Performance Modeling of Anonymity Protocols

Niklas Carlsson[a,*], Carey Williamson[b], Andreas Hirt[b], Michael Jacobson, Jr.[b]

[a] *Linköping University, Linköping, Sweden*
[b] *University of Calgary, Calgary, AB, Canada*

Abstract

Anonymous network communication protocols provide privacy for Internet-based communication. In this paper, we focus on the performance and scalability of anonymity protocols. In particular, we develop performance models for two anonymity protocols from the prior literature (Buses and Taxis), as well as our own newly proposed protocol (Motorcycles). Using a combination of experimental implementation, simulation, and analysis, we show that: (1) the message latency of the Buses protocol is $O(N^2)$, scaling quadratically with the number of participants; (2) the message latency of the Taxis protocol is $O(N)$, scaling linearly with the number of participants; and (3) the message latency of the Motorcycles protocol is $O(\log^2 N)$, scaling logarithmically with the number of participants. Motorcycles can provide scalable anonymous network communication, without compromising the strength of anonymity provided by Buses or Taxis.

Key words: Anonymous network communication, Protocol performance

[0]NOTICE: this is the authors' version of a work that was accepted for publication in *Performance Evaluation*. The final version will appear in the Performance Evaluation journal (Elsevier). Changes resulting from the publishing process, such as editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication.

[*]Corresponding author

Email addresses: `niklas.carlsson@liu.se` (Niklas Carlsson), `carey@cpsc.ucalgary.ca` (Carey Williamson), `hirt@cpsc.ucalgary.ca` (Andreas Hirt), `jacobs@ucalgary.ca` (Michael Jacobson, Jr.)

1. Introduction

Certain Internet applications require anonymity, wherein the identities of communicating participants are concealed. These applications can be used both for benevolent social purposes (e.g., crime stoppers, freedom of speech, on-line counseling for victims of abuse, protecting human rights, whistle blowing), as well as for nefarious purposes[1] (e.g., criminal activities, illegal file-sharing, malware distribution, terrorism). In general, anonymous communication provides privacy, and eliminates the risks associated with compromised identities.

Anonymous communication on the Internet is supported using *anonymity protocols* [1]. These protocols protect the identities of communicating parties from others, thus ensuring privacy for network-based communication. Such protocols can provide both *data anonymity* and *connection anonymity.* Data anonymity [9] removes identifying data in messages, such as the sender address in an e-mail. Connection anonymity [9] obscures the traffic communication patterns, preventing traffic analysis that traces a message through the network from the initiator (the original sender of a message) to the responder (the final receiver of a message). Connection anonymity can be further subdivided into *sender anonymity*, *receiver anonymity*, *mutual anonymity*, and *unlinkability* [27]. The type of connection anonymity required is application-dependent.

Two important characteristics of anonymity protocols are the *strength* of anonymity that they provide, and the *scalability* of the protocol itself. The strength of an anonymity protocol is usually measured in an information-theoretic sense, by determining the probability that an attacker or observer can identify communicating parties. These issues are well addressed in the security and privacy literature. Scalability refers to the communication overhead associated with a protocol (e.g., end-to-end latency), and how this grows with the size of the network. These issues are less well-studied in the literature, particularly since few anonymity protocols are implemented and used in practice. For example, Boukerche *et al.* [6] study an anonymous routing protocol for wireless ad hoc networks, but only using simulation.

There is typically a tradeoff between strength of anonymity and the scalability of an anonymity protocol. For example, strong anonymity can be provided by aggregating many messages into batches (mixes) before forwarding

[1]As with encryption, legislation may be required to curtail malicious wrongdoings.

them, but the batching process itself increases the end-to-end message delay. Similarly, anonymity can be fortified with dummy cover traffic (i.e., fake messages) in the network, but this increases the bandwidth consumption on the network as well as the processing overhead for participating nodes.

In this paper, we focus on the performance and scalability of three anonymous network communication protocols: Buses, Taxis, and Motorcycles. We focus on the end-to-end message latency in these protocols, and how this latency scales with the network size (i.e., number of participants). To the best of our knowledge, our paper is the first to provide a detailed performance analysis of multiple anonymity protocols to assess their practical scalability.

The primary contributions in this paper are the following:

- We use analysis, experiments, and simulation to show that the end-to-end message latency of the Buses protocol is $O(N^2)$, scaling quadratically with the size of the network.
- We use analysis, experiments, and simulation to show that the message latency of the Taxis protocol is $O(N)$, scaling linearly with the network size.
- We propose a new anonymity protocol, Motorcycles, that improves upon Buses and Taxis.
- We use analysis, experiments, and simulation to show that the message latency of the Motorcycles protocol is $O(\log^2 N)$, scaling logarithmically with the network size.

The rest of this paper is organized as follows. Section 2 provides background information on anonymous communication. Section 3 describes the three anonymity protocols analyzed in our paper. Section 4 and Section 5 present our analysis of these protocols. Section 6 presents simulation and experimental results to validate our analytical models. Finally, Section 7 concludes the paper.

2. Background

2.1. Anonymity Techniques

Anonymous communication is a vibrant research area with many anonymity schemes proposed [1, 15, 26]. Delay-tolerant applications, such as e-mail, can use strong anonymous communication schemes [10]. However, interactive applications, such as Web browsing and SSH, require lower overhead to minimize the end-to-end message latency.

Designing a strong anonymous communication scheme with low latency is a challenge [11]. As an interim solution, anonymous communication schemes such as Crowds [29] and Tor [11] have been used for interactive applications, despite their known vulnerabilities [11].

The three basic techniques underlying most anonymous communication protocols are *broadcasting*, *mixes*, and *buses*. The strength and scalability of an anonymity protocol are inherited from these basic techniques.

The broadcasting technique requires each participant to conduct secret coin flips, and then broadcast the resulting parity to all the other participants [8]. Since this traffic overhead is $O(N^2)$, the broadcasting approach is no longer mainstream in the literature [1], and is not discussed further here.

The mixes and buses techniques can be measured by their performance and the strength of anonymity provided. The performance metric considered in this paper is the average message latency, and how it scales with the number of participants. The measure of anonymity is how well an initiator or responder is hidden among the participants.

The following subsections introduce mixes and buses, while Section 3 describes the three specific anonymity protocols considered in our work.

2.2. Mixes

Mixes provide very strong anonymity [7]. The basic idea is to re-route a message through a sequence of mixes, each of which collects a batch of inputs and randomly re-orders the outputs. In addition, the message is encrypted more than once at the source, with one layer of encryption removed by each mix along the routing path.

The goal of mixes is to prevent an attacker from correlating an input message with its corresponding output message. However, the only proven secure solution requires each initiator to send at least one message to each responder in every batch. This cover traffic scales quadratically with the number of participants.

Debate arises as to how much anonymity is sacrificed when cover traffic is reduced or eliminated [11]. To reduce the overhead of input-output mixing, anonymous communication schemes such as Tor [11] and Crowds [29] use no cover traffic at all. The authors acknowledge the tradeoffs between overhead and vulnerabilities to known attacks [11, 29]. Other schemes such as

JAP [5] and MorphMix [30, 35] use partial cover traffic, while Mixmaster [25], Mixminion [10], and Tarzan [14] use full cover traffic.

2.3. Classic Buses

The buses anonymity technique was proposed by Beimel and Dolev [3]. We refer to their work as Classic Buses.

The Classic Buses protocol is based on the metaphor of a city bus, which traverses a circular path among all of the participants in the anonymity network (see Figure 1). Messages travel in the seats of a bus, appropriately obscured by layers of encryption. The bus traverses the network, passing from one node to the next. When in possession of the bus, a node replaces the bus with its decryption and retrieves messages intended for itself, replacing the message's seat with random data. In addition, the node places layered encrypted messages on the bus to be delivered to other nodes. By encrypting the message, only the intended recipient can recover its messages. Since each node alters the bus, an adversary cannot tell if a node inserts a new message or replaces an existing seat with its decryption.

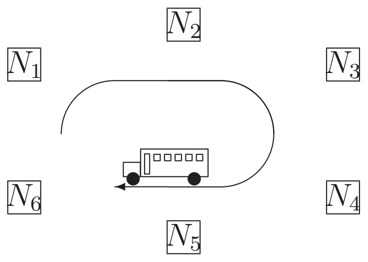


Figure 1: Example of Routing in Buses

The operation of Classic Buses is best illustrated with an example. Consider the bus network in Figure 1, where nodes N_1 through N_6 are part of the network, and the bus tours the network in a clockwise round-robin fashion. If N_1 wants to send an anonymous message M to N_4, then the message routing path will be N_1 to N_2 to N_3 to N_4. To send the message, N_1 encrypts it as $E_{K_{N_2}}(E_{K_{N_3}}(E_{K_{N_4}}(M)))$, where $E_{K_{N_i}}(x)$ denotes the encryption of x with N_i's public key. Node N_1 then places the message on a *randomly* chosen seat on the bus. When N_2 receives the bus, it decrypts all messages on the bus, peeling away a layer of encryption, and forwards the resulting contents (including the still-obscured valid message) on the bus to N_3, which in turn decrypts and forwards to N_4. When N_4 receives the bus, it decrypts all seats, revealing the message M destined for itself. The message is removed and replaced with random bits (or a new outgoing message) before the bus is forwarded.

2.4. Qualitative Assessment

Table 1 provides our qualitative assessment of Classic Buses compared to several other anonymity protocols in the literature [17]. The theoretical design of Classic Buses [3] provides strong anonymity guarantees. The primary weaknesses are with respect to active adversaries (e.g., replay attacks); however, these problems are fixable. While the original authors never implemented or deployed the Classic Buses protocol, we use their underlying design as the starting point for our own work.

Table 1: Vulnerabilities of Anonymity Protocols

Anonymity Protocol	Vulnerable to Global Eavesdropper?	Vulnerable to Passive Adversary?	Vulnerable to Active Adversary?
Anonymizer [2]	Yes	Yes	Yes
Babel [16]	Yes	Yes	No
Chaum's Mixes [7]	No	No	Yes
Classic Buses [3]	No	No	Yes
Crowds [29]	Yes	Yes	Yes
Hordes [32]	Yes	Yes	Yes
LPWA [24]	Yes	Yes	Yes
Onion Routing [28]	Yes	Yes	Yes
P5 [31]	Yes	No	Yes
Tor [11]	Yes	Yes	Yes

3. Anonymity Protocols

3.1. Buses

There are several versions of the buses protocol described in [3]; we now focus on a specific version that we have implemented and demonstrated experimentally [17].

There are three key innovations in our version of the buses protocol. First, we use *owned seats*. This feature restricts participating nodes to only insert messages into specific (owned) seats on the bus. This feature drastically reduces the number of bus seats required, while avoiding the random seat collisions that can occur in the Classic Buses protocol, especially under high load. Note that ownership of a seat does not compromise anonymity, since an observer cannot determine whether a seat contains a message or not (i.e., a semantically secure cryptosystem is used). Second, our protocol provides *reliable message delivery*, using anonymous acknowledgements, timeouts, and retransmissions. This mechanism also provides protection against replay attacks. Third, we use *nested encryption with indirection.* This feature decouples the logical re-routing path of messages from the physical routing of messages around the network. This change reduces the dependence on a known network topology (i.e., static route), making the Buses protocol usable as a peer-to-peer (P2P) overlay network on the Internet.

A new nested encrypted message is created by encrypting a message with the responder's key, selecting a random set of up to L indirection participants, and then recursively encrypting the message with each indirection participant's corresponding key. In practice, a relatively short indirection path length of 1 or 2 nodes is sufficient to provide strong anonymity [19]. For example, consider the bus network in Figure 1. If N_1 wants to send an anonymous message M to N_4, then N_1 randomly chooses intermediate nodes for the re-routing path. Suppose that it chooses two indirection nodes, N_2 and N_5, in that order. The message routing path will be N_1 to N_2 to N_5 to N_4, requiring *more than one cycle* of the bus to achieve delivery. That is, the bus circulates the network clockwise, visiting every node in turn, but only the nodes involved in the message re-routing path alter the contents (seats) of the bus. The message is encrypted multiple times as $E_{K_{N_2}}(E_{K_{N_5}}(E_{K_{N_4}}(M)))$ and placed on a seat owned by N_1. N_2 receives and decrypts a copy of the message, peeling away a layer of encryption, and forwards the message using its own seat to N_5, which in turn forwards to N_4 following decryption. When N_4 receives the bus, it decrypts the seat from N_5 to reveal the message M from N_1.

The basic operation of the bus otherwise remains the same. When a participant receives the bus, it copies the bus, modifies the bus, and forwards the bus. Unlike Classical Buses, the bus is copied for *off-line* processing, which can take place after the incoming bus has been forwarded. The main component of the off-line processing is to decrypt the bus, removing a layer of encryption from any nested messages. A participant modifies the bus by inserting any messages to forward or send in its owned seats, and replacing any unused owned seats with random data[2] before forwarding the bus to the next participant on the route.

While the buses protocol is conceptually elegant, and requires no additional cover traffic, it nonetheless has an end-to-end message latency that scales quadratically with the number of participants. Simply stated, the cycle time of the bus for N nodes is $O(N)$, and the processing and network delays at each node are also $O(N)$. The resulting latency is $O(N^2)$, which limits the practical scalability of the protocol. See Section 4 for details of this analysis.

3.2. Taxis

The Taxis protocol [18] improves upon the buses protocol significantly. Rather than having one large bus with $O(N)$ seats circulating the network, the Taxis protocol has N taxis each with $O(1)$ seats circulating the network, concurrently and independently.

This basic idea reduces both the processing and network delay per tour. Instead of a bus with rows of owned seats, each participant owns a single taxi that contains only its owned seats. This allows a participant to separate the taxis into owned and unowned taxis. Unowned taxis require minimal processing, and can simply be forwarded, resulting in a metaphorical 'fast lane'. A taxi is only delayed once per tour for full seat processing at the participant that owns the taxi, resulting in a message delay that is $O(N)$. See Section 4 for details of this analysis. Similar to the Buses protocol, indirection and nested encryption are used.

3.3. Motorcycles

Our new proposed anonymity protocol combines the idea of Taxis with the idea of Chord [34], an efficient Distributed Hash Table (DHT) mechanism for content-based routing and lookup on a peer-to-peer (P2P) overlay network. In Chord, the identifier space is organized into a logical ring, and information lookup can be done in logarithmic time, using a *finger table* that leads directly to a (logarithmic) subset of nodes that are strategically placed around the ring.

[2]Note that even the *dummy traffic* seats containing random data must be encrypted before transmission, so that the processing delays are consistent at each node, and not vulnerable to a *timing attack* by an adversary.

To continue the transportation analogy, we refer to our Chord-based Taxis protocol as Motorcycles. While motorcycles have limited seating, they can easily take off-road shortcuts to minimize travel time to a destination.

In the Motorcycles protocol, each node has a finite set of motorcycles, each with one seat, that are sent to nodes in its finger table. A node with a message to send or forward inserts a message into one of its owned motorcycles. The message travels a Chord-like (logarithmic) routing path through the network to its indirection nodes, transferring to other motorcycles as necessary for forwarding, until it eventually reaches its target destination. While this protocol requires messages to be transferred between motorcycles at each node (not only at indirection nodes), we note that the Chord-like routing significantly reduces the path length from source to destination.

Technically, an external observer might attempt to discern the sender-receiver pair for a message, based on the routing activity of motorcycles. To prevent this traffic analysis, an outgoing motorcycle is *multicast* simultaneously[3] to all entries in the outgoing finger table. Most nodes silently discard the dummy traffic, while the intended forwarder sends the message onward in the same fashion, using its own motorcycles.

Figure 2 illustrates how Motorcycles work, when routing from node 0 to node 7 using indirection nodes 14 and 4. In this example, we assume that there are exactly 16 nodes participating in the network. Figure 2(a) shows the finger table of node 0. Figure 2(b) shows the total message traffic needed to route from node 0 to indirection node 14. Here, the message path is illustrated using solid lines and the dummy traffic using dotted lines. Finally, Figure 2(c) shows the message path. The source, destination and indirection nodes are marked in red. The fowarding nodes are shown in green. Note that nested decryption is only done at the indirection and destination nodes.

3.4. Security

The security of Buses is discussed in [17], and that of Taxis in [18]; neither protocol appears susceptible to any known active or passive attacks. Motorcycles is built upon the security provided by Buses and Taxis. Furthermore,

[3]Alternatively, (encrypted) messages can be sent in round-robin fashion to each node in the node's finger table. In either case, the traffic pattern observed by an external observer would be independent of the message load and would not provide insights with regards to the message patterns.

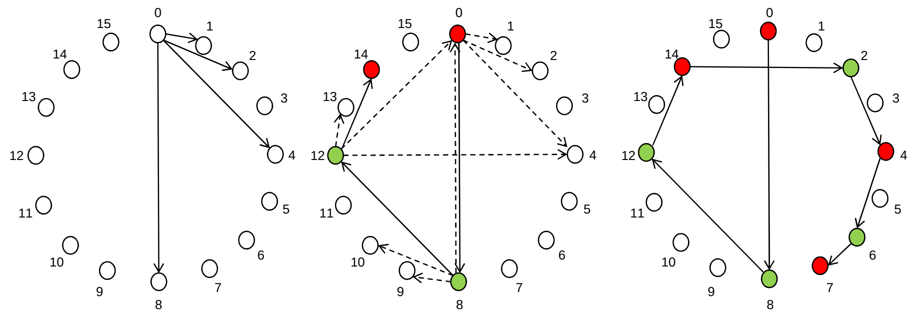


Figure 2: Motorcycles protocol. (a) Finger table of node 0. (b) Message traffic associated with routing from node 0 to node 14. (c) Message path from node 0 to node 7, using indirection nodes 14 and 4.

careful design and analysis were conducted to ensure that the size and probability distribution over the anonymity sets yield maximum entropies. A detailed security analysis appears in [19]. This paper focuses solely on the performance of the protocols.

4. High-Level Performance Model

In this section, we first develop a high-level mathematical model to characterize end-to-end message delay in our anonymity protocols. This unified model applies for all three protocols of interest. We then refine the analysis to produce closed-form solutions for each protocol.

4.1. Notation and Assumptions

Table 2 summarizes the notation used in our model. From top to bottom, the table is organized into four logical sections: anonymity protocol configuration parameters, message traffic workload parameters, network model parameters, and the variables used to represent different aspects of the end-to-end delay, as described in Section 4.2.

We make the following assumptions in our model:

- New messages are generated according to a Poisson process with an aggregate arrival rate λ. Messages are variable-sized, but each message

Table 2: Notation for Anonymity Protocol Analysis

Symbol	**Definition**
N	Number of nodes in the anonymity network
K	Number of seats per node on bus, taxi, or motorcycle
L	Number of indirection layers used
s	Seat size in bits
λ	Aggregate generation rate for new messages (per sec)
λ_i	Message generation rate for node i (per second)
r	Network bandwidth in bits per second (bps)
p	Per-hop network propagation delay in seconds
D_{SR}	One-way message delay from S to R
D_S	Time for source to encode a nested message
W_S	Waiting time at the originating node S
H_{SR}	Average number of hops traversed from S to R
D_{seat}	Processing time per seat on bus, taxi, or motorcycle
D_{proc}	Per-hop nodal processing delay
D_{net}	Per-hop transmission and propagation delay
H_T	Average number of transfer nodes from S to R
W_T	Message waiting time at a transfer node
D_R	Time for receiver to decode a message
T_C	Cycle time for a bus, taxi, or motorcycle
Q_C	Number of cycles incurred

fits[4] within a fixed-size seat (e.g., $s = 3$ KB).

- Message traffic is homogeneous and uniformly distributed across all N participating nodes in the network. That is, $\lambda_i = \frac{\lambda}{N}$. The receiver of a message is always distinct from the original sender, and the receiver is selected uniformly at random from the *other* nodes. Indirection nodes are uniformly selected from *all* the nodes (including the original sender and the receiver).

- The nodal processing time is short relative to the cycle time of a bus, taxi, or motorcycle. In particular, we assume that a message to be

[4]Larger messages can be split into smaller pieces, and modeled with a *batch* message generation process.

transferred (or acknowledged) by a node is fully processed (decoded) by the time the next eligible bus, taxi, or motorcycle arrives.

- The network nodes are homogeneous. The processing time (D_{proc}) is the same for all nodes, and is independent of the load. Hence, the cycle time T_C depends only on the number of hops H_{SR} on the routing path and the per-hop processing (D_{proc}) and network delays (D_{net}). In particular, $T_C = H_{SR}(D_{proc} + D_{net})$.

- The network delay (D_{net}) for transmission and propagation is the same for each node pair (hop) traversed by a message. In a wide-area network, this means that the transmission and propagation delays dominate the (network) queueing delays (which differ from the *message* queueing delays considered in our analysis of anonymity protocols).[5]

- Message queueing occurs *only* when waiting for an outgoing bus, taxi, or motorcycle. This can happen at the original sender and at each transfer node. With Buses and Taxis, transfers only take place at the indirection nodes. With the Motorcycles protocol, both indirection and forwarding nodes can act as a transfer nodes. In a scenario with acknowledgements, message queueing can occur at the receiver as well.

- Finally, our queueing analysis assumes that the transfer queue of each node is independent of the transfer queue of any other node, and also that arrival events are Poisson. While this may not be completely true in practice, we do believe that this provides a reasonable approximation.

4.2. Model Overview

This section provides an overview of the structural model for a homogeneous system with N nodes, each generating new messages at a rate λ/N. Throughout our analysis, the primary metric of interest is the expected delivery time D_{SR} for a message from the sender to the receiver. For simplicity, we initially focus on one-way message traffic (i.e., ignoring acknowledgements),

[5] In general, we do not consider background traffic or other external elements that potentially may affect the *network* queueing delays.

since this suffices for relative performance comparisons of the three protocols. We also assume that the traffic load is evenly distributed[6] across all nodes. Extensions to heterogeneous and bi-directional traffic are discussed in Section 5.

Our unified model is developed to allow us to compare and contrast the three different protocols, and does not consider second-order effects due to slight variations within each protocol. While protocol-dependent policies such as timeouts and retransmissions can be important for reliability purposes, we do not model such effects since their performance impacts are typically small.

4.2.1. Delay Components

Consider an arbitrary message in the system. The delivery time of a message has several components:

1. The originating sender S must create the message. We model this delay as the source encoding time (D_S). This delay is independent of the routing path chosen, and independent of the load along this path.
2. The sender must wait for the next outgoing bus, taxi, or motorcycle at the sending node. We denote the average such queueing delay by W_S. This delay is a *load-dependent sender-side delay*.
3. There is a *load-independent path delay* associated with forwarding the message from the sender to the receiver. If there are H_{SR} hops traversed along the delivery path from S to R, then a message incurs a processing delay (D_{proc}) and a network delay (D_{net}) at each hop. The total load-independent path delay is $H_{SR}(D_{proc} + D_{net})$.
4. A *load-dependent transfer delay* arises from the message queueing delay experienced at nodes where a message transfer occurs along the path. When a message switches from one bus, taxi, or motorcycle to another, it must wait for the next bus, taxi, or motorcycle to become available. If each message requires H_T transfers, and the average queueing delay at a transfer node is W_T, then the total expected transfer delay is $H_T W_T$.
5. The target receiver R must decrypt and receive the message. We model this load-independent delay as the receiver decoding time D_R.

[6]This case holds if destination and indirection nodes are selected uniformly at random, for instance.

Table 3: Summary of Delays and Cycle Times

Anonymity Protocol	**Load-independent**		**Load-dependent**		**Cycle Time**
	Processing D_{proc}	**Network** D_{net}	**Sender** W_S	**Transfer** W_T	T_C
Buses	KND_{seat}	$\frac{KNs}{r}+p$	$T_C(Q_C+\frac{1}{2})$	$T_C(Q_C+1)$	$N(KND_{seat}+\frac{KNs}{r}+p)$
Taxis	KD_{seat}	$\frac{Ks}{r}+p$	$T_C(Q_C+\frac{1}{2})$	$T_C(Q_C+\frac{1}{2})$	$N(KD_{seat}+\frac{Ks}{r}+p)$
Motorcycles	KD_{seat}	$\frac{Ks}{r}+p$	$T_C(Q_C+\frac{1}{2})$	$T_C(Q_C+\frac{1}{2})$	$\log N(KD_{seat}+\frac{Ks}{r}+p)$

The foregoing components are mutually-exclusive serial delays, and together constitute the entire end-to-end delivery delay. Combining the above components, the expected delivery time D_{SR} can be calculated as

$$D_{SR} = D_S + W_S + H_{SR}(D_{proc} + D_{net}) + H_T W_T + D_R. \qquad (1)$$

While the above model is universal for all three protocols, the protocols may differ significantly with regards to the individual delay components. Using the same model for all three protocols allows us to provide insights with regards to these differences. Throughout the remainder of this section, we highlight similarities and differences between the protocols.

We now turn our attention to how the individual terms in this model can be calculated for each of the protocols. Note that the nodal processing delays at the sender and receiver, D_S and D_R, are basically small constants, and can be ignored in relative performance comparisons. The other delays, in general, differ for each protocol (see Table 3). We analyze these delays next.

4.2.2. Per-Hop Delays

We first derive expressions for the processing delay (D_{proc}) and network delay (D_{net}) for each hop of the routing path.

For the Buses protocol, each node has K seats, so the bus has KN seats. If D_{seat} is the seat processing time, then $D_{proc} = KND_{seat}$. If s is the size of a message and r is the network bandwidth, then $D_{net} = \frac{KNs}{r} + p$, where p is the propagation delay to the next hop.

A similar analysis applies for Taxis and for Motorcycles, though the total number of seats per taxi or motorcycle is lower by a factor of N. Assuming small propagation delays, both these protocols therefore reduce the total per-hop delay ($D_{proc} + D_{net}$) by roughly a factor of N. The leftmost part of Table 3 summarizes these load-independent delays.

4.2.3. Number of Hops

We next compute the number of hops (H_{SR}) traversed along a delivery path from the sender to the receiver. Throughout this analysis, we assume that the sender never sends an end-to-end anonymous message to itself, that destination and indirection nodes are selected independently (uniformly at random), and that *any* node (including the sender, receiver, and the indirection node itself) can be used as the next indirection node. We further assume that there are L indirection steps.

Consider first the Buses and Taxis protocols. For these protocols, transfers (and possible message queueing) occur only at the indirection nodes; thus $H_T = L$. Intuitively, each message is forwarded $(N + 1)/2$ hops per indirection, on average. With $L > 0$ indirections, the average is $H_{SR} = (L+1)\frac{N+1}{2}$. However, this expression is not exact for the special case when $L = 0$. For this case, there are only $N - 1$ potential receivers, for an average distance of $N/2$ hops.

Consider now the Motorcycles protocol. In this protocol, messages change motorcycle at *every* node along the path; thus $H_T = H_{SR} - 1$. With Chord-based routing, there are $O(\log N)$ hops per indirection node, so both H_{SR} and H_T are *roughly* $(L + 1) \log N$. However, a more careful analysis shows that $\log N$ is actually the *worst case* upper bound for Chord-based routing; the average case is about half of this [34]. Analysis verifies that $\frac{\log N}{2}$ is a very good approximation, especially for larger N. (See appendix for details.)

Table 4 summarizes the resulting expressions for the expected number of hops (H_{SR}) traversed along a delivery path, as well as for the expected total number of transfer nodes (H_T) at which the message switches from one bus, taxi, or motorcycle to another (and queueing may occur).

Table 4: Summary of Hop Counts

Metric	**Buses/Taxis**	**Motorcycles**
H_{SR}	$\frac{N}{2}$, if $L = 0$ $(L+1)\frac{N+1}{2}$, otherwise	$\frac{L+1}{N-1}\sum_{h=1}^{\log N} h\binom{\log N}{h}$ $\approx (L+1)\ \frac{\log N}{2}$
H_T	L	$H_{SR} - 1$

We note that the Motorcycles protocol reduces the total number of hops compared to the Buses and Taxis protocols. On the other hand, the Motorcycles protocol increases the number of nodes at which message transfers occur. Future sections will discuss this tradeoff further.

4.2.4. Cycle Time

We next derive the *cycle time* for a bus, taxi, or motorcycle, which is an essential part of the (load-dependent) waiting time for messages. In particular, a message can only be launched when an eligible bus, taxi, or motorcycle is available to depart.

For the Buses protocol, the cycle time for a node is the elapsed time between visits by the (single) bus. For the taxis protocol, the cycle time is the elapsed time between visits by the node's own taxi. With Motorcycles, the cycle time is the elapsed time between visits by one of its own motorcycles.

In general, the cycle times are roughly deterministic (i.e., equal to the total processing and forwarding delay during a cycle). Assuming messages are sent as soon as possible, the cycle times can be calculated as given in the rightmost column of Table 3.

We observe that the Buses protocol has the largest cycle time among these three protocols. The cycle time for the Taxis protocol is lower by a factor of N, since each node has its own taxi. The Motorcycles protocol has the lowest cycle time, which depends on $\log N$ rather than N.

4.2.5. Waiting Times

Consider now the load-dependent delays associated with accessing a (new) bus, taxi, or motorcycle. This delay can be calculated based on the cycle time T_C and the number of cycles Q_C that the message must wait before it is among the next K messages to be served. We also need to add the expected waiting time until service for a message arriving to an empty queue.

We assume that cycle times are roughly deterministic, and messages queued at an individual node are served in FIFO order when the bus, taxi, or motorcycle is available to serve these messages. Each node (and its queue) is modeled as an independent K-limited server with deterministic vacation periods. When a bus, taxi, or motorcycle is available to the node, and there are outstanding messages in the queue, up to K messages can be served.

If there are no queued messages, then the bus, taxi, or motorcycle departs without serving any requests. Independent of the number of messages served, the next bus, taxi, or motorcycle will return in another cycle.[7] While we leave the details of our queueing analysis to Section 5, we note that with evenly distributed load, the total average (queueing) load per node is $(1 + H_T)\frac{\lambda}{N}$. Here, the $\frac{\lambda}{N}$ corresponds to new messages being generated, while $H_T\frac{\lambda}{N}$ is due to message transfers.

Now, consider the waiting time for a message that arrives to an empty queue. Clearly, this delay will depend on when in a cycle the message arrives. Assuming that cycle times are deterministic and messages are generated according to a Poisson process, the expected waiting time at the sender is half a cycle ($T_C/2$). For the case of transfer nodes, however, this is not necessarily the case. For example, with the Buses protocol, a message that has gotten off the bus for local nodal processing actually waits an *entire* cycle (T_C) before being placed back on the (same) single circulating bus. There may be some structural dependence for the other protocols as well; however, in general, we have found that $T_C/2$ provides a reasonable approximation of the average waiting time (see Section 6).

The middle columns of Table 3 summarize the load-dependent (message queueing) delays. Section 5.2 provides a more detailed analysis of the queueing delays.

4.3. Model Summary

Table 3 summarizes the component delays in our models for the three anonymity protocols. We reiterate that the cycle times for the Buses protocol are the largest, while those of the Motorcycles protocol are the smallest. The cycle-time advantage of the Motorcycles protocol is mitigated in part by its requirement for roughly $\frac{\log N}{2}$ more message transfers.

The average utilization ($\rho = \frac{\lambda}{N}(1 + H_T)T_C$) values for Buses, Taxis, and Motorcycles are $O(\frac{\lambda}{N}LN^2)$, $O(\frac{\lambda}{N}LN)$, and $O(\frac{\lambda}{N}L\ \log^2 N)$, respectively. This analysis shows that the shorter path lengths used by the Motorcycles protocol effectively compensate for the additional queueing events incurred. For a fixed load of injected message traffic, the Motorcycles protocol will have much lower node utilization. Stated another way, Motorcycles can tolerate significantly higher message generation rates than Buses or Taxis before the system saturates ($\rho \rightarrow 1$).

[7]Using terminology from the queueing literature, the node can be thought of as a server. If at least one message is served, the server endures a *service period* of duration T_C. Otherwise, it endures a *vacation period* of duration T_C.

5. Detailed Models

This section presents detailed performance models for several special cases. First, we consider the case when there are no queueing delays (i.e., $Q_C \to 0$). Here, results are obtained for the asymptotic scalability of the three protocols. Second, we use a Poisson assumption to obtain explicit expressions for the queueing delay when $K = 1$. Third, we show how the model can be extended to handle heterogeneous message generation rates and message acknowledgements, respectively. Section 6 validates our models with both simulation and experimental results.

5.1. Protocol Scalability

This section considers the asymptotic scalability of the end-to-end delivery delay for each anonymity protocol. We assume a lightly loaded system with N nodes.

In a lightly loaded system, the message queueing delays are negligible and can be ignored. For this case, $Q_C = 0$, and the waiting times W_S and W_T reduce to either T_C or $\frac{T_C}{2}$, depending on the protocol. With these simplifications, the expressions[8] for H_{SR} and H_T from Table 4 can be used to refine Equation (1) for each anonymity protocol:

$$D_{SR}^{Buses} = D_S + (L+1)\frac{N+1}{2}(D_{proc} + D_{net}) + (\frac{1}{2} + L)T_C + D_R \qquad (2)$$

$$D_{SR}^{Taxis} = D_S + (L+1)\frac{N+1}{2}(D_{proc} + D_{net}) + \frac{1}{2}(L+1)T_C + D_R \qquad (3)$$

$$D_{SR}^{Motor} = D_S + (L+1)\frac{\log N}{2}(D_{proc} + D_{net} + \frac{T_C}{2}) + D_R \qquad (4)$$

These expressions show how the end-to-end message delivery delay increases with the network size N, in a lightly loaded system ($Q_C = 0$). We observe the following:

- For Buses, both D_{proc} and D_{net} are proportional to N, and T_C is proportional to N^2, so D_{SR}^{Buses} is $O(N^2)$.

- For Taxis, D_{proc} and D_{net} are independent of N, while T_C is proportional to N, so D_{SR}^{Taxis} is $O(N)$.

[8]See Section 4.2.3 for a detailed discussion of H_{SR}.

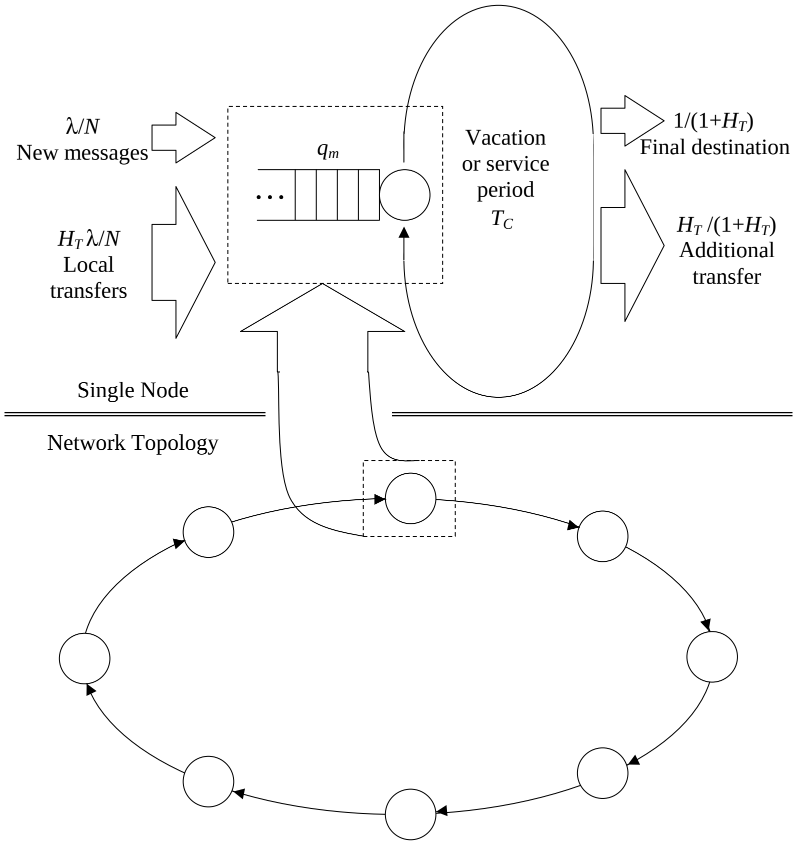


Figure 3: Queueing Model.

- For Motorcycles, D_{proc} and D_{net} are independent of N, while T_C is proportional to log N, so D_{SR}^{Motor} is $O(\log^2 N)$.

These results confirm our intuition regarding the relative latency of each protocol. In particular, Motorcycles has a significant performance advantage, especially in lightly loaded systems.

5.2. Queueing Analysis for $K = 1$

We now derive an exact closed form solution for the queueing at an arbitrary node for the single-seat ($K = 1$) case.[9] Figure 3 illustrates our queueing

[9]A discussion of the general case $K > 1$ is provided at the end of this section.

model. For simplicity, we assume that the total average (queueing) load per node is equal to $\tilde{\lambda}$. For the case when all nodes are homogenous and the load is evenly distributed across all nodes, this aggregate rate is equal to $\tilde{\lambda} = (1 + H_T)\frac{\lambda}{N}$.[10] Of this load, $\frac{\lambda}{N}$ is associated with new messages being generated by the node itself; the rest is due to message transfers (i.e., changing bus, taxi, or motorcycle at the node). Furthermore, our analysis assumes that both the message generation and the message transfer events at an individual node are Poisson. (Note that this implies that the aggregate sequence of message events is also Poisson.) While we acknowledge that this may not be true in practice, we do believe that this provides a reasonable approximation (as verified by our simulations).

We model each node as an independent queueing system with vacation times. As noted in Section 4.2.5, pending messages in the queue can only be served when a bus, taxi, or motorcycle is available at the node. Since cycle times are independent of the current forwarding load, the next (useful) bus, taxi, or motorcycle will not become available until the next cycle. For the purpose of the analysis, we refer to the time durations (T_C) between consecutive service instances as either a service period or a vacation period. From the perspective of a single node, a cycle is considered a service period if it is able to send at least one queued message; otherwise, the cycle is considered a vacation period.

Consider an (open) M/G/1 queueing system with vacation times (e.g., as in Exercise 5.23 in Kleinrock [21]). Assuming generating functions $F(z)$ and $V(z)$ for the arrivals during a service period and during a vacation period, respectively, the generating function of the queue lengths (including any jobs/messages in service) as observed at an arrival instant can be shown to be equal to $Q(z) = V(z)\frac{p_0(1-F(z))}{V(z)-z}$, where $p_0 = \frac{1-\rho}{\frac{dF}{dz}|_{z=1}}$ and $\rho = \tilde{\lambda} T_C$.

With load-independent cycle times, the service and vacation periods are identically distributed. Thus $F(z) = V(z)$. Furthermore, assuming deterministic cycle times, $V(z)$ can be calculated as $V(z) = \sum_{i=0}^{\infty} v_i z^i$, where the probability v_i that there are exactly i arrivals during a cycle time T_C is

$$v_i = \frac{(\tilde{\lambda} T_C)^i}{i!} e^{-\tilde{\lambda} T_C}. \tag{5}$$

[10]As previously noted, the Motorcycles protocol can use a round-robin implementation. In this case, each node has one queue per finger table entry, the queues are served in round-robin fashion, and the aggregate load is split among the log N queues.

Given these observations, the generating function of the queue lengths reduces to

$$Q(z) = p_0 \frac{V(z)(1 - V(z))}{V(z) - z}, \tag{6}$$

where $p_0 = (1 - \rho)/\rho$.

Given explicit expressions for the generating function of the queueing at each node, it is now possible to calculate queue statistics that can provide insights with regards to the overall system behavior for the different protocols.

Consider first the expected queue length $E[q]$ (including any jobs/messages in service). The expected queue length ($E[q] = \sum_{i=0}^{\infty} q_i i$) can be derived by differentiating the generating function, and evaluating the derivative at $z = 1$. That is,

$$\begin{aligned} E[q] &= \frac{dQ}{dz}|_{z=1} \qquad (7) \\ &= \frac{p_0 V(z)}{(V(z) - z)^2} \left[\frac{dV}{dz}(2z - \frac{z}{V(z)} - V(z)) + (1 - V(z)) \right]_{z=1}. \end{aligned}$$

Since $V(1) = 1$, applying l'Hôpital's rule twice gives

$$E[q] = \frac{\rho(3 - 2\rho)}{2(1 - \rho)}, \tag{8}$$

where $\rho = \tilde{\lambda} T_C$. The omitted intermediate steps use the fact[11] that $V(z) = e^{\tilde{\lambda} T_C (z-1)}$. This allows the m^{th} derivative $\frac{d^m V}{dz^m}|_{z=1}$ to be calculated $(\tilde{\lambda} T_C)^m e^{\tilde{\lambda} T_C (z-1)}$. Evaluated at $z = 1$, this further reduces to $\tilde{\lambda}^m T_C^m$ (or ρ^m).[12]

For the purpose of our analysis, we note that for the case when $K = 1$, as in this section,

$$Q_C = E[q] - \rho = \frac{\rho}{2(1 - \rho)}. \tag{9}$$

Note that the subtracted ρ term corresponds to the probability that some job/message currently is being served (i.e., a message is being sent from one

[11] To see this, substitute Equation (5) into the definition of $V(z)$ (i.e., $\sum_{i=0}^{\infty} v_i z^i$).

[12] These expressions can also be obtained in other ways. For example, the probabilities v_i must sum to one (i.e., $V(1) = \sum_{i=0}^{\infty} v_i$), and the first derivative evaluated at $z = 1$ (i.e., $\frac{dV}{dz}|_{z=1} = \sum_{i=0}^{\infty} v_i i$) gives the expected number of arrivals during a cycle ($\tilde{\lambda} T_C$).

node to the next).[13]

Finally, we note that the generating function can also be used to calculate explicit state probabilities and queue-size variations. For example, the probability q_0 that we find the system empty can be calculated by evaluating $q_0 = Q(0)$, while the probability that we find exactly one message in the queue is equal to $q_1 = \frac{dQ}{dz}|_{z=0}$. (In general, $q_m = \frac{1}{m!}\frac{d^mQ}{dz^m}|_{z=0}$.) Substituting $z = 0$ rather than $z = 1$ into Equations (6) and (7), we obtain $q_0 = \frac{(1-\rho)(1-e^{-\rho})}{\rho}$ and $q_1 = \frac{(1-\rho)(1-e^{-\rho}-\rho e^{-2\rho})}{\rho e^{-\rho}}$. Note that for $\rho < 0.5$ these two states account for 60% of the system state occupancy.

We do not have a closed form result for the more general case $K > 1$. While queueing models of vacation systems have generated much research, there are only limited results for systems with *both* batched service and multiple vacations. To the best of our knowledge, no explicit expressions have been derived for average waiting times in the system of interest here, namely a single-server queue with multiple vacations, batched service, and deterministic vacation (and service) periods.

Since existing results do not provide explicit expressions, and may not provide additional insights regarding our protocols, we leave the analysis of such systems as future work. However, the necessary set of equations (such as the generative equation) and a discussion of the numerical solution approaches needed to obtain the solutions are described in Lee *et al.* [22], and elsewhere. For a general discussion on queuing systems with vacation periods, please see [12, 13, 36], as well as papers on vacation periods with batch service [22, 23, 33].

5.3. Heterogeneous Traffic

We now discuss how the analysis can be extended to handle heterogeneous traffic load. While this requires some modifications to the individual terms in Equation (1), it is important to note that most of the analysis stays the same and only the two load-dependent terms (i.e., the sender-side delay and the transfer delay) are affected. In fact, assuming that the average number

[13]The above expression can also be obtained from general results for the M/G/1 queue with multiple vacations. Specifically, the number of messages in the queue (but not in service) should be $\frac{\tilde{\lambda}^2\overline{S^2}}{2(1-\tilde{\lambda}\overline{S})} + \frac{\tilde{\lambda}\overline{V^2}}{2\overline{V}}$, where S and V are the service and vacation period durations, respectively. Substituting $\overline{S^2} = \overline{V^2} = T_C^2$, $\overline{V} = T_C$, and observing that $\rho = \tilde{\lambda}T_C$, leads to the same expression.

of transfer nodes (H_T) and cycle times (T_C) are independent of the nodes generating the traffic, these terms will be affected only through the queueing delays Q_{C_i} (measured in cycle times). Of course, assuming that the load $\tilde{\lambda}_i$ on each node i can be estimated, the queueing delays can be estimated (independently) using the same queueing analysis as before.

Consider for example the case when messages are generated at rate λ_i and indirection nodes are selected uniformly at random; the average load on each node is $\tilde{\lambda}_i = \lambda_i + \frac{\lambda}{N} H_T$. Using these rates, the Q_{C_i} values can be obtained on a per-node basis using Equation (8), for example.

Finally, it is important to note that average delivery delay must be calculated on a per-message basis (rather than on a per-node basis, for example). With messages generated unevenly, messages are more likely to be sent from a node generating lots of traffic. The expression for the average sender-side waiting times (W_S) must therefore be calculated as $W_S = \sum_{i=1}^{N} \frac{\lambda_i}{\lambda} W_{S_i}$. In contrast, assuming that the transfers are evenly distributed across nodes, the average transfer time (W_T) can be calculated on a per-node basis; i.e., $W_T = \frac{1}{N} \sum_{i=1}^{N} W_{T_i}$.

5.4. Message Acknowledgements

Thus far we have considered anonymity protocols without any acknowledgements. This subsection outlines how the model can be extended to include these. For the purpose of our analysis, we assume that the receiver (R) of a message automatically generates a return (acknowledgement) message to the original sender (S), which is basically indistinguishable[14] in size, content, and delivery from a regular message.

Acknowledgements affect three components of our model:

- Acknowledgements increase the overall load $\tilde{\lambda}_i$ at each node. Part of this load comes from additional queueing at R (where an outbound message is now being generated), and part of it comes from additional indirections on the return path. Assuming that the same number of indirections are used for the return path, this increases the load $\tilde{\lambda}_i$ and utilization ρ_i by exactly a factor of 2. The waiting times W_S and W_T must be updated accordingly.

[14]There is an acknowledgement flag in the innermost core of the message returned to S.

- The number of transfer nodes W_T increases. Conceptually, R can now be considered a transfer node, since it has to wait for the next outgoing bus, taxi, or motorcycle. The modified number of transfer nodes H_T^{ack} is therefore equal to $H_T^{ack} = 2H_T + 1$. While acknowledgements roughly double the delays in the system, there are some subtle differences to consider. In particular, even for light loads, the Buses protocol causes the acknowledgement message to wait for a full cycle T_C (the worst case) before it can be sent back. At the original sender, on the other hand, a new message on average only has to wait half a cycle. By considering the receiver as a transfer node, rather than a sender, our model automatically handles this asymmetry.

- The total number of hops (H_{SRS}) traversed must be increased to include the return path from R to S. With the assumption that the return path uses the same number of indirection nodes as the forwarding path, H_{SRS} can be calculated as $H_{SRS} = 2H_{SR}$. Asymmetric paths (e.g., a lengthy forward path, and short reverse path, or vice versa) are possible, especially when L is small.

With these changes, our model extends easily to handle acknowledgements. As noted earlier, the presence of acknowledgements does not change the relative performance of the protocols.

6. Numerical Results

In this section, we validate our analytical model for the anonymity protocols, and present numerical results illustrating the performance and scalability of these protocols. We start with cross-validation of our analytical models and simulation models using small-scale experimental results. We then consider larger network scenarios using simulation and analysis. Finally, we consider the effects of protocol configuration parameters and offered load.

6.1. Experimental Results

Experimental results were collected using two different proof-of-concept implementations of the three anonymity protocols [19, 20].

The first prototype was implemented in C++ and tested in a 1 Gbps Ethernet LAN environment using a 14-node Beowulf cluster [4]. This implementation was a *full-fledged* prototype of the Buses protocol [17] and the

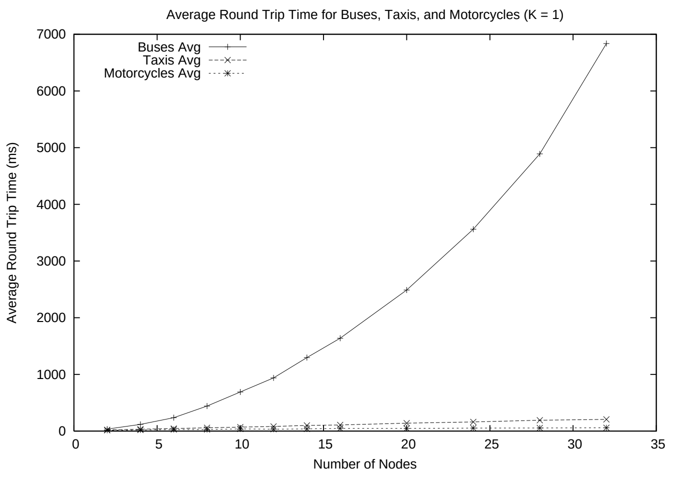
Taxis protocol [18], but not the Motorcycles protocol. The implementation included packet encryption, decryption, and other security related features. We used empirical measurements from this prototype to calibrate our analytic and simulation model parameters. See Table 5.

Table 5: Parameter Settings and Model Calibration

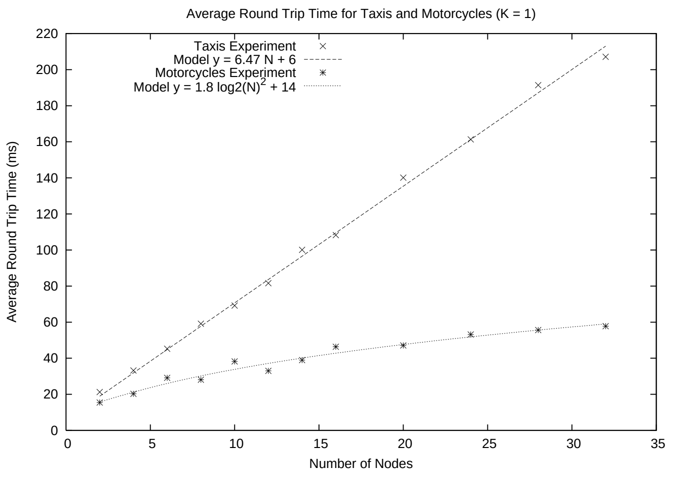
Symbol	**Definition**	**Value**
N	Number of nodes	16
K	Seats per node	1
L	Indirection layers	2
s	Seat size	3 KB
r	Network bandwidth	1 Gbps
D_S	Encoding time (Buses)	0.110 s
D_R	Decoding time (Buses)	0.110 s
D_S	Encoding time (Taxis and Motorcycles)	0.041 s
D_R	Decoding time (Taxis and Motorcycles)	0.041 s
D_{seat}	Seat processing time (Buses)	0.0018 s
D_{seat}	Seat processing time (Taxis and Motorcycles)	0.012 s

The second prototype was implemented in Java, and run on a laboratory of desktop computers connected by a 100 Mbps Ethernet LAN. This prototype is a *partial* implementation of the Buses, Taxis, and Motorcycles protocols, in that it contains all of the required networking aspects, including socket-based TCP communication, but *none* of the encryption and decryption code. Rather, all messages are exchanged in plain text. Nonetheless, this prototype provides useful insights regarding the relative performance and scalability of the three protocols.

Figure 4 shows the experimental results from the Java-based prototype. The top graph shows results for Buses, Taxis, and Motorcycles. The message latencies grow quickly (quadratically) for Buses (as has been shown in prior work [18]), while those for Taxis and Motorcycles are much lower. The bottom graph zooms in on the results for Taxis and Motorcycles. In addition, a best-fit line is drawn through each of the latter results, demonstrating that the results for Taxis are highly linear (as expected), while those for Motorcycles are consistent with $O(\log^2 N)$ growth. We have obtained similar scaling results (for Buses and Taxis) using our full-fledged prototype implementation [18, 19].



(a) Measurement results for Buses, Taxis, and Motorcycles

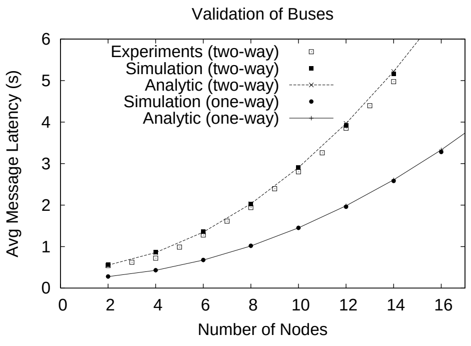


(b) Measurement results for Taxis and Motorcycles

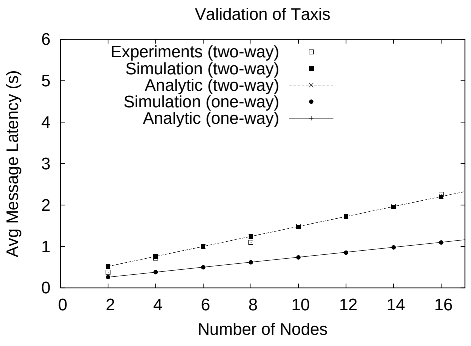
Figure 4: Experimental results for Buses, Taxis, and Motorcycles (Java)

6.2. Model Validation

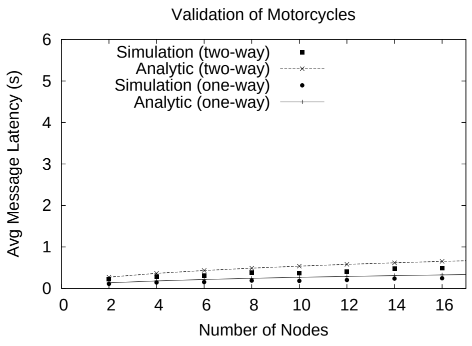
To validate our analytic model, we used our experimental results, as well as results using a discrete-event simulator for Buses, Taxis, and Motorcycles (approximately 1,000 lines of C code). We first use the small-scale experiments to validate our simulator, and then use the simulator to validate the analytic model for a wider range of scenarios. For realistic evaluation, the processing delays used at each node in the simulation and analytic models were calibrated using the experimental measurements from the full-fledged implementation.



(a) Buses



(b) Taxis



(c) Motorcycles

Figure 5: Validation using small-scale experiments and simulations.

Figure 5 shows validation results using small-scale experiments on our testbed implementation of the Buses and Taxis protocols, as well as simulations using all three protocols. For the Motorcycles protocol, for which we do not have a full-fledged implementation, we assumed that the processing overheads were the same as in the Taxis protocol (i.e., only the message routing changes). Since our implementations measured round-trip message latency, including acknowledgements, we used half of the two-way message latency as an estimate of the one-way message delays. While the primary purpose of these results is to validate the simulator, for completeness, we also show results for the analytic model.

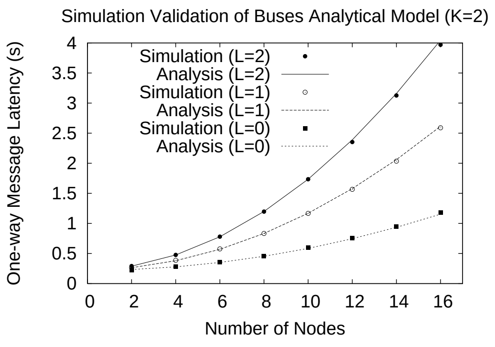
In our simulation experiments, messages are generated according to a Poisson process. The system is simulated for at least 10,000 messages, in order to capture the steady-state behavior of the system. Specifically, 12,000 messages are generated, with the statistics for the first 1,000 messages (warmup) and the last 1,000 messages (cooldown) excluded from the analysis. This methodology is used in all of our simulation experiments.

Figure 5(a) shows the validation results for Buses. The lower line and dots on the graph show the one-way message transfer delays in the analytic model and simulations, while the upper line and dots show these values scaled (doubled) to reflect round-trip message delay. The experimental results are presented as square dots on the graph. There is excellent agreement between the experimental, simulation, and analytic results, thus validating our Buses model.

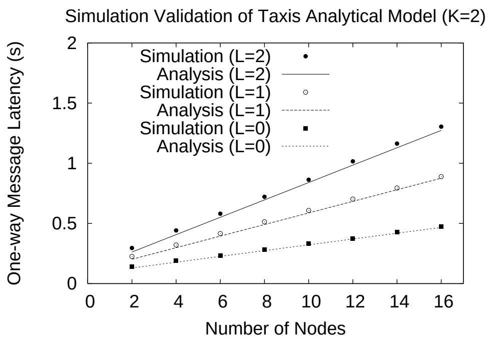
Figure 5(b) shows the validation results for Taxis. The lower line again shows one-way message latency, while the upper line shows round-trip delay. The experimental measurement results are presented as square dots on the graph. There is good agreement between the experimental and simulation results. These results provide validation for our Taxis model.

Figure 5(c) shows the corresponding validation results for Motorcycles. While the fit is not as tight as for Buses and Taxis, there is again good agreement between the analytic model, simulation, and the experimental results.

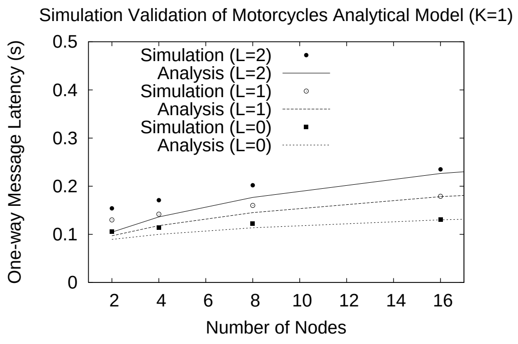
We next use the simulator to validate a wider range of scenarios. Focusing on the one-way delays, Figure 6 shows validation results for the analytical models, using simulation results using different numbers of indirection nodes. Figure 6(a) shows the one-way message latency for Buses, from our simulation (dots) and our analytical model (lines). The agreement is very close. Similar observations apply for the Taxis protocol in Figure 6(b). The validation results for the Motorcycles protocol appear in Figure 6(c). For simplicity, we only considered cases in which the number of nodes is a power of two. The qualitative agreement is good, though there are some small differences observed between the simulation and analytical results.



(a) Buses



(b) Taxis



(c) Motorcycles

Figure 6: Simulation validation of analytical models for Buses, Taxis, and Motorcycles

6.3. Heterogeneous Traffic

Using our analytic and simulation models, we next explore the effects of heterogeneous traffic. The corresponding analytical model for this scenario was developed in Section 5.3.

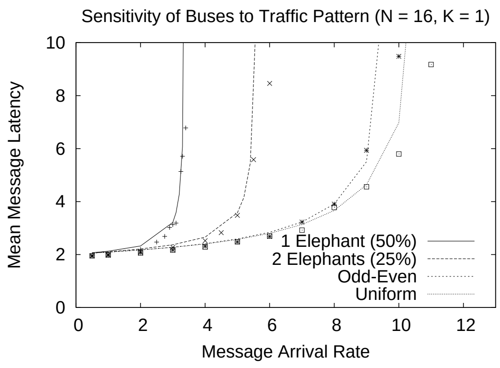
Figure 7 shows results for four example scenarios, with analytic results plotted using lines, and simulation results plotted as points. We consider a uniform traffic scenario as our baseline, and then study three non-uniform traffic scenarios. In the simplest one, even-numbered nodes generate twice as much traffic as the odd-numbered nodes. In the second scenario, there are two highly active nodes, each generating 25% of the overall traffic, while the remaining nodes uniformly generate the rest of the traffic. In the most extreme scenario, there is a single heavy-hitter in the network, generating 50% of the aggregate traffic. In all cases, the aggregate traffic load is the same, and traffic destinations (and indirection nodes) are chosen uniformly at random. The network size is $N = 16$, with $K = 1$ and $L = 2$.

The results in Figure 7 show the expected behaviour. Message latencies remain quite reasonable until the system nears saturation, at which point the latencies rise abruptly. The odd-even scenario saturates slightly sooner than the uniform traffic scenario, while the heavy-hitter scenarios saturate much sooner. The single heavy-hitter has the most extreme results, since the saturated node becomes a bottleneck for all messages circulating on the Bus or Taxi tour of the network. The impact of the heavy-hitter is less pronounced for Motorcycles, since not all routes traverse the saturated node.

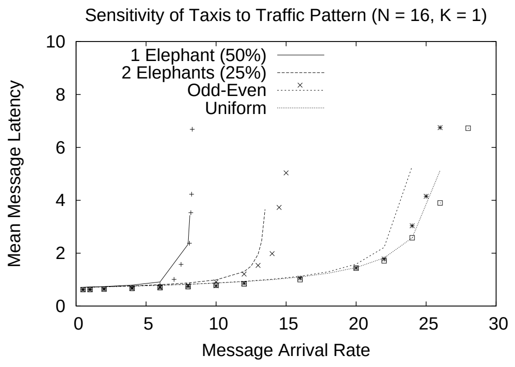
We note that the results using the analytic model and the simulation model are qualitatively very similar. While there are some quantitative differences in the average message delays for a given arrival rate, the relative differences in the saturation points for the different traffic scenarios are well captured. The agreement is very good for the Buses and Taxis protocols, but less accurate for the Motorcycles protocol, particularly for certain traffic patterns.

One particular limitation of the analytical model is that it ignores temporal and spatial correlations that can occur in the traffic, particularly for the Motorcycles protocol. To illustrate this effect in more detail, we next examine the Motorcycles traffic scenario with two heavy-hitters. While the first heavy-hitter can be placed arbitrarily at node 0, the location of the second heavy-hitter can have a great impact on the performance. Figure 8 illustrates this effect. Here, the second heavy-hitter is placed in turn at nodes 1, 2, 4, and 8, which happen to be the finger table entries for node 0 in a network of size $N = 16$. We note that the message latency is much worse with node 8 as the second heavy-hitter, and that node 4 is the second-worst choice.

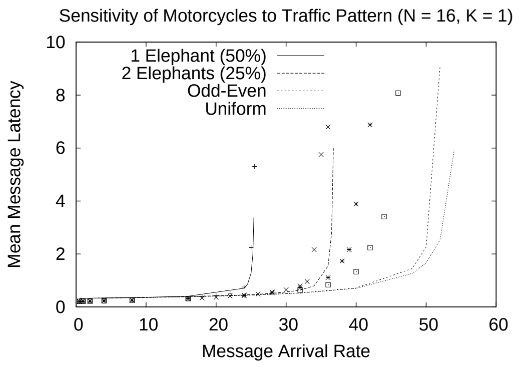
This performance degradation is due to high correlation between the queue lengths in the Motorcycles protocol. More precisely, the structure of the finger tables will cause node 0 to use node 8 as the next forwarding node for *half* of the possible destinations. Similarly, node 8 will use node 0 as the forwarding node for half of its destinations. Clearly, this will further increase the load on these two nodes. In contrast, node 4 will only receive one-quarter of the traffic from node 0, and node 0 will never receive traffic



(a) Buses Analysis and Simulation



(b) Taxis Analysis and Simulation



(c) Motorcycles Analysis and Simulation

Figure 7: Heterogeneous traffic results for Buses, Taxis, and Motorcycles

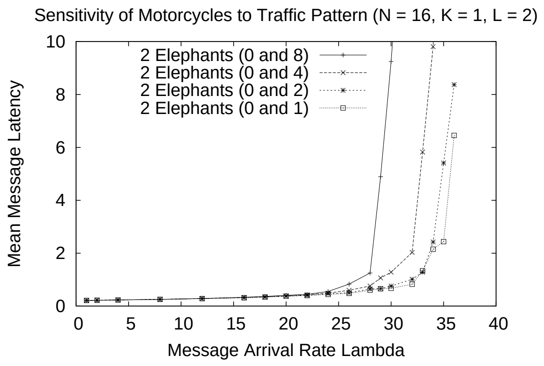


Figure 8: Sensitivity of Motorcycles Protocol to Elephant Placement

directly from node 4. Clearly, placing the second heavy-hitter on node 8 is the worst-case scenario.

The analytic model does not capture any of these second-order effects, and therefore predicts the same performance, independent of which node is chosen as the second heavy-hitter. For Buses and Taxis, the choice of the second heavy-hitter has negligible impact on the performance, and such assumptions are therefore okay. However, as illustrated above, node selection can impact the performance of the Motorcycles protocol, and hence limits the accuracy of the Motorcycles model.

This observation provides new insights for interpreting the results in Figure 7(c) for the Motorcycles model. In particular, the analytic model fit is actually very good for the single heavy-hitter, and even for two heavy-hitters, which are placed at node 0 and node 5 in these experiments. For the odd-even scenario, structural correlations again degrade the accuracy of the analytical model. In particular, since the finger table entries are predominantly even (i.e., only the very first entry is odd), this traffic scenario tends to cause the "busy" even-numbered nodes to forward their traffic to other "busy" nodes, even though half of the nodes in the network (i.e., the odd-numbered ones) are lightly loaded. The correlated queues result in degraded message delivery. Finally, the uniform traffic scenario tends to degenerate in this way as well. In essence, this is a manifestation of the "birthday paradox", wherein transient queues tend to build up on some (temporarily popular) nodes in the network, and then the queues percolate elsewhere in the network in a structured fashion (based on the finger table), before they gradually dissipate. In the simulation results, this effect manifests itself in pronounced short-term correlation in the message delivery latencies. As the load nears saturation, this phenomenon degrades message delivery far more than predicted by the analytical model, which is based on an independence assumption.

6.4. Scalability Results

In the previous subsections, we validated our analytic models using small-scale experiments and simulations. In this section, we use the analytic model to study scalability issues. While the queueing model is limited to the one-seat per node case, or the light-load scenarios (in which case the queuing delays are small), the fast computation times offered by the analytic model allow us to explore questions that would require many time-consuming experiments.

Figures 9 through 12 provide examples of protocol scalability results from the analytical models. These results all use the same parameter settings as in Sections 6.3 and 6.4.

Figure 9 shows results for small networks. As the message arrival rate is increased, the Buses protocol saturates first, then the Taxis protocol, and finally the Motorcycles protocol. The advantages of Taxis over Buses are evident even on small networks ($N = 4$), while Motorcycles are clearly superior when the network is larger. Our analytical models provide a convenient way to determine the saturation point, which is important in system capacity planning.

Figure 10 shows results for different load levels, on networks ranging from $N = 4$ to $N = 256$ nodes. The graphs show how the average message latency increases with load. Queueing delays start to dominate when $\rho > 0.8$. The delays are extremely high for Buses, tolerable for Taxis, and modest for Motorcycles. Also recall that the Motorcycles protocol can sustain a much higher message generation rate, for a given network utilization, so the user-perceived performance differences between these protocols are even more pronounced than illustrated here (note that the vertical axis is logscale).

Figure 11 shows results for the light load case on larger networks. The graphs show how the average message latency increases with the number of nodes for Buses, Taxis, and Motorcycles. The growth is quadratic, linear, and logarithmic, as expected, regardless of the number of seats.

Figure 12 shows results for different load levels on larger networks. The graphs show how the average message latency increases with load, as queueing delays start to dominate. Fortunately, the relative performance differences observed between the anonymity protocols at light load are maintained at higher loads. That is, the scaling behaviors for the three protocols are consistent across a wide range of ρ values. These results show that Motorcycles provide a robust and scalable approach for anonymous network communication.

7. Conclusions

In this paper, we focus on the performance and scalability of anonymous network communication protocols. In particular, we develop end-to-end message latency models for three anonymity protocols: Buses, Taxis, and Motorcycles. The latter is a new anonymity protocol proposed in this paper.

Using a combination of analytical, experimental, and simulation results, we show that the message latency of the Buses protocol scales quadratically with the number of participants, while that of the Taxis protocol scales linearly, and that of Motorcycles scales logarithmically with the network size. The analytic models are validated with simulation results and experimental results from prototype implementations of the three anonymity protocols. Motorcycles provides scalable anonymous network communication, without compromising the strong anonymity provided by Buses and Taxis.

Acknowledgements

The authors thank the reviewers for their constructive feedback and suggestions, which helped to improve the clarity of the final paper. We are also grateful to Ibrahim Ismail, who implemented the first-ever prototype of the Motorcycles protocol for his M.Sc. thesis [20]. Financial support for this work was provided by Canada's Natural Sciences and Engineering Research Council (NSERC), and by the Informatics Circle of Research Excellence (iCORE) in the Province of Alberta.

References

[1] Anonymity bibliography, 2005. `http://www.freehaven.net/anonbib/`.

[2] Anonymizer, 2003. `http://www.anonymizer.com`.

[3] A. Beimel and S. Dolev. Buses for anonymous message delivery. *Journal of Cryptology*, 16(1):25–39, 2003.

[4] Beowulf cluster, 2007. http://www.beowulf.org/overview/.

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

[6] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba. Performance evaluation of an anonymity providing protocol for wireless ad hoc networks. *Performance Evaluation*, 63(11):1094–1109, November 2006.

[7] D. Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[8] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[9] J. Claessens, B. Preneel, and J. Vandewalle. Solutions for anonymous communication on the Internet. In *Proceedings of the 33rd IEEE Annual International Carnahan Conference on Security Technology (ICCST)*, pages 298–303. Madrid, Spain, October 1999.

[10] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy (S&P)*, pages 2–15. Berkeley, CA, May 2003.

[11] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–320. San Diego, CA, USA, August 2004.

[12] B. Doshi. Queueing Systems with Vacations - A Survey. *Queueing Systems*, vol. 1, February 1986, pp. 29–66.

[13] B. T. Doshi. Single Server Queues with Vacations. *Stochastic Analysis of Computer and Communication Systems (Ed. H. Takagi)*, North-Holland, Amsterdam, 1991.

[14] M. 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), pages 193–206. Washington, DC, USA, November 2002.

[15] Y. Guan, X. Fu, R. Bettati, and W. Zhoa. An optimal strategy for anonymous communication protocols. In *Proceedings of the IEEE 22nd International Conference on Distributed Computing Systems (ICDCS)*, pages 257–266. Vienna, Austria, July 2002.

[16] C. Gulcu and G. Tsudik, "Mixing E-mail with Babel", *Proceedings of the IEEE Symposium on Network and Distributed System Security*, pp. 2-16, 1996.

[17] A. Hirt, J. Michael Jacobson, and C. Williamson. A practical buses protocol for anonymous Internet communication. In *Proceedings of the Third Annual Conference on Privacy, Security, and Trust (PST)*, pages 233–236. St. Andrews, NB, Canada, October 2005.

[18] A. Hirt, J. Michael Jacobson, and C. Williamson. Taxis: scalable strong anonymous communication. *Proceedings of the IEEE/ACM 11th International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS)*, Baltimore, MD, USA, October 2008.

[19] A. Hirt. *Strong Scalable Anonymous Network Communication.* Ph.D. Thesis, University of Calgary, March 2010.

[20] I. Ismail, *Analysis and Evaluation of Anonymity Protocols.* M.Sc. Thesis, University of Calgary, January 2011.

[21] L. Kleinrock. Queueing systems: Theory, Vol. I. Wiley, New York, NY, USA, 1976.

[22] H. W. Lee, D. Chung, S. S. Lee, and K. C. Chae. Server Unavailability Reduces Mean Waiting Time in Some Batch Service Queuing Systems. *Computers and Operations Research*, vol. 24, iss. 6, June 1997, pp. 559-567.

[23] H. W. Lee, S. S. Lee, K. C. Chae, and R. Nadarajan. On a Batch Service Queue with Single Vacation. *Appl. Math. Modelling*, vol. 16, Jan. 1992, pp. 36–42.

[24] Lucent Personal Web Assistant, `http://www.math.tau.ac.il/matias/lpwa.html`

[25] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster Protocol — Version 2. Draft, July 2003.

[26] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity: A proposal for terminology. Draft, version 0.31, February 2008.

[27] A. Pfitzmann and M. Waidner. Networks without user observability. *Computers & Security*, 2(6):158–166, 1987.

[28] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous Connections and Onion Routing", *Proceedings of IEEE Symposium on Security and Privacy*, pp. 44-54, 1997.

[29] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[30] 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)*, pages 91–102. Washington, DC, USA, November 2002.

[31] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, "P5: A Protocol for Scalable Anonymous Communication", *Proceedings of IEEE Symposium on Security and Privacy*, pp. 53-65, 2002.

[32] C. Shields and B. Levine, "A Protocol for Anonymous Communication over the Internet", *Proceedings of ACM Conference on Computer and Communication Security*, pp. 33-42, 2000.

[33] K. Sikdara, and U. C. Gupta. On the Batch Arrival Batch Service Queue with Finite Buffer under Server's Vacation: MX/GY/1/N Queue. *Computers and Mathematics with Applications*, vol. 56, iss 11, Dec. 2008, pp 2861–2873.

[34] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, February 2003.

[35] P. Tabriz and N. Borisov. Breaking the collusion detection mechanism of MorphMix. In G. Danezis and P. Golle, editors, *Proceedings of the Sixth Workshop on Privacy Enhancing Technologies (PET)*, Springer, pages 368–384. Cambridge, UK, June 2006.

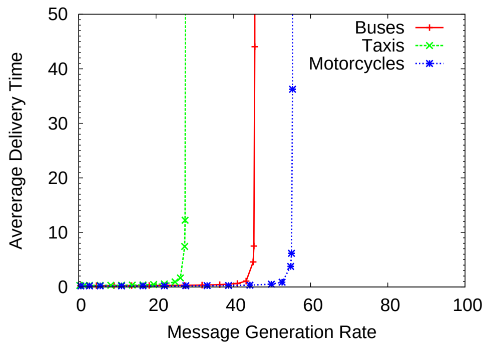
[36] H. Takagi. *Queueing Analysis: A Foundation of Performance Evaluation, Volume 1: Vacation and Priority Systems.* Elsevier Science Publishers, Amsterdam, 1991.

A. Chord Routing Analysis

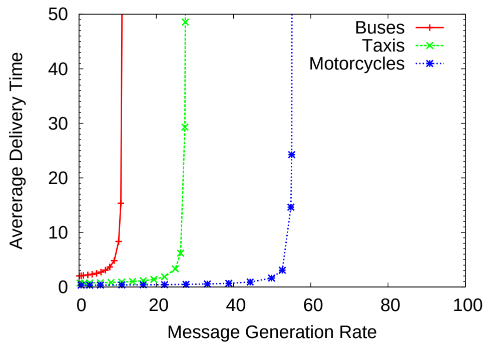
This section presents our analysis of the average path length from a sender to a receiver on a Chord ring. For simplicity, we assume that a Chord message is never sent to the node itself, and the next indirection node (or receiver) is 1, 2, ..., or $N-1$ node ids away. This differs slightly from our assumptions in the analysis of Buses and Taxis.

In a Chord ring, where N is a power of 2, the number of nodes that are distance h from the sender is equal to $\binom{\log N}{h}$. For this case, we can hence calculate the average number of hops per indirection as $\sum_{h=1}^{\log N} h\binom{\log N}{h}$. This expression has been empirically verified by comparison against the average path distance, as calculated over *every* possible path, for each Chord ring of size 2 through 2^{20}. The match is exact for all such rings.

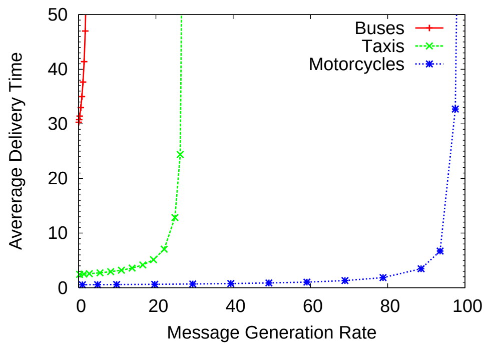
In our Motorcycles model, we use $\frac{\log N}{2}$ as our approximation. Comparison against the exact values shows that this is a very good approximation, especially for large N. For example, with N equal to 16, 256, and 4,096 the errors are 6.25%, 0.39%, and 0.02%, respectively.



(a) $N = 4$

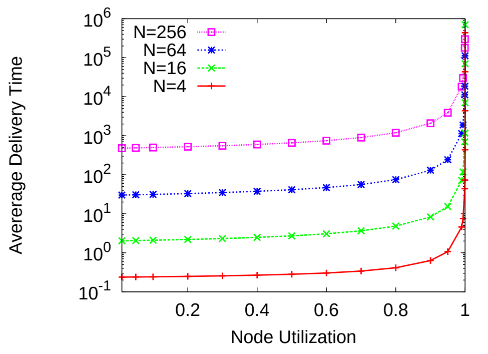


(b) $N = 16$

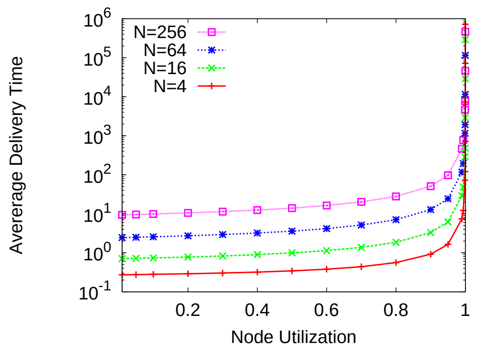


(c) $N = 64$

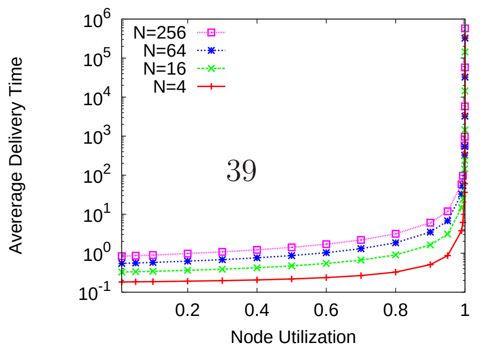
Figure 9: Impact of message generation rate λ for different N.



(a) Buses

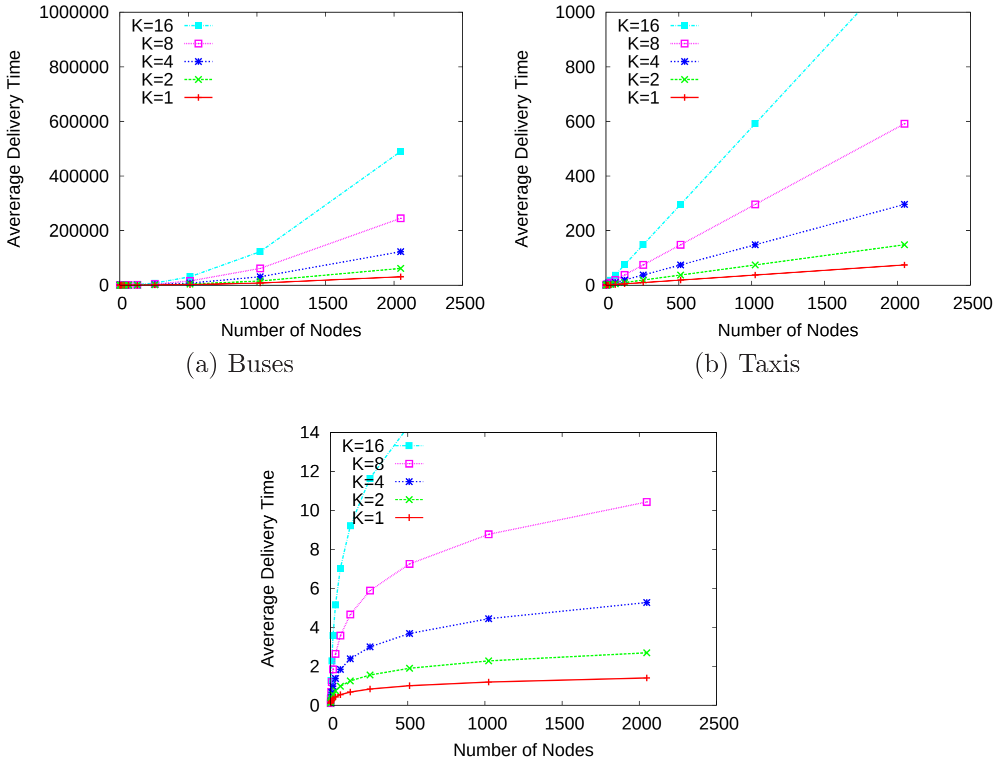


(b) Taxis



(c) Motorcycles

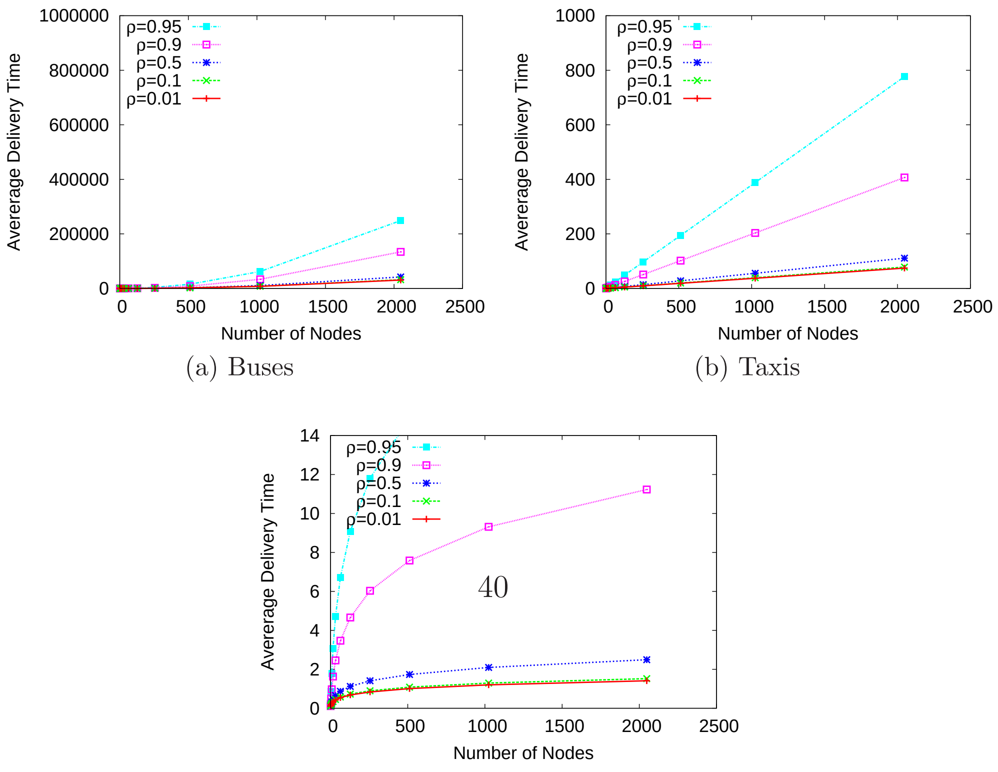
Figure 10: Impact of node utilization for different N.



(a) Buses (b) Taxis

(c) Motorcycles

Figure 11: Scaling results for light load with K seats per node.



(a) Buses (b) Taxis

(c) Motorcycles

Figure 12: Scaling results for different load levels ($K = 1$, $L = 2$).