfamiliarity with DTLS and to prevent some message alignment issues that exist with DTLS[37]. A move to DTL is planned for the next release of the transport design.

Other Details Since we have decoupled the SOCKS server application from the OR infrastructure it is necessary to select a compatible SOCKS server for clients to use. We use Antinat[38] as our socks server for our tests.

4 Measurements and Comparison with tor

This section compares the design with Tor's. First we model some of the differences and scaling properties of the design and then me do some actual measurements on the developed software.

4.1 Modeled differences

Memory Use One of the concerns for the development of the system are the hidden memory requirements of having many connections with large Bandwidth delay products. Since onion routing has a mesh topology we should understand the memory requirements assuming a distribution of half nodes in Eurpoe and half of the nodes in the United States. The optimal buffer size for a TCP connection is given by:

$$BufferSize = Bandwidth * RTT$$

For Tor, the RTT (Round trip time) between nodes is of up to 300ms (And a back of the envelope expected RTT of 140 ms) . This worst case value was found by empirically doing 'ping' between nodes in PlanetLab, in a connection between Germany and Brazil. Graphing the maximum throughput vs the allocated buffer size results in figure 4. From this figure we can notice that for 'high' throughput nodes (nodes with 10 to 100 Mpbs connectivity) the buffer size requirements to be able to use all the available bandwidth is in the order of 512 KB for the average RTT value of 140 ms. Scaling this value to the number of connections, assuming scaling our network to 8000 hosts means we require 4GB of memory, just network buffers, in the core high throughput nodes.

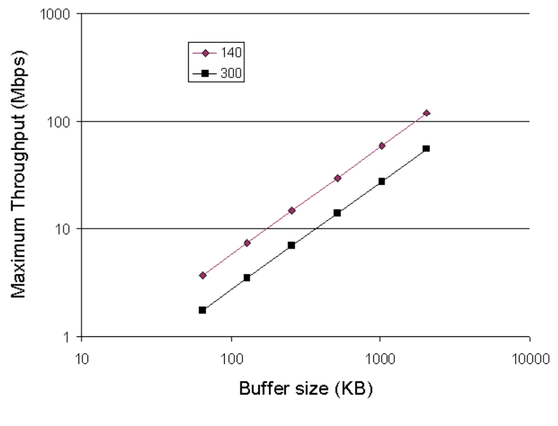

Fig. 4. Maximum Throughput vs Buffers size for Tor's RTT.

Bandwidth Efficiency The change of transporting data streams to transport datagrams can potentially result in a significant increase in the number of messages and bytes transported on the network. We compare the bandwidth requirements of the system with Tor, illustrating the need of multiple cell sizes in our design.

It is estimated that by using 1024 data cells the transport ineficiencies would of the order of 8.9% worse than using Tor if there is one ack per message (we assume that tor TLS data is perfectly packed). If there is one 128 byte for every two 1024 data cells, the inefficiency comared to tor is only of 1.3%. Using 512 size cells the inefficiences are of 30% and 16% in the same cases.

4.2 Measurements

We compared a network of onion routers running Tor and our design.

The two of the systems were located in a high-bandwidth network connection in a mayor US university and one of the routers was located in a commercial DSL line in London, UK.

Fair Bandwidth Allocation By fair bandwidth allocation, we mean that if n high throughput nodes are using the system, the bandwidth allocation will for each node will be close to the B/n where is the total bandwidth available to the node. To test this allocation we developed a multi-threaded client that tries to download an CD-ROM image of a high bandwidth host. Each thread downloads up to 20MB of data or downloads as much as it can for 30 seconds. We tested our design and a similar set of nodes running tor version 0.2.0.9 alpha. The results are summarized in table 2.

	Fast Network	DSL
Tor	Fair up to 3 streams	Fair up to 3 streams
UDP-OR	Fair	Fair

Table 2. Bandwidth allocation fairness

The results are the results of running from 1 to 8 simultaneous downloads. For Tor, the problem is located in the Socks code, not in the network code as similar tests using multiple clients over the same network show again fairness.

Latency We also measured the latency for application data. To do so we developed a multi-threaded FTP client and measured the round trip time between commands and server responses. For latency we tested the latency when the system had no load and when the system was operating at maximum load. The results are summarized in table 3. Load was placed into the system by issuing one or four simultaneous before the latency tests were made.

	Fast Network	DSL
No Load	97 ms	341 ms
Maximum Load (1 thread)	123 ms	5454 ms
Maximum Load (8 threads)	653 ms	6730 ms

Table 3. Effect on latency of load

The results are a little encouraging, for high speed networks. However, when using the high latency low throughput network, we find abysmal delays. Some improvements were achieved by changing some of the parameters of the RED queue at the OR but we wanted to show the worst scenarios seen.

5 Future Work and Conclusions

We have introduced a new design for data transport in onion routing systems. An implementation of the design has given mixed results: while the fairness of network allocation has improved, latency when the system is under load has not changed significantly. Further research is required to determine where in the the system things can be improved.

There are two design variations the the authors will like to test in future systems. The first is the use of multi-path routes and the second is the use of priorities on the routed messages. The use of multi-path routes promises solutions on robustness of the network, improvements on bandwidth allocation and potentially increase the anonymity set of long lived connections. By allowing on-demand path changes long lived connections will no longer have the same 'netflow signature' at entry and exit points. This can potentially eliminate the Internet exchange attack introduced by Murdoch and Zieliński [39], as no one-to-one mapping will exists for the entry and exit flows. We also believe that better bandwidth assignment can be done with the use of multiple paths as the clients have a more fine grained way to distribute the network resources.

Regarding the publicly available implementation, there are many issues that the authors would like to address. In particular we would like to: add identity checking to the circuit creation cells, use DTLS instead of our homegrown link level encryption, test the efficacy of the router level congestion control mechanisms, use an internal TCP stack for clients, and create a transition plan for integration with Tor.

Finally this designs leaves open many questions regarding the anonymity of the users. Does the use of two cell sizes affect significantly the anonymity of users? if so how much? Do bandwidth side-effects can be as problematic for anonymity than latency side effects?

6 Acknowledgments

hidden for peer review.

References

1. M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," *IEEE JOURNAL on Selected Aread in Communications*, May 1998.
2. R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *12th USENIX Security Symposium*, pp. 303–320, USENIX, 2004.
3. J. McLachlan and N. Hopper, "Don't clog the queue: Circuit clogging and mitigation in P2P anonymity schemes," in *Proceedings of Financial Cryptography (FC '08)*, January 2008.
4. J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Transactions on Computer Systems*, vol. 2, pp. 277–288, Nov. 1984.
5. D.Chaum, "Untraceable electronic mail, return addresses and digital pseudonyms," in *Communications of the ACM*, February 1981.
6. D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, pp. 65–75, 1988.
7. G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type iii anonymous remailer protocol," in *2003 Symposium on Security and Privacy*, IEEE, 2003.
8. C. Gülcü and G. Tsudik, "Mixing E-mail with Babel," in *Proceedings of the Network and Distributed Security Symposium - NDSS '96*, pp. 2–16, IEEE, February 1996.
9. A. Inc., "Anonymizer website." http://www.anonymizer.com/consumer/products/anonymous_surfing/, 2007.
10. T. Inc., "Ghostsurf website." http://www.tenebril.com/consumer/ghostsurf/ghostsurf_standard.php, 2007.
11. E. Nakashima, "A story of surveilance." "http://www.washingtonpost.com/wp-dyn/content/article/2007/11/07/AR2007110700006_pf.html", November 2007.
12. EFF, "Att class action." http://www.eff.org/cases/att.
13. R. Singel, "Encrypted e-mail company hushmail spills to feds." http://blog.wired.com/27bstroke6/2007/11/encrypted-e-mai.html, November 2007.
14. L. Thompson, "Hushmail turns out to be anything but." http://www.itnews.com.au/News/65213,hushmail-turns-out-to-be-anything-but.aspx, november 2007.
15. M. J. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, (Washington, DC), November 2002.
16. Z. Brown, "Cebolla: Pragmatic IP Anonymity," in *Proceedings of the 2002 Ottawa Linux Symposium*, June 2002.
17. M. Reiter and A. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security*, vol. 1, June 1998.
18. W. Dai, "Pipenet 1.1." Post to Cypherpunks mailing list, November 1998.
19. P. Boucher, A. Shostack, and I. Goldberg, "Freedom systems 2.0 architecture," white paper, Zero Knowledge Systems, Inc., December 2000.
20. M. Rennhard and B. Plattner, "Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002)*, (Washington, DC, USA), November 2002.
21. M. Rennhard and B. Plattner, "Practical anonymity for the masses with morphmix," 2004.

22. P. Herman, “Jap webpage.” http://anon.inf.tu-dresden.de/index.html, 2006. (last access March 1 2007).
23. B. N. Levine and C. Shields, “Hordes — A Multicast Based Protocol for Anonymity,” *Journal of Computer Security*, vol. 10, no. 3, pp. 213–240, 2002.
24. S. Goel, M. Robson, M. Polte, and E. G. Sirer, “Herbivore: A Scalable and Efficient Protocol for Anonymous Communication,” Tech. Rep. 2003-1890, Cornell University, Ithaca, NY, February 2003.
25. R. Sherwood, B. Bhattacharjee, and A. Srinivasan, “P5: A protocol for scalable anonymous communication,” in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, May 2002.
26. H. Xu, X. Fu, Y. Zhu, R. Bettati, J. Chen, and W. Zhao, “SAS: A scalar anonymous communication system,” in *Proceedings of ICCNMC*, pp. 452–461, 2005.
27. M. Liberatore, “Tor unreliable datagram extension proposal,” 2006. (http://www.torproject.org/svn/trunk/doc/spec/proposals/100-tor-spec-udp.txt).
28. W. Stevens, “Tcp slow start, congestion avoidance,fast retransmit, and fast recovery algorithms.” http://www.rfc-editor.org/rfc/rfc2001.txt, 1997.
29. L. S. Brakmo, S. W. O’Malley, and L. L. Peterson, “TCP vegas: New techniques for congestion detection and avoidance,” in *SIGCOMM*, pp. 24–35, 1994.
30. V. Jacobson, R. Braden, and D. Borman, “Tcp extensions for high performance.” http://www.rfc-editor.org/rfc/rfc1323.txt, May 1992.
31. S. Floyd and K. Fall, “Router mechanisms to support end-to-end congestion control,” 1997.
32. S. Floyd and K. Fall, “Promoting the use of end-to-end congestion control in the Internet,” *IEEE ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999.
33. S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” in *IEEE/ACM Transactions on Networking*, IEEE, August 1993.
34. S. J. Murdoch and G. Danezis, “Low-cost traffic analysis of tor,” in *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, 2005.
35. K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, “Low-resource routing attacks against anonymous systems.” http://www.cs.colorado.edu/department/publications/reports/docs/CU-CS-1025-07.pdf (last access: March 1 2007).
36. R. Dingledine, N. Mathewson, and P. Syverson, “Challenges in deployinig low-latency anonymity.” http://tor.eff.org/svn/trunk/doc/design-paper/challenges.pdf (last access: Nov 28 2006).
37. N. Modadugu and E. Rescorla, “The design and implementation of datagram tls,” 2004.
38. M. Smith, “Antinat socks server website,” 2005. http://antinat.sourceforge.net/.
39. S. J. Murdoch and P. Zieliński, “Sampled traffic analysis by internet-exchange-level adversaries,” in *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)* (N. Borisov and P. Golle, eds.), (Ottawa, Canada), Springer, June 2007.