

# Cone of Silence: A Layered Approach for Network-level Protocol Anonymization

Katherine Deibel and Andrew Petersen and Andrew Schwerin

November 14, 2003

## 1 Introduction

As the commercial use of the Internet has increased, the quantity of information that corporations and governments can and do gather about users has grown astoundingly. Recent events have increased support for enhancing the surveillance capabilities of governments, and the high commercial value of consumer data encourages information sources to amass it in bulk. Given the probability that this data can be mined and sold, we expect the amount of information gathered about Internet users by commercial ventures will continue to increase. In addition, employers and schools often log network traffic for a variety of reasons, and these logs can later be mined for clues about network users' private social and health concerns.

These trends threaten the privacy of all users and disturb those who wish to prevent others from having knowledge of their identities and activities. While end-to-end cryptography ensures the confidentiality of the contents of information exchanges, it does nothing to preserve the anonymity of the participants. It is important to realize that confidentiality and anonymity are separate concerns, and each requires a separate solution. As we have mentioned, simple end-to-end cryptographic solutions suffice for the security properties of confidentiality and authenticity. True anonymity, however, can only be achieved with network support. Without some facilitating agent in the network, it is impossible to route data between two nodes without revealing the identity of the sender, since the very action of sending data to the recipient must reveal the identity of the sender for two-way communication to occur. For the same reason, it is also impossible to avoid revealing, to any observer in the network, that the two nodes were communicating unless some node on the path between them supports anonymization.

In this paper, we discuss our experience designing and analyzing our flexible anonymization network infrastructure Cone of Silence (CoS). CoS allows end users to select varying degrees of anonymization based upon the threat model that best applies, the resources available to their adversaries, and the increased latency and decreased

throughput they are willing to tolerate in exchange for anonymity. To provide increased efficiency and flexibility, CoS is designed to be deployed on a global, spatially distributed network like PlanetLab [14], rather than on peer-to-peer networks like most anonymization services available today. This does not preclude running CoS on other platforms (including peer-to-peer networks). However, when developing CoS, our goal was not only to create an anonymization service, but also to build the framework for a test bed. Our anonymization system provides the infrastructure needed to allow interested parties to experiment with and reason about the costs and benefits of different degrees and forms of anonymization, and PlanetLab is an ideal platform for this purpose.

CoS is intended to be the network-level service component of a full anonymization service. Therefore, it does not scrub application-level protocols for sensitive information (e.g. javascript fragments that request the client's IP address). We believe that application-level anonymization is best achieved and controlled at the end nodes, when information is transferred via the application-level protocol. This allows the user fine control over what is filtered. Scrubbing may be performed privacy-respecting application implementations, sandboxed applications that are not given access to protected information, or protocol-aware proxies running on the user's local machine that scrub outgoing application-level traffic. Several software packages that perform filtering are listed on the EPIC privacy page [8], including the Proxomitron [15] and the LPWA [12].

## 2 Threat Model

One must consider the degree of anonymity that a principal participating in a system desires. More secure systems incur higher performance costs than less secure systems, so the user must balance her need for anonymity with her desire for performance. As an example, over insecure networks, the degree of expected confidentiality is measured in terms of the amount of effort an adversary must expend to learn the contents of the communica-

tion. Therefore, participants in confidential communications choose cryptographic algorithms and key sizes based upon the expected computational resources of their adversary, its desire for the information, and the likelihood that adversary will resort to obtaining the information without breaking the cryptography (i.e., by breaking kneecaps instead).

Similarly, one can imagine different users of an anonymization system being concerned about adversaries with varying resources and desire to learn the identity of the user. Our system allows users to tune the behavior of the anonymization system based upon the capabilities of potential adversaries. Specifically, users must decide whether or not adversaries might have control over the following resources:

**Information Servers (IS Threat)** If an adversary provides a service, such as a web site, that an anonymization user wishes to use or has the ability to take control of that service, then the user must hide her identity from the server.

**Originating or Intervening ISP (ISP Threat)** If the adversary is also the user's network service provider or controls links on any possible path between the user and her intended destination, then the user must be able to hide the ultimate destination of her communications and their contents.

**Nodes in the Anonymization System (AS Threat)**

If the anonymization system works and a user's identity or activities are considered particularly valuable, then it is reasonable to expect that one or more nodes in the anonymization network may become compromised. This is especially true of an anonymization system whose components reside within different administrative domains, as might be the case in a system set up to allow mutually distrusting organizations to provide an anonymity service that leverages that distrust.

An approach to anonymization that can service users whose threat models place varying importance on these different kinds of threats must be tunable to the desires of the user. We believe that a system that allows software running on the user's own machine the power to combine a set of anonymization techniques can provide the required flexibility. As we discuss the related work, we hope to emphasize the components of anonymization that recur throughout the corpus.

### 3 Related Work

Several other systems have been developed that use centralized or distributed proxies to anonymize traf-

fic. However, to our knowledge, CoS is the only distributed system that allows the user to select the level of anonymization required.

#### 3.1 Absolute Trust Systems

In an absolute trust system, the user knows that the system is working to protect his or her identity. Because of this assumption, only the IS and ISP threats apply, and a single proxy node or set of single proxy nodes is sufficient to serve the client's needs. Assuming the client scrubs messages at the application level, information servers only know that data forwarded through the proxy are arriving from an anonymization service. Furthermore, if communication between the client and the anonymization service is also encrypted, then any intervening ISPs know the client is using the service but do not have access to the content being transmitted, or the identity of the second party of the communication.

As only one node is required in a single proxy system, absolute trust systems are often centralized. Unfortunately, this approach leads to bottleneck issues and provides a single point of attack. While load-balancing and server farms help with the first problem, centralized systems remain vulnerable to aggressive adversaries. This is so because, for any communications, the proxy knows both the origin and destination parties. This makes proxies lucrative targets for subversion.

The Anonymizer [2] is a centralized, commercial system. Varying levels of anonymization may be purchased. A client gains anonymity by making HTTP or other application-level requests through a proxy hosted on Anonymizer's machines to neutralize the IS threat. Optionally, requests may be tunneled via SSH for protection from nosy ISPs. For added security, all HTTP requests are filtered for malicious, identity-seeking code. However, in addition to the normal drawbacks of a centralized system, Anonymizer logs its clients' requests, which making it an unusually lucrative target for subversion.

In discussing SafeWeb [20], we must use the past tense. Although the company still exists, it has shifted focus and primarily handles industrial privacy needs. Before this shift, SafeWeb worked much the same as Anonymizer. However, large security problems were discovered (see [13]), resulting in the termination of this anonymization service. Chief among these problems was a security hole that allowed any web site to learn the entire contents of a browser's cookie cache. As discussed in [13], the hole stemmed from a flaw in the design philosophy of the anonymizer: only actions known to compromise anonymity were blocked, rather than blocking all actions not known to be anonymity-preserving. Though the decisions made by SafeWeb were motivated by a vari-

ety of oft-conflicting goals, and not out of malice, the consequence reminds us that the physician’s motto applies to security as well: “First, do no harm.”

### 3.2 Untrusted Systems

Untrusted systems acknowledge that nodes may become compromised, resulting in an AS threat. Remarkably, even if some nodes of the anonymization system are compromised, a system may still provide reliable anonymization. By forwarding data through multiple nodes that do not all collude to share information, the chance that a user’s identity will be revealed can be reduced to an arbitrarily small value. This approach avoids the single point of attack of single proxy systems, and presents opportunities for scaling because of its distributed nature. For example, if a user trusts multiple nodes in the system, she may balance her load across multiple nodes. In general, untrusted systems are implemented either as a dedicated set of (potentially distributed) anonymization nodes or in a peer-to-peer fashion.

In Onion Routing [10, 17], communications are wrapped in an *onion*, a layered data structure that contains encrypted routing information. At each node, a single layer is decrypted, and the remaining onion is forwarded to the indicated address. Ultimately, the request is sent to the desired recipient, and any response is re-wrapped, in a similar fashion, on the return trip. Because the encryption used at each hop changes the appearance of the packets, eavesdroppers are foiled, much as in Chaum mixes [5]. In fact, adversaries can only associate the identity of the client and the content of the request if all nodes on the path collude.

An attack known as the predecessor attack, which analyzes the reuse of paths in the anonymization network to identify users, may weaken anonymity assurances in this system. The specific details of this attack and its effect on several anonymization systems (including Onion Routing and Crowds) is discussed in [23]. It makes the assumption that an adversary may monitor, but not tamper with, traffic among all or at least many nodes participating in the onion overlay. It observes the arrival and departure of packets in time, and attempts to correlate them based upon timing in order to reconstruct entire onion routes. However, the predecessor attack is only viable in Onion Routing over long-lived flows, and we believe it could be made even less viable by allowing the route taken by data in the anonymization network to change during the course of a flow – a technical challenge, but not an impossibility.

Crowds [18] employs the users of the system as forwarding nodes in the anonymization network. Upon joining a *crowd*, the client contacts a *blender* to exchange

keys and obtain a random path through the crowd. Messages are forwarded on this path, and the last member on the path services the request. Until the crowd is ordered to perform a regular path reformation, all subsequent messages use the same path as the first. Because of these long-lived paths, the predecessor attack is potentially quite potent in Crowds [23]. A more serious limitation is the bottleneck and attack point found at the centralized blender. In addition, the overall bandwidth available to a crowd member is the minimum bandwidth available to any member on its anonymization path, which sharply degrades performance.

AP3 [3] is another peer-to-peer system. It is built on top of Pastry [19], a peer-to-peer distributed hash table and overlay routing mechanism. Requests are routed across the system randomly until a weighted coin flip causes a node to forward the request to its intended destination. While this method avoids the blender overhead of Crowds, it still suffers from the bandwidth limitations of a peer-to-peer network. Furthermore, the Pastry system itself has numerous security vulnerabilities that endanger the anonymity of users. No application of traditional symmetric or asymmetric cryptography may be used, since the route followed by data through the Pastry network is not known by the sender. As a result, every AP3 node knows the identity of the recipient of a communication. This problem is compounded by AP3’s lack of admission control, which allows a single host to behave as though it were an arbitrary number of AP3 nodes. This allows an adversary to inflate the probability of seeing a message by simply agreeing to handle the work of a higher number of AP3 Pastry nodes.

## 4 CoS from a User’s Perspective

Like most networking services, a user of CoS should not need to know how it functions on the network level. The only information the user must be able to provide is the degree of anonymization she desires. Figure 1 shows four example scenarios that may be encountered when using the CoS system. We discuss each scenario in turn.

Client A, or Alice, in the figure is only concerned about the IS threat. To hide her identity from the server, she uses one CoS node to forward her requests and does not perform encryption. The server she contacts believes CoS-1 or CoS-2 is making the request. Note that while Alice is using CoS like a centralized, absolute trust system, she may choose her proxy from a pool of available nodes. This allows her to improve the performance of CoS (by selecting a lightly loaded anonymization node) and further obscures her identity, as multiple CoS nodes may be used to make requests.

Like Alice, Client B, or Bob, also trusts CoS. However,

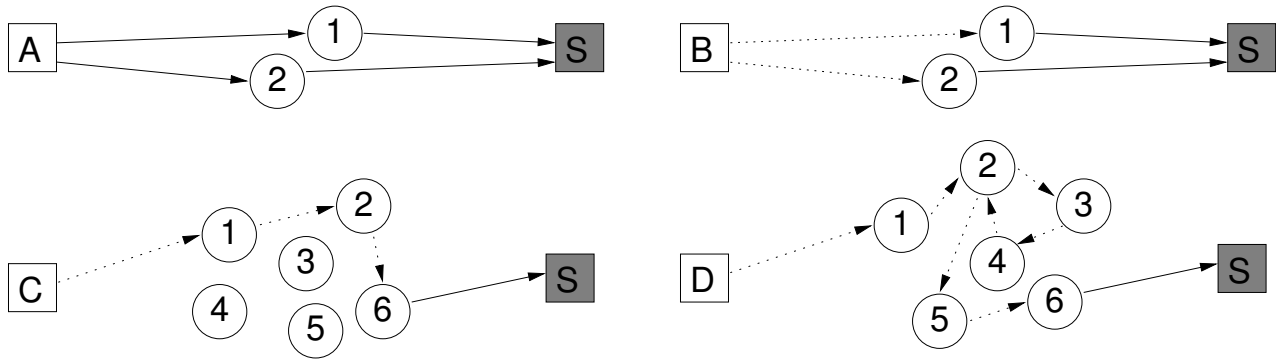


Figure 1: Scenarios for using CoS. White squares represent clients, circles are CoS nodes, and shaded squares are servers. Solid and dotted lines indicate unencrypted and encrypted communication, respectively.

he does not trust his ISP. To address the ISP threat, Bob encrypts his requests to CoS, making it computationally difficult for the ISP to determine the destination of his requests. Nodes CoS-1 and CoS-2 function much like they did in the previous scenario, except they must now decrypt messages from Bob and encrypt messages to him. The extra computational overhead (and hence, latency) is the price he pays for his greater anonymization needs.

Client C, Carol, is not convinced that CoS is worthy of her trust. She believes that either some nodes may be compromised or that adversaries may be eavesdropping, both of which constitute an AS threat. To combat this, she chooses a random path consisting of three CoS nodes to forward her requests. Using the onion routing principle, her requests are wrapped in three layers of encryption, and one layer is removed at each hop. On the return trip, the response is encrypted at each hop, and Carol must decrypt each layer of the onion when she receives her response. The overhead of the multiple encryptions and decryptions and the increased path length both affect performance.

Carol uses a relatively short path through CoS. While she is concerned about the AS threat, she is not too worried about the system’s security. In contrast, Client D, or Dave, has very low confidence in the security of the system. He has chosen the seven hop path 1 – 2 – 3 – 4 – 2 – 5 – 6 in order to make it very difficult for adversaries to identify him. This anonymity guarantee comes with a significant performance penalty. Note, in addition, that CoS-2 is used twice on the path Dave chose. Currently, CoS allows such “loops,” though it may weaken anonymity guarantees.

These four scenarios describe the range of anonymization that CoS provides. A few other options are available, however. For example, not every link between anonymization nodes need be encrypted if the user does not fear from eavesdroppers. This increases performance while maintaining the requirement that either all

anonymization nodes be compromised or an eavesdropper be present in the system to break the user’s anonymity guarantee. Another option available to CoS users is anonymization through other services. The user may route packets through CoS to another anonymization service which may either service the request or forward a payload back to CoS.

One other user aspect we have not yet mentioned is path reuse. The longer a path session is used by a client, the more vulnerable it becomes to various attacks. To insure anonymity, it is best for the user to switch between various paths and to kill off old path sessions and establish new ones fairly frequently. This is true for a wide range of users seeking anonymity, including both Alice and Dave. In the current implementation, the user has some control over the frequency of path reformation, but we expect to implement a requirement for regular path reformation in later versions of CoS.

## 5 The CoS Protocol

CoS is a source-routed, unreliable datagram protocol. It is session-oriented, and routers must store a small amount of per-session state. We believe the utility of CoS lies in using it to test and compare a wide variety of anonymization techniques. As such, no strong distinction exists within the overlay between routing or end nodes, and no rules govern which nodes may inject, receive, or route packets. Therefore, any node can operate as a router or client, and CoS is deployable on platforms ranging from a dedicated infrastructure of routers to a peer-to-peer network. This design was chosen for its simplicity and flexibility.

Though all nodes are capable of performing all behaviors defined in the CoS protocol, within a particular routing session nodes often take one of several distinct roles:

**ingress nodes** Ingress nodes in a session are the nodes at which the IP data is first tunneled into CoS datagrams. In the most common operational profile we envision, such nodes are end-user clients. The ingress node is the first in a source route for a CoS datagram.

**egress nodes** IP data emerges from the CoS tunnel and is routed to its final destination at an egress node. The egress node is the last in a source route for a CoS datagram.

**forwarding nodes** Forwarding nodes lie between the ingress and egress nodes in the source route of a CoS datagram.

IP traffic through CoS achieves its anonymity by tunneling IP packets inside of CoS datagrams. (In addition, source identifying information must have been scrubbed at the network and application layers of the client host.) When an IP packet forwarded through CoS reaches the last node in its source route, that egress node acts as a network address translator [7] for the flow. This fact requires that the egress node remain constant in the CoS route for the duration of any connection-oriented IP flow. However, any other node in the route, including the ingress node, may be changed. Hence, it is possible to use the anonymization network to support client mobility at little or no extra cost above the cost of anonymization. In principle, it should also be possible to extend CoS to be resilient to failures of nodes along a given source route by choosing an appropriate new source route [1] The current implementation does not support this, however, and implementing it may present interesting technical challenges.

## 5.1 Forwards and Backwards

All datagrams moving through the CoS overlay network have a direction. Either they originate at the client node that initiated the communication, in which case they are said to be moving forwards, or they are a reply to a forwards-moving datagram. In the latter case, we say the packets are moving backwards.

Forward-moving (forwarding) datagrams may be source routed, because the initiator of a communication is expected to be able to identify both itself and the destination of the communication. Backward-moving (backwarding) datagrams, because they are moving toward the anonymized initiator of the communication process, must be routed by the network to preserve anonymity. To enable this, the network must store, at every node in the forward route, the identity of the node immediately preceding it in the route. Backwarding packets are then routed

along the exact reverse path of their corresponding forwarding packets.

The metaphor of an onion applies well to the structure of forwarding datagrams. The sender of a source-routed forwarding datagram builds a message and wraps it in a forwarding message to the last node in the source route (the egress node). It then wraps that message in a forwarding message to the node immediately preceding the egress node, and this process continues until a forwarding message to the first node in the source route has been created. This message is sent, and each receiver peels away the outermost layer of the datagram, gleans forwarding information from it, and sends the remainder of the datagram to the next node in the route. Hence, the last node in the overlay to receive the datagram knows the receiver but not the initiator, as this information was peeled away some number of hops previous [10].

Each node keeps a forwarding table that contains information on all the active sessions routed through it. In essence, each entry in the forwarding table defines a “circuit” between the previous node, the current node, and the next node. As such, each entry contains the address of the previous node and the session identification number affiliated with it, the address of the next node and its associated session number, and a cryptography engine for decoding and encoding the message. Either session ID and address is sufficient to lookup the entry, so between the data stored in a CoS packet and the state stored at the node when a new session is initiated, messages can be correctly routed to and from the client.

For the multiple layers to be effective, the initiator of the communications must encrypt each successive layer of the onion so only the intended receiver of that layer can decrypt it. Either public or symmetric key cryptography may be used, but in the case of symmetric key cryptography, per-session cryptographic keys must be established. This added burden is acceptable because symmetric key cryptographic algorithms are generally several orders of magnitude faster at encrypting and decrypting than public key algorithms that provide equivalent expected security [22]. Session establishment payloads, which we describe later, allow for the establishment of such symmetric session keys. Session establishment messages must themselves be protected, and assuming that session establishment is infrequent compared to data forwarding, public key cryptography seems an appropriate choice for this.

Another advantage of symmetric session keys is that new symmetric keys are often trivial to generate, while new asymmetric keys are computationally intensive to generate. Since a re-used key is an identifying piece of information, new keys must be generated frequently by end clients. This suggests a deployment of CoS in which any transaction uses public key cryptography to

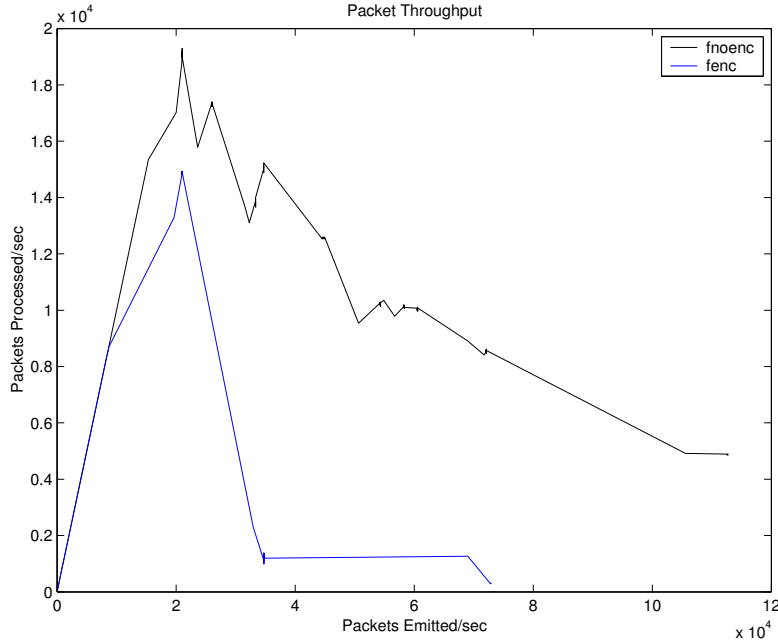


Figure 2: Throughput of forwarding packets at a single CoS node.

send single-packet forwarding payloads and a symmetric key to be used for backwarding the response. Another option, which we do not yet support, would be to send, in a single message, a session key encrypted with a public key and a forwarding payload encrypted with the symmetric session key, much as PGP messages are encrypted [zimmermann95pgp].

## 5.2 Session Establishment

At each node in a route, CoS sessions are identified by a session identifier. When a node detects a new session identifier in a forwarding or session-establishing message, it chooses its own session identifier and reports it to the next hop for use in backwarding. Hence, every node can uniquely identify sessions without global session state.

In the absence of admission control (and CoS currently provides none), simply processing a message with a new session id leads to the storage of session information. However, if a symmetric session key is to be established for use in onion-style forwarding and backwarding as described in the previous section, a session establishment message containing key setup information must be forwarded. The key setup information consists simply of the cryptographic algorithm and mode to be used (currently, Blowfish in CBC mode is supported), and the key data. For security, symmetric key establishment should only be permitted using an already-known symmetric key or a public key, but at present this is not enforced. Al-

ternatively, a Diffie-Helman [6] key exchange could be performed, but since public key cryptography would be required to guarantee the identity of the router node, it seems superfluous. It is in the ingress node's interest to select a truly random key, so it need not use a complicated agreement protocol with the router nodes.

## 5.3 Egress

Egress nodes behave as network address translators, causing IP traffic emerging from the CoS tunnel to appear to originate from the egress node. This requires them to map transport-protocol state, such as port numbers, to previous hop session state. Such state must be periodically flushed from the egress node's forwarding tables, since network protocols do not always terminate in a manner identifiable by the egress node. However, the egress node is not an actual participant in the transport protocol connection. At its simplest, it is a packet rewriter. A more sophisticated implementation might use information marked in the CoS datagram it receives to perform congestion notification [16] or other operations, but it need not do so.

Unfortunately, at the current time, CoS does not provide ingress and egress functionality. This is a technical challenge, but we do not foresee any large problems implementing either. The ingress router will intercept IP packets sent to a virtual network interface, wrap them as specified by the user, and then place them on the network. We

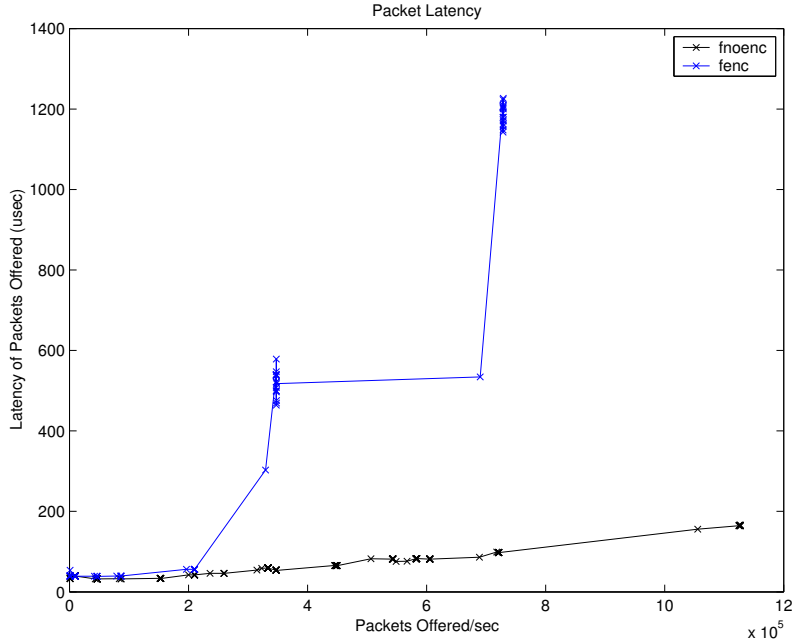


Figure 3: Latency of forwarding packets at a single CoS node.

would like for the ingress router to be intelligent enough to identify the source of the packets, so for example, the user could specify that her ssh client not be anonymized but that her web traffic use a three hop route through CoS. On the egress side, the router must correctly identify the transport type of the packet and create the correct type of socket. We would like it to understand the various transports so that TCP options are handled appropriately and ICMP messages are backwarded to the ingress router or handled by the CoS system, as appropriate. An excellent implementation of ingress and egress would allow, for example, the end host to use path MTU discovery algorithms without explicitly interacting with the CoS system and to receive the ECN notifications that packets carrying the CoS datagrams receive.

## 6 Performance

In evaluating CoS, two kinds of measurement are of interest: those assessing the performance of a single node in the CoS overlay, and those assessing the performance of the overlay as a whole. For a single node, we are interested in the overall throughput and latency in packets and bytes with respect to the offered load in terms of packets offered per second, sessions offered per second and bytes offered per second. For the whole system, we are interested in the latency required to complete a short anonymous transaction, such as an HTTP GET, and a long

transaction, such as large file transfer, as the length of the anonymizing path increases, as the strength of the cryptography increases, and as the cross traffic load increases.

At this time, we have only had the opportunity to focus on the single node measurements. Representative of the measurements we have taken are the latency and throughput of packet forwarding, with and without layered encryption, at a server node. Related information that would be interesting to collect would be the computational cost at the client node for wrapping up a packet in  $n$  layers of cryptography as  $n$  increased, unwrapping an  $n$ -layered response packet, and generating new session keys for a route of length  $n$ . Similar results are gained by taking the same measurements on backwarding datagrams, as is to be expected since the operations are symmetric in terms of computational requirements.

In all of these measurements, the cryptographic algorithm used was Bruce Schneier's Blowfish [21] symmetric key algorithm, with 128-bit session keys. CoS operates the cipher in cipher-block chaining mode (see [22] for details), with every CoS datagram being a unique chain. This eliminates insertion and substitution attacks, while still allowing processing to proceed when datagrams do not arrive, or arrive out of order.

To measure our system, we utilized a testbed of six 1.7Ghz single Pentium processor machines each with approximately 1Gb of memory. Aside from the traffic we created, no other traffic was present on this testbed. As in-

indicated in figure 4, one computer was designated as a CoS server, another served only as a receiver, and the other four were set to emit packets to the server at a fixed rate.

Each node had an appropriate measurement device embedded in its code. The server measured the time (in microseconds) needed to process a CoS packet. Each emitter recorded the number of packets it sent to the server, and the receiver counted the number of packets it received from the server. In all the machines, the data was gathered at ten second granularity and saved to a file every minute. Although the machines' clocks were not synchronized, fixed durations of silence interrupted by isolated, single packet transmissions made it possible to keep the machines in synch with one other. This method was especially useful when gathering measurements that required that a route be established (encrypted forwarding messages and backwarding messages of all sorts, for example). By creating a pause between route establishment traffic and the messages we wished to measure, it became easy to filter out route establishment packets from the interesting data.

From 2, we see that the computational cost of encrypting small packets does not affect throughput until the packet rate reaches approximately 10 kpackets per second. However, the peak drop-free throughput in packets when encryption is used is only about half of that when encryption is not used. An interesting question that we did not have the opportunity to ask was, how is throughput affected as the offered load in terms of bytes increases?

From 3, we see that as offered load increases, the per-packet latency increases much more rapidly for layer-encrypted packets than for unencrypted packets. However, since latencies are almost the same per packet up until the offered load when both encrypted and unencrypted throughputs drop drastically, we see that for small packets under nominal load, symmetric cryptography has a minimal affect on latency. Above 20,000 offered packets per second, it is probable that the cryptographic algorithm must compete with the network queue management functions of the operating system for the processor, causing the observed spike in per-packet latency. A question to ask if a user is concerned about the responsiveness of this system will be, "How does per-byte-offered latency change with packet size?" That is, will latency improve with larger packets because of less per-packet processing, or will it remain about the same, dominated by the per-byte cost of cryptography?

From a quick measurement taken of session establishment costs, minus the necessary and computationally overwhelming public key cryptography, we have found that session key establishment is necessarily a limiting factor on the rate at which new routes may be established and old routes retired. A detailed study on how quickly

these changes may be made is the subject of future work.

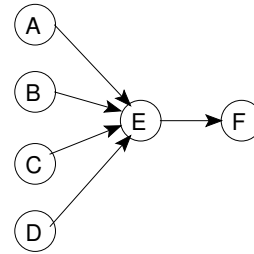


Figure 4: Topology for analyzing the latency and throughput of a CoS node.

## 7 Future Work and Conclusions

The CoS system remains in its infancy. In addition to several technical challenges (for example, improving the system's robustness and correctly egressing IP packets), several research questions remain unanswered.

We have yet to identify the appropriate mechanism for the distribution of public keys. Existing public key distribution models, such as those used in PGP [25] or PEM [11] provide a basis for such work but are not complete solutions. Onion Routing, which faces a similar key distribution problem, does not offer a solution [10]. One possibility would be to use random cross traffic to deliver this information. With more traffic on an anonymization network, a person's own traffic is more obscured. CoS nodes could produce their own random traffic and embed key into to other nodes where clients could access it. However, cross traffic needs to be controlled enough as to not overburden the system.

It would be valuable to fully deploy the system and hence, to learn what aspects of its performance are desirable and which need improvement. The microbenchmarks we have do not provide sufficient information to understand how the system will be used or how it reacts to being used in creative ways. Particularly, as evidenced in the effects encryption has on the throughput and latency of forwarding packets, formal route establishments needs investigating. Since public key encryption is needed to make route establishment secure, the latency and throughput of route establishments will likely suffer in performance. Rapid changing of sessions inside the network will likely not be feasible, but constant reuse of the same session over time weakens one's anonymity. To study this inherent tradeoff, more thorough analysis is needed on several fronts. First, the degradation of anonymity over time should be investigated more. Also, further measurements of route establishment performance in terms of key size,

number of hops in the path, how many sessions are being handled by a router, etc. should be gathered. Together, this data would provide a means of determining how often to establish routes based upon one's time and anonymity needs.

It would also be interesting to perform an empirical comparison of different anonymization techniques using the CoS protocol and implementation as a foundation. The CoS protocol is flexible enough to emulate the existing anonymization systems, and using it as the basis for implementations of other techniques eliminates performance differences due to implementation details.

Much of CoS's overlay functionality is not related to anonymization so much as it is related to overlay source routing. This suggests a fair amount of implementation detail in CoS is redundant when compared to implementation details in other overlay networks – especially RON [1] and OpenVPN [24]. It would be interesting to abstract the components of these systems into modular pieces that could be mixed and matched to achieve desired functionality. In the process, the production of redundant code would be reduced. Braden et al. propose this in [4], using the name "Role-based Architecture." Overlays such as CoS and RON could provide a fertile testbed for studying and developing role abstractions.

In conclusion, upon deployment, CoS could provide a platform for delivering an anonymization service and for facilitating further experimentation and research in the areas of efficient, secure anonymization and in overlay networks in general. We believe that this paper hints at the research that could yet be performed in these areas.

## References

- [1] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient overlay networks. *Proc. 18th ACM SOSP*, oct 2001.
- [2] The Anonymizer. <http://www.anonymizer.com>.
- [3] C. Bornstein, G. Oberoi, and C. Reis. Anonymization of network usage through peer-to-peer proxies. Technical report, Rice University, 2001. Available: <http://www.owlnet.rice.edu/creis/ap3/ap3.pdf>.
- [4] Robert Braden, Ted Faber, and Mark Handley. From protocol stack to protocol heap – role-based architecture. *Proceedings of the First Annual HotNets Workshop*, October 2002.
- [5] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.
- [6] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.
- [7] K. Egevang and P. Francis. The ip network address translator (nat). *RFC 1631*, May 1994.
- [8] EPIC online guide to practical privacy tools. <http://www.epic.org/privacy/tools.html>.
- [9] E. Felten and M. Schneider. Timing attacks on web privacy. In *ACM Conference on Computer and Communications Security*, pages 25–32, 2000.
- [10] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM (USA)*, 42(2):39–41, 1999.
- [11] S. Kent. Privacy enhancement for internet electronic mail: Part II: Certificate-based key management. *Internet Request for Comments*, (1422), 1993.
- [12] Lucent personal web assistant. [http://www.bell-labs.com/project/lpwa/proxy\\_index.html](http://www.bell-labs.com/project/lpwa/proxy_index.html).
- [13] D. Martin and A. Schulman. Deanonimizing users of the safeweb anonymizing service. Technical Report 2002-003, Boston University, November 2002.
- [14] L. Peterson, D. Culler, and T. Anderson. Planet-Lab: A testbed for developing and deploying network services. White paper, University of Washington, June 2002. Available: <http://www.planet-lab.org/pubs/vision.pdf>.
- [15] Proxomitron. <http://www.spamblocked.com/proxomitron>.
- [16] K.K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN) to IP. *RFC 3168*, September 2001.
- [17] M. Reed, P. Syverson, and D. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4), 1998.
- [18] M. Reiter and A. Rubin. Crowds: anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
- [19] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–??, 2001.
- [20] SafeWeb. <http://www.safeweb.com>.
- [21] B. Schneier. The blowfish encryption algorithm. *Dr. Dobbs's Journal*, 19(4):38–40, April 1994.

- [22] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms and Source Code in C*. John Wiley & Sons, New York, NY, second edition, 1996.
- [23] M. Wright, M. Adler, B. Levine, and C. Shields. An analysis of the degradation of anonymous protocols, 2001.
- [24] James Yonan. Openvpn home page. URL: <http://openvpn.sourceforge.net/>, 2002.
- [25] P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, Boston, MA, 1995.