Branching Messaging for Anonymous Communication

Isaac Sheff
SURF
Undergraduate Majoring in Computer Science
California Institute of Technology
isheff@caltech.edu

Tracey Ho
Mentor
Assistant Professor of Electrical Engineering and Computer Science
California Institute of Technology
tho@caltech.edu

***Abstract*—We consider anonymous communication between pairs of nodes in the presence of an adversary who can observe all network traffic. Existing schemes involve partially trusted central servers, a large amount of cover traffic, or high latency. We propose a scheme which improves upon the trade-off, in which messages require only logarithmic time to deliver, as well as computational time to send. In this scheme, any node receiving a message applies a private decryption key to discover content, or instructions to forward the message to zero or more other nodes. Furthermore, when a node applies its decryption key to a message not encrypted with that node's encryption key, the result is indistinguishable from a message with forwarding instructions. A sender wraps a message in an onion route of logarithmic length, branching (forwarding to multiple nodes) at random points, resulting in a tree of forwarded messages. Nodes wait to initiate messages of their own until they have received a message, so it is indistinguishable whether they are sending or forwarding. Through analysis and simulation, we show this system preserves a high degree of sender and receiver anonymity, as well as unlinkability between communicating pairs.**

I. Introduction

Encryption has made great strides in hiding the contents of transmitted data. Encryption alone, however, cannot conceal all information from unwanted third parties. Most notably, encryption alone cannot hide the identity of communicating parties. Beyond merely encrypted communication, anonymous communication discloses neither the creator of message content nor its recipient.

Intuitively, the idea behind Branching Messaging is that if a set of nodes are communicating on a fairly well connected network (most nodes can communicate with most other nodes), one node can hide the recipient of a message by sending it along a multi-stop route to its destination (several other nodes forward the message on its way). What distinguishes Branching Messaging from other systems is that each node that receives a message may send several, including forwarding the message's content. Due to the encryption and message structure used, nodes themselves are usually not aware of which, if any, of the messages they're sending are forwarded content or mere cover traffic. An observer, even one with access to all traffic on the network, could try to keep track of all possible recipients of a message, but as time goes on, that set grows in size exponentially. Nodes can hide when and if they are themselves sending content by sending it as a new message only when they've just received a message. If all traffic is encrypted, an observer cannot tell which, if any of the messages sent contains new content.

A. Previous Work

In principle, Branching Messaging builds upon Chaum's 1981 idea of onion-routing with *mixes* [1]. A *mix* is a node with a public encryption key and a private decryption key. Other nodes can encrypt a message and a forwarding address to a mix, and the mix would decrypt them, and send the message on its way. If the forwarding address is another mix, and the message itself is an encrypted message-address pair, a chain of forwarding procedures is produced, called an onion-route. Each mix knows only where it received the message from, and where it sent it to, but not the original sender or eventual recipient. Unfortunately, for an attacker which sees all traffic on the network, the full route of the content sent can be traced. To avoid this, protocols such as Rackoff and Simon's "Rapid Mixing" rely one mixes receiving multiple messages before sending, so which message has which origin cannot be traced [2]. These fundamentally rely on high traffic density (messages sent per node per unit time) relative to latency to function.

Beimel and Dolev's "Busses" solution, on the other hand, has no reliance on any kind of statistical traffic assumptions to preserve anonymity [3]. Instead, all traffic is onion-routed along pre-determined routes, which require a capacity sufficient to hold all messages likely to be sent at any time, and the ability for all traffic to reach all destinations via some combination of routes. The most reliable such solution requires $\mathcal{O}\left(n^2\right)$ information (enough for a message from each node to each other node) to be transmitted each time step, and to pass over n time steps to each of the n nodes. Compromise solutions feature more linear computational complexity, and $\mathcal{O}\left(n\right)$ latency between message sending and receipt.

Kannan, Ray, and Iyengar provide a solution of similarly linear computational complexity and latency using Randomized Message Forwarding [4].

Other solutions, such as Chaum's Dining Cryptographers Problem, rely on multi-party communication and evaluation to hide senders and receivers [5]. These methods may work

well in systems with multicasting or broadcast, but less so on node-to-node networks.

B. Our Contribution

To explore the middle ground between statistically secure protocols involving high-traffic mixing, and absolutely secure, if slow protocols such as Randomized Message Forwarding and Busses, we propose the idea of Branching Messaging. The fundamental idea is that a message may contain instructions to forward its contents not only to one, but to multiple recipients. This builds upon the idea of a mix creating an onion route to create a route that branches, an encrypted tree. Multiple messages forwarded by the same node in a time step add to the anonymity (one cannot tell which forwarded message was which), as do the messages themselves, branching out to multiple destinations. In principle, the idea of branching messaging is very broad, and can include constructs similar to pre-structured multicast trees, but in order to keep message overhead and computational complexity low, it is easiest to create messages that "branch out" at random. The result is a system which can provide logarithmic (in the number of nodes) latency and computational complexity (for each node), as well as low traffic density (logarithmic in the number of nodes).

II. The System

A. The Graph

Communication takes place between the n nodes on a graph. The nodes are fairly well connected, which is to say most nodes are capable of communicating with most other nodes. For the statistics presented hereafter, the graph shall be assumed to be well-connected, but this is not strictly necessary for the protocol presented to function.

B. Time

All nodes can measure time the same way. As part of the communication protocol, nodes agree on the length of a time step.

C. Desire to Communicate

Each time step, each node has a probability f of generating a piece of content meant for another node (chosen in effect at random).

Nodes want this content to reach the node for which it is meant in as few time steps as possible, but will not do anything to compromise their anonymity. No network observer or intermediary node should be able to tell if and when content is first sent, or who receives it.

Furthermore, nodes do not want to send more traffic, or perform more computation, than necessary. They desire to minimize the use of these resources.

D. Messages

When nodes transmit data to other nodes the transmission is called a *message*. For simplicity, nodes agree on a standard size for all messages that is large enough to encapsulate any content and small enough to be transmitted in one time step between any pair of nodes that can communicate.

E. Identification, Encryption and Transmission

Assume all nodes to have unique and succinct ($\mathcal{O}(\log n)$) identifiers, known to all other nodes.

Much like Beimel and Dolev's requirements for semantic security in "Busses," nodes have semantic security [3]. This means that all nodes know the public encryption keys of all other nodes, and each node keeps secret its corresponding private decryption key. If two distinct messages are encrypted with a public key, no node but the one possessing the corresponding private key can tell which encrypted message has which plaintext in polynomial time (in the number of nodes). Note that this means that if a message is encrypted with one node's encryption key, and then another node's decryption key is applied, the results are effectively random, as they cannot be predicted in polynomial time without knowing the original message and both keys.

All channels of communication, which is to say all information transmitted from one node to another, are assumed to be encrypted. If a node sends the same message to two different destinations, because each is being sent over a channel encrypted with a different key, the resulting transmissions will be indistinguishable to an observer from the case of two entirely different messages being transmitted.

III. Threats to Anonymity

A. Terminology

Using Pfitzmann and Hansen's terminology [6] , an *attacker* is an entity seeking to learn or interfere with the nature of communication in a system. The attacker may perceive a number of things:

- Given a message, a *sender set* is the set of nodes from whence the message could have originated, and the *recipient set* is the set of nodes for which the message content, if it has any, could be destined.
- *Unlinkability* is the quality of a set of items (such as nodes and messages) indicating no change in their relationship after the attacker has made an observation than before.
- *Sender anonymity* is the property of a communication system in which the attacker (which is granted some set of powers determined by the system) is finds messages and the nodes that sent them to be unlinkable.
- *Recipient anonymity* is the property of a communication system in which the attacker (which is granted some set of powers determined by the system) is finds messages and the nodes that receive their content to be unlinkable.
- *Relationship anonymity* means that the sender and recipient of a piece of content are unlinkable.

B. Attackers

An attacker is an entity including 0 or more nodes which seeks any of the following:

- *observe:* The attacker wishes to know who is sending content at any given time (or for any given message),

who is receiving, and the relationships between the two (which nodes are "talking" to which).
- *eavesdrop:* The attacker wishes to know the content being generated and/or received by nodes outside the attacker.
- *interrupt:* The attacker wishes to prevent communication between some pair of nodes outside the attacker.
- *forge:* The attacker wishes to convey content, appearing to be a node not included in the attacker (this is easily prevented by digital signatures), or in violation of some kind of rules or restrictions.

Attackers are defined by the set of nodes they contain (including those nodes' positions on the graph), as well as the following characteristics (any given attacker has some subset):

- *passive:* The attacker obeys all rules of the system as far as interaction with other elements of the system are concerned, but may, for example, share information outside of the system.
- *active:* The attacker may break rules of the system, such as failing to forward messages the system declares ought to be forwarded. An attacker may not be both passive and active.
- *traffic-aware:* The attacker is aware of all the traffic on the graph, not just the traffic of the nodes it contains. "Aware" here means that the attacker knows the information sent along each edge.

Attackers are assumed to have the ability to compute polynomial time operations (in the number of nodes).

The primary point of interest for this system is defense against passive, traffic-aware attackers. These may posses a small number of nodes (a large number of nodes is extremely hard to hide from, as they can develop a near-complete picture of the traffic on the graph, and how it's routed), and as passive attackers, can at most observe and eavesdrop. This is a similar situation to that explored by Beimel and Dolev's "Busses" [3].

IV. Methods

The basic idea behind this protocol is to send onion-routed messages through the graph, with probabilistic message branching at each node, so that very quickly, a message's recipient set can grow very large (exponentially, even). This is accomplished with headers written such that, if a message received is random (which is the same as if it were encoded for a different node than the one that's just received it), then that message will be forwarded at random with desirable probabilistic branching. In this way, it is possible for cover traffic to be produced without nodes even knowing they're producing or forwarding it.

It is important to note that due to semantic encryption, nodes receiving a random message cannot distinguish it from a message that may contain encrypted or onion routed content. They will then follow any forwarding instructions in this message, producing "cover traffic," which is to say traffic without content, with desirable probabilistic properties, without knowing that they are not distributing content.

Additionally, upon receipt of a message, and while forwarding that message to its destinations, a node can initiate a message of its own. It will do this if it has content to send, or at random. The random message initiation serves both to keep the growth rate of traffic under control, and to protect against the situation in which an attacker has sent a message, and knows how many "forwardings" it has, and is watching for message initiation.

Message sending, encryption and decryption are assumed to all occur all within a time step. As such, this protocol takes place in rounds, each of which is composed of nodes receiving messages, decrypting them, reading the headers and deciding what to do with them, and then sending along any forwarded messages, as well as any new messages of their own.

It is important to note that nodes only send messages in rounds when they've received messages. To an attacker, this makes it impossible to distinguish if a node is a sender or simply a forwarder.

Communication begins when one node sends out a message, which either because of planning or because of probabilistic forwarding, is forwarded on to many other nodes, providing the opportunity for messages to be sent.

In full, upon receipt of a message m, a node x executes the following procedure:

1) The node applies its private decryption key to the message.
2) The forwarding addresses of the message are determined. A forwarding address, here, is simply an identifier of a node that this message should be forwarded to. Let the probability distribution of the number of forwarding addresses per message, if the messages are purely random bits, be called b. b is simply the distribution of the number of other nodes a received message is forwarded to. It is independent of the path length of messages. Ideally, b should be some distribution that's fairly low, so the number of messages doesn't grow out of control ($|b| < 1$), but it should not make any number of messages suspiciously unlikely, so that it is never the case that a node adding its own message to a set of forwarded messages produces an unlikely number. An exponential decay should do.
   There are a number of ways to set b. For example, b could be some function known to all nodes that takes as input some set number of bits, and provided those are random returns answers with the desired distribution. The first set number of bits of the decrypted message would then serve as inputs to determine the number of forwarding addresses, a. The addresses themselves could simply be the first a sets of bits long enough to code for an address, after the bits used for b. Addresses should be uniformly distributed over all nodes with which this node can communicate, so the process of turning these sets of bits into addresses necessitates an even distribution function, again some standard conversion known to all nodes.
3) The message body is forwarded. The node sends the body of the message (not including bits used to calculate the forwarding addresses) on to all the forwarding

addresses. It is assumed that forwarding addresses are small, and random padding can be used to maintain uniform message size across several such forwardings.

4) If the node has generated content since last it received a message, it *onion* wraps this content. That is to say that it affixes to the message a header featuring a random number of forwarding addresses, including the destination address, and encrypts it with the public key of a node that can communicate with the destination. This node is the destination of the encrypted message, which itself must have a header affixed, and so on, for some number of layers that define the route. Though route lengths can be random, they should have some average length ℓ, used to determine statistics about the system. The outermost layer should be encrypted to some node with which this node can communicate, and sent to that node.

   This process creates a content-containing route which is prescribed, and at each node along this route, the message body is forwarded to at least one other node, the node one ahead on the route, as well as zero or more other nodes, which will receive effectively random messages, since the body was not encrypted with those nodes' public keys.

   The number of forwarding addresses in the header of a message that is part of a content-containing route must be at least one, and as such that distribution cannot be b. Let the distribution of the number of forwarding addresses in the header of a message containing content be b_f.

5) Furthermore, a node that has received 1 or more messages this round must choose a number from the probability distribution p to be the number of additional "cover messages" to initiate. These are messages with random content sent to random nodes with which this node can communicate. Obviously, nodes would prefer not to have to send excess traffic, so p should be kept low, but it is necessary to regulate the amount of traffic in the system, as well as provide plausible deniability in the case that all messages received by the node this round were sent by attacker nodes, and they're counting the number of forwaridngs they asked for and the number of messages sent out. As nodes can control the headers of "cover messages," let the statistical distribution of number of forwarding addresses in the header of a cover message be b_p.

A. Optimization

The challenge in constructing a randomized branching messaging system is in preserving short latency and high anonymity while minimizing the number of messages nodes must forward each round. Furthermore, since it is an exponentially branching system, care must be taken to see that the number of messages does not grow out of control, nor diminish entirely. If nodes should altogether stop sending messages, none will be able to start, since nodes only send if they've received. (For practical applications, there might be some contingency, or small probability of spontaneous message sending.)

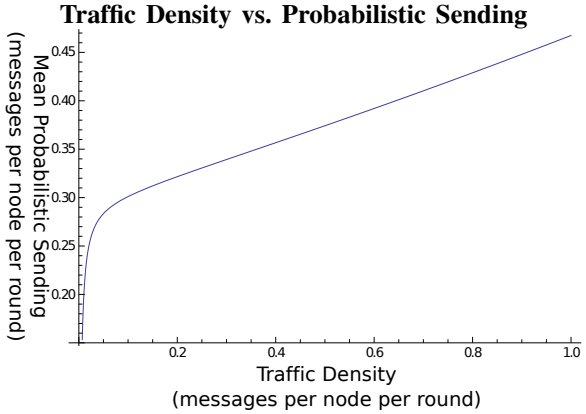

Fig. 1. With $f = 0.001$ and $b = 0.703506$ (see (2))

The parameters are:

n	:	number of nodes on the graph
f	:	frequency (content per node per time step)
ℓ	:	length of an average onion route
p	:	cover messages per node that's received this round
b	:	forwarding addresses per random message
b_f	:	forwarding addresses per content message
b_p	:	forwarding addresses per cover message

Define traffic density d to be the mean number of messages per node in a given round.

To begin with, b has to be less than 1, and the number of messages shouldn't be set to grow out of control if $p \leq 0$.

Given that there exists traffic density d, and message destinations are random and evenly distributed, each target (node) has a poisson distribution of messages sent to it, so each target's probability of receiving no messages in a given round is e^{-d}.

Therefore, each node's probability of having the opportunity to send messages in a given round is $1 - e^{-d}$, and so each node will want to send an average of $\frac{f}{1-e^{-d}}$ content-containing messages when it has the opportunity.

Furthermore, a node that has received a message receives an average of $\frac{d}{1-e^{-d}}$ messages, and so will be obligated to forward $\frac{bd}{1-e^{-d}}$ messages.

In addition, a node that has received a message will want to send out probabilistic traffic to maintain some desired traffic density, so it sends p more messages.

The number of messages sent by a node that has received messages should equal, on average, the number of messages a node that has received at least one message has received. That is to say, the system is in equilibrium when:

$$\frac{d}{1-e^{-d}} = p + \frac{db+f}{1-e^{-d}} \tag{1}$$

$$p = \frac{(1-b)\,d - f}{1-e^{-d}} \tag{2}$$

Note that p is increasing as a function of d for reasonable values of f and b. That's good, because it means that nodes can keep a constant p to maintain a level of traffic.

Nodes onion route a message some distance ℓ, and that distance is determined by the number of nodes the traffic reaches at that distance. With each successive round, a message is forwarded by another node. When this happens, the node also may forward several other messages, which, to an observer, are all equally likely to be the continuation of the message received. Therefore the number of possible recipients after one round is expressed in (1), and the number of possible recipients after two is that number squared, and so on. The number of possible recipients after n rounds is the expression from (1) to the nth power, and so to get a recipient set of order size n, the path length should be:

$$\ell = \frac{\ln(n)}{\ln\left(\frac{d}{1-e^{-d}}\right)} \tag{3}$$

1) The Talkative and the Paranoid Should be Indistinguishable: In order to set these parameters, however, the nodes must choose some optimal level of traffic density. Consider that nodes creating messages with content (all messages they put in the onion route, not just the first one) don't produce messages with a distribution of forwarding instructions b. That is to say, all messages in the onion route will say "forward me at least once," but random messages forward a number of times taken from the distribution b, which must be less than 1. Content-full messages instead have a number of forwarding instructions with statistical distribution b_f.

By analyzing the network traffic, and any messages sent to its nodes, an attacker could determine the distribution of number of forwarding addresses per message, which will vary with the portion of the messages that are part of an onion route. In order to cancel this out, and make the statistical distribution of the number of forwarding addresses per message equal to that of a set of random messages, b_p must be less than b, since b_f is greater than b. Specifically, the mean of the two distributions b_p and b_f, each weighted by the average number of such messages generated by a node that's active this round, should be equal to b.

$$b = \frac{pb_p + \frac{\ell f b_f}{1-e^{-d}}}{p + \frac{\ell f}{1-e^{-d}}} \tag{4}$$

$$b = \frac{\left(\frac{(1-b)d-f}{1-e^{-d}}\right) b_p + \frac{\ell f b_f}{1-e^{-d}}}{\left(\frac{(1-b)d-f}{1-e^{-d}}\right) + \frac{\ell f}{1-e^{-d}}} \tag{5}$$

$$b = \frac{((b-1)d+f)\, b_p \ln\left(\left(1+\frac{1}{e^d-1}\right)d\right) - b_f f \ln(n)}{((b-1)d+f)\ln\left(\left(1+\frac{1}{e^d-1}\right)d\right) - f\ln(n)} \tag{6}$$

Solving (6) for b, it is possible to numerically solve for traffic density d. Thus, taking b_f, and b_p to be defining parameters of the system, p can be varied to achieve minimum d. For an example, see figure 2. Unfortunately, d cannot be solved for analytically.

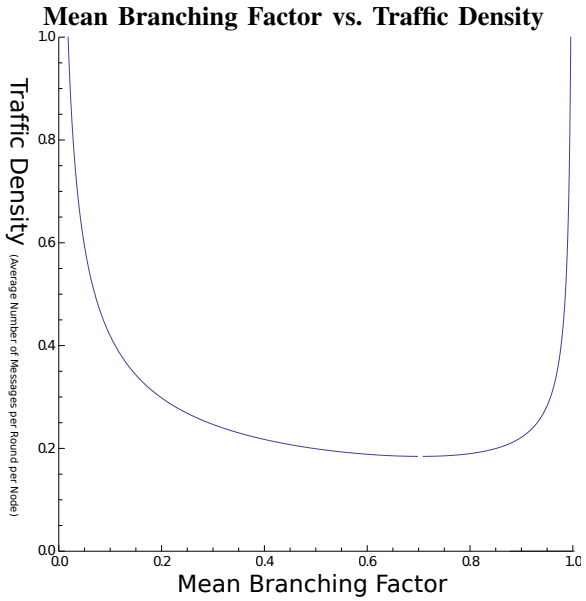

Fig. 2. Maintaining a constant receiver anonymity set size of 1000 with varying $b, b_f = 1.2, f = 0.001, b_p = 0$ (cover traffic goes one round and does not forward). p is set by (2), and ℓ by (3). Note that here we can numerically optimize b.

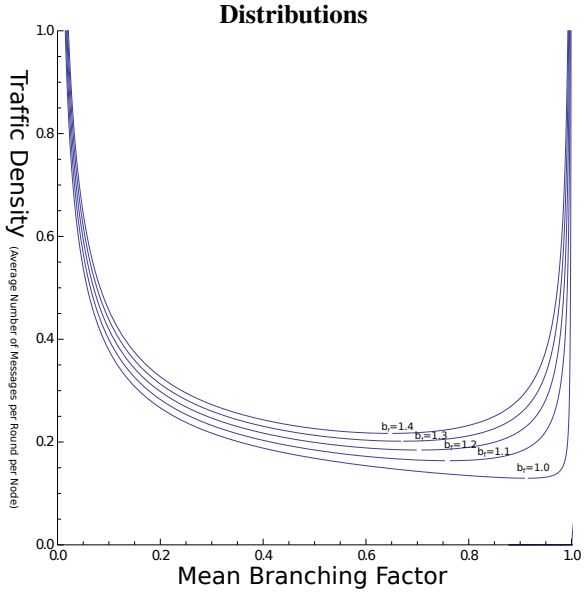

Fig. 3. Maintaining a constant receiver anonymity set size of 1000 with varying $b, f = 0.001, b_p = 0$ (cover traffic goes one round and does not forward), and $bf = 1, 1.1, 1.2, 1.3, 1.4$. p is set by (2), and ℓ by (3). Note that here we can numerically optimize b, but optimizing for b_f will lead you to $b_f = 1$, so your choice of b_f must take other factors into account.

2) Choosing f, n, b_f, and b_p: The defining parameters of the system are thus, in effect, the frequency at which nodes generate content f, the size of the recipient anonymity set of a message n (this is generally assumed to be the number

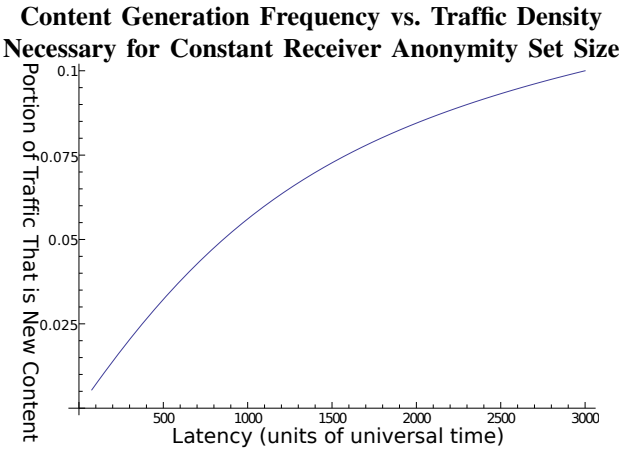

Fig. 4. Maintaining a constant receiver anonymity set size of 1000 with varying $c, f = 0.001, b_f = 1.2, b_p = 0, b$ selected according to the above optimization conditions (and boundary constraints, so there is only one possible optimum b). This demonstrates the tradeoff using message transmission delaying between latency and $\frac{fc}{d}$, the portion of traffic that is new content each round.

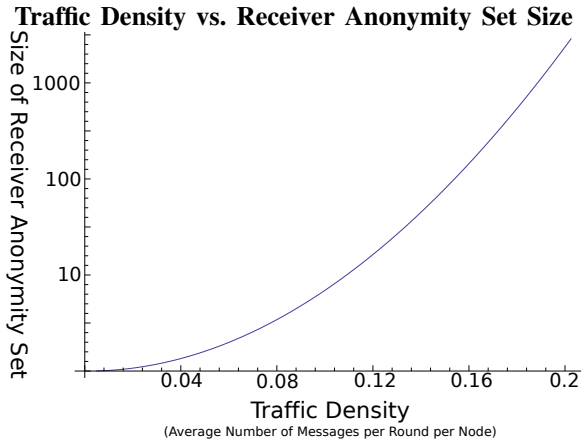

Fig. 5. Varying ℓ, the length of onion routes for content, with $f = 0.001, b_f = 1.2, b_p = 0$. In linear cases (such as an "n-seat bus," or random equalized in-out forwarding) traffic density is constant (usually 1). Note that traffic density of 1 here is a receiver anonymity set size of 2.6×10^{83} nodes.

of nodes), the number of forwarding addresses in a content-filled message b_f, and the number in a cover message b_p. Note that as b_f becomes as small as possible (approaches 1), messages become essentially onion routed only. The only anonymity comes from long routes and traditional mixing (cases when routes cross at the same node and the same time). This is higher latency, but lower traffic density. (See figure 3) Note also that b_p contributes to the recipient set of a message directly, so while it may seem like useless cover traffic, boosting b_p can also be helpful.

Consider that if b is extremely low, branching will come primarily from probabilistic traffic. This is ok, and can even lead to lower traffic density if the attacker possesses very few nodes. However, lower b means a larger portion of the tree is consciously constructed by nodes, and so the nodes constructing it know what is and is not cover traffic. Attackers with more nodes can be better thwarted with higher b.

3) Time Distortion: Inserting Delay to Artificially Boost Frequency: When choosing the size of a time step, nodes can send messages at some slower speed relative to the frequency at which they generate content to in effect produce content at some higher rate. This slower speed, of course, has a detrimental effect upon the message latency, which is proportionally slowed, as well as the traffic density, which increases relative to the slower speed of messaging, but decreases relative to the rate of content generation. This changed density decreases ℓ, and so latency does not increase linearly. The net result is that by slowing transmission speed linearly by some factor c, content is in effect generated at a rate of fc relative to the new transmission speed, and d must be calculated accordingly. Latency becomes ℓc, and the useful parameter to compare it to is $\frac{fc}{d}$, or the portion of the messages a node sends that are new content. See figure 4.

B. Some Numeric Examples

To help get an idea of what this amounts to, consider some numeric examples of communication schemes. Again, b_p, b_f, f, and n are all parameters of the system. ℓ, b, and d are all calculated.

b_p	b_f	f	n	ℓ	b	d
0	1.2	0.001	1000	76.0718	0.70351	0.18445
0.5	1.2	0.001	1000	45.9454	0.69936	0.30863
0	2	0.001	1000	49.0758	0.70001	0.28844
0	1.2	0.1	1000	7.74501	0.65630	2.1578
0	1.2	0.001	100000	59.1834	0.70168	0.2382

V. Conclusions

Branching messaging illustrates many of the trade-offs in systems of anonymous communication. Increases in latency can buy lighter use of resources, and increases in resource use can buy larger anonymity sets, and thus larger networks. Increasing content generation demands increased traffic density, but in many ways these trade-offs compare favorably with existing systems.

Most traditional anonymity solutions, such as Rackoff and Simon's Rapid Mixing [2], Beimel and Dolev's "Busses" [3], and Kannan, Ray, and Iyengar's Randomized Message Forwarding [4] rely on relatively high traffic density to maintain anonymity. Mixing solutions can require as many as all nodes each round to send messages, in order to preserve anonymity for any number of messages. Busses necessitates relatively few transmissions of data, but each transmission carries $\mathcal{O}(n) - \mathcal{O}(n^2)$ data, amounting to an effectively linear (or higher) amount of cover traffic in the number of nodes, the same as Randomized Message Forwarding.

In contrast, the potential size of the receiver anonymity set (which need be no larger than the number of nodes communicating) in a branching system is at least exponential in the traffic density (number of messages sent per node per unit time). For relatively small levels of content-generation,

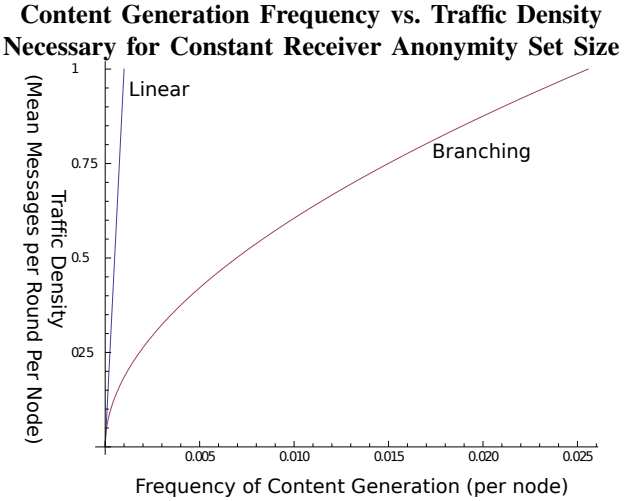

Fig. 6. Maintaining a constant receiver anonymity set size of 1000 with varying $f, b_f = 1.2, b_p = 0, b$ selected according to the above optimization conditions (and boundary constraints, so there is only one possible optimum b). Linear results represent the traffic density of linear cases (such as an "n-seat bus," or random equalized in-out forwarding).

the number of nodes that would necessitate the kind of "every node sends a message each round" communication in some other systems is astronomical. For example, with $f = 0.001, b_f = 1.2, b_p = 0$, $d = 1$ would imply that there are 2.6×10^{83} nodes. See figure 5.

Furthermore, unlike protocols such as busses, in which proportionally more rounds of busses are required to deliver content at higher frequencies, the traffic density in a branching system necessary to maintain higher levels of content creation grow substantially sub-linearly. See figure 6.

Branching messaging exists between the extremes of non-statistical hard anonymity of busses and randomized forwarding and the highly statistical high-traffic systems like rapid mixing. It is tunably to adapt to a variety of desirable latencies, sizes, and resource levels, and it is this very flexibility that makes it worthy of study.

ACKNOWLEDGMENT

Professor Ho has been remarkably helpful in guiding me through the research process, from project ideas to publication. I cannot thank her enough.

REFERENCES

[1] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, pp. 84–90, February 1981. [Online]. Available: http://doi.acm.org/10.1145/358549.358563

[2] C. Rackoff and D. R. Simon, "Cryptographic defense against traffic analysis," in *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, ser. STOC '93. New York, NY, USA: ACM, 1993, pp. 672–681. [Online]. Available: http://doi.acm.org/10.1145/167088.167260

[3] Beimel and Dolev, "Buses for anonymous message delivery," *Journal of Cryptology*, vol. 16, pp. 25–39, 2008, 10.1007/s00145-002-0128-6. [Online]. Available: http://dx.doi.org/10.1007/s00145-002-0128-6

[4] R. Kannan, L. Ray, and S. Iyengar, "Randomized message forwarding with equalized incoming/outgoing traffic rate: A mechanism for ensuring anonymous communication," in *Intelligent Sensing and Information Processing, 2005. ICISIP 2005. Third International Conference on*, dec. 2005, pp. 183 – 188.

[5] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, pp. 65–75, 1988, 10.1007/BF00206326. [Online]. Available: http://dx.doi.org/10.1007/BF00206326

[6] A. Pfitzmann, T. Dresden, and M. Hansen, "Anonymity, unlinkability, unobservability, pseudonymity, and identity management a consolidated proposal for terminology," Tech. Rep., 2005.