SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2007; **37**:1515–1538
Published online 1 May 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/spe.816

A study of e-mail patterns

Sam Shah*,† and Brian D. Noble

Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2121, U.S.A.

SUMMARY

Although electronic mail is an increasingly important service, there are few empirical studies of e-mail traffic. We have observed over 2.85 million messages passing through our departmental servers over the course of seven months, and derived distributions that approximate several important e-mail parameters including message sizes, message senders and receivers and the burstiness of message deliveries. Our work is unique in that we also analyse message payloads: attachment content types, e-mail redundancy, and the use of e-mail as a sharing mechanism. These data can be used in developing e-mail workloads for mail system engineering or benchmarking. To this end, we provide an improved version of Postmark, a small-file Internet benchmark, that better approximates mail server characteristics. Copyright © 2007 John Wiley & Sons, Ltd.

Received 26 April 2006; Revised 11 January 2007; Accepted 14 January 2007

KEY WORDS: e-mail characterization; workload characterization; analytical modeling; electronic mail benchmark

1. INTRODUCTION

E-mail is one of the prevailing methods of communication: 60% of Americans use e-mail on the job; translating to about 57 million adults in 2002 [1]. However, few empirical studies have attempted to quantify this important service, especially with regards to the examination of message content.

E-mail has also become a *de facto* file-transfer mechanism. Users often send vacation photos to their friends and families via e-mail rather than uploading them to a Web site and sending a link to that site. Colleagues often make revisions to a document in a collaborative manner using e-mail as a version-control system. The simplicity of e-mail makes it the preferred vehicle for transport.

This begs several questions. Does this kind of collaboration occur? What kinds of content payloads are being transmitted? Is there any redundancy in this content? What's the lifetime of this content? What's the typical message size? What access pattern does e-mail delivery follow?

In this paper, we attempt to answer these questions by tracing our department's mail server, examining about 2.85 million messages over seven months. We found distributions that closely

*Correspondence to: Sam Shah, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2121, U.S.A.
†E-mail: shahsam@umich.edu

approximate message sizes, the number of recipients, message delivery burstiness, payload content types, data redundancy and content lifetimes.

These empirical data are useful in modeling e-mail workloads. For example, Porcupine [2] uses a cluster of ordinary PCs to provide a replicated, high-performance, available e-mail service. To evaluate this service, Saito *et al.* [2] used a synthetic e-mail workload with 'a mean message size of 4.7 KB, with a fairly fat tail up to about 1 MB'. Our empirical analysis found that message sizes were indeed heavy tailed, but had evidence of infinite variance. This causes extreme skew in the distribution: the mean message size is 17.9 KB. Furthermore, a maximum message size of 1 MB may not have properly modeled the outliers in the message size distribution, yielding incomplete results. While these distributions are likely to change over time as broadband network deployments increase, this study provides a useful starting point.

As an application of our analysis, we evaluate the Postmark benchmark [22]. Postmark, designed to measure small-file Internet server applications such as electronic mail, performs small reads, appends and deletes throughout a populated file system to emulate an e-mail workload. Postmark exclusively performs small appends, mimicking message delivery, by drawing from a small, uniform distribution. However, changing Postmark's message recipient and size distributions to match observed behavior has a measurable impact on its performance; the more accurate benchmark is approximately 20% slower in our experiments. Worse, one can optimize for Postmark in ways that will not improve actual e-mail performance.

2. RELATED WORK

There have been few characterizations and analyses of e-mail workloads. SPECmail2001, the industry standard mail server benchmark, provides a white paper detailing some of its benchmarking parameters [3], but lacks analytical distributions.

Bertolotti and Calzarossa [4,5] performed a characterization of mail server workloads, focusing only on message sizes, interarrival times and the number of recipients per e-mail. Their distributions lacked heavy-tail analysis.

In their study of spam, Gomes *et al.* [6] studied message arrivals, sizes and recipient distribution properties over eight days; our study considered seven months of e-mail traffic. Some of our results are contradictory: we found a Pareto heavy tail in message sizes and arrival times, while they found that message sizes exhibit a log-normal heavy-tail and that arrival times lack any heavy-tail behavior. While Gomes *et al.* were focused on SPAM delivery characteristics, our focus is rather on the content of message bodies.

More importantly, our study is more comprehensive than any of those previously mentioned in that we also analyse the content of messages, providing data on content types and data lifetimes of message payloads.

3. COLLECTING THE DATA

3.1. Basics of mail transfer

Figure 1 shows the interactions between mail components in a typical mail transaction when Alice sends a message to Bob. Alice's client contacts its *message transfer agent* (MTA), typically `sendmail`, which relays her message over the *Simple Mail Transfer Protocol* (SMTP) [7] to the

DOI: 10.1002/spe

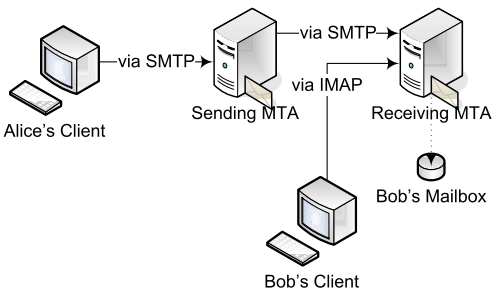

Figure 1. Interactions between mail subsystems in a typical mail transaction.

destination MTA, which stores it to disk. Bob's client reads his mailbox via the *Internet Message Access Protocol* (IMAP) [8] or, alternatively, the older *Post Office Protocol* (POP3) [9], and presents the message to Bob.

SMTP is a plain-text, unauthenticated protocol that is used to send mail. A sample SMTP session is shown in Figure 2. The protocol is rather self-evident. Multiple recipients of a message can be specified as multiple *envelope to* commands or can be separate mail transactions.

It is important to note that the message's header is completely unrelated to and unverified against the envelope's contents. In other words, the message header could contain different sender and recipient information to that given in the actual envelope contents.

SMTP is closely related with the *Multi-purpose Internet Mail Extensions* (MIME) specification [10]. Virtually all Internet e-mail is transmitted in the MIME format. Each MIME message can contain one or more *parts*. These parts could specify alternate versions of the message's textual body (e.g. plain text or HTML) or specify the inclusion of attachments. Each part has an associated content type (e.g. 'text/plain' for simple text). Furthermore, since SMTP is a 7-bit protocol, each MIME part specifies an encoding rule for 8-bit binary content; the most common of these is base64.

3.2. Content-based indexing

An ideal method of evaluating the redundancy in e-mail would be to log each message and perform pairwise deltas. However, since we would be storing payloads, we would violate privacy requirements at our institution. In addition, we would like to find similarity among message attachments while being agnostic to the underlying structure of the data. A scheme that has low spatial requirements is also attractive.

To fill this need, we chose content-based indexing [11]. The basic premise of content-based indexing is to partition a byte stream into boundary regions, called *anchors*, using Rabin fingerprinting [12]. An offset is marked as anchor if the low-order bits of a Rabin fingerprint match a predetermined value.

These anchors divide a byte stream into *chunks*. These chunks are a function of the content of the file; editing a byte stream will only modify its respective chunk. Each chunk is of variable size as the distance between anchors is variable. A chunk is represented as a pair: the content's SHA-1 hash and

DOI: 10.1002/spe

```
1  220 nobody.com ESMTP [...]
2  HELO A
3  250 nobody.com Hello mail [127.0.0.1]
4  MAIL FROM: sender@mail.com [ENVELOPE FROM]
5  250 2.1.0 sender@mail.com... Sender ok
6  RCPT TO: recipient-a@nobody.com [ENVELOPE TO]
7  250 2.1.5 recipient-a@nobody.com... Recipient ok
8  RCPT TO: recipient-b@nobody.com [ENVELOPE TO]
9  250 2.1.5 recipient-b@nobody.com... Recipient ok
10 DATA
11 354 Enter mail, end with "." on a line by itself
12 From: Nobody <sender@mail.com>
13 Message-Id: <2004171916.GA132@mail.com>
14 Subject: hello
15 To: Recipient A <recipient-a@nobody.com>
16 Cc: Recipient B <recipient-b@nobody.com>
17 Date: Sun Apr 18 15:34:49 EDT 2004
18
19 Hello!
20 .
21 250 2.0.0 g9JMqs600 Message accepted for delivery
22 QUIT
23 221 2.0.0 nobody.com closing connection
```

Figure 2. A sample SMTP session. The italic type indicates server responses and the boxed contents denote typical names for parts of the transaction.

its size. Owing to the collision-resistant nature of SHA-1, it is assumed that two chunks that hash to the same value are in fact the same data. A multiset[†] of chunks forms a *signature* of the byte stream.

For example, the two streams F_1 and F_2 shown in Figure 3 are broken into three and two chunks, respectively, through Rabin fingerprinting, each chunk being identified by its SHA-1 hash. The multiset $\{h_1, h_2, h_3\}$ represents the signature of F_1, while $\{h_4, h_3\}$ represents the signature of F_2. The set intersection of the signatures of F_1 and F_2 indicates that they have chunk h_3, and its associated bytes, in common.

There are several attractive features of this scheme. First, since a one-way hash function is used, it is impossible to reconstruct the payload from its signature. Second, this scheme is indifferent to a stream's underlying representation: it is only considered a byte sequence. Third, content-based indexing has the following two additional properties.

1. Non-propagation: the signature corresponds linearly with the changed part of the file, meaning that small changes in a file will leave most of the signature the same.
2. Alignment robustness: byte shifts in the file do not affect the signature.

Finally, since each signature is small, it is easily stored in a database.

[†]While the list of chunks is ordered and may contain duplicates, for our analysis we are indifferent about order and can represent the signature as a multiset.

DOI: 10.1002/spe

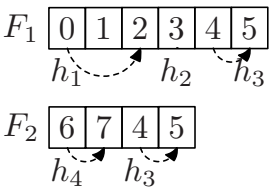

Figure 3. Example of content-based chunking of two streams.

3.3. Mail instrumentation

To instrument mail, we captured messages received on our department's mail server. At the time, our departmental mail server supported about 250 users, virtually all faculty and graduate students in the department.

We placed a dedicated collection machine on the same Ethernet segment and sniffed messages as opposed to directly capturing messages on the server as not to increase system load and disturb e-mail delivery. To capture packets and reconstruct mail sessions, we used a daemon built on `libpcap` to capture mail traffic on port 25 (SMTP).

The message is parsed as follows: the SMTP envelope MAIL FROM (also called envelope from) and RCPT TO addresses (envelope recipient) are anonymized to maintain privacy; the message body is MIME-parsed and chunked; and the resulting chunks and their MIME types, as well as the anonymized from and to addresses and a timestamp, are stored in a database. The anonymizing procedure consists of taking the SHA-1 hash of the data.

We use the envelope from and envelope recipient instead of parsing the message body's from and to headers as these are the most accurate: they also capture blind carbon-copies. Since mail is unauthenticated we cannot discern senders, but for our purposes we do not care.

The message body is parsed such that each attachment—a MIME boundary—is chunked individually. This is necessary as each MIME part is informationally distinct from the previous one and in multi-attachment messages the order of attachments may change. We attempt to discern the type of attachment based on the file name when a generic octet-stream type or a non-IANA[‡] compliant MIME type is found.

We also run SpamAssassin 2.43, a filtering tool that outputs a number based on its interpretation of the likelihood of a message being unsolicited commercial e-mail (spam).

3.4. Missing data

Since our departmental mail server allows SSL/TLS connections, we cannot sniff those connections. Secure connections account for 12.9% of all deliveries.

[‡]IANA, the Internet Assigned Names Authority, is responsible for maintaining the directory of MIME content types.

DOI: 10.1002/spe

Furthermore, many local machines are configured not to use the departmental SMTP server, but rather a local installation of `sendmail`. It is not practical to quantify how many machines do so.

4. DATA ANALYSIS

The trace data collection ran for more than seven months from 27 November 2002 until 9 July 2003. In that time, we instrumented 2.85 million messages totaling 50.5 GB.

The term *message* within this context must be properly defined. If the same message is sent to r recipients within one transaction, it is designated as r messages. This is because the same r messages could have been as easily sent as separate transactions and will eventually be stored separately.

Furthermore, since messages, particularly attachments, are encoded, for simplicity we use term *size* hereinafter to mean the encoded size: the actual bytes stored on disk and transmitted over the wire. All attachments seen were base64-encoded. As base64 uses a simple translation table, the 8-bit data size is approximately three-quarters of its encoded size.

We wish to provide simple, analytical models for some properties of e-mail. We use a quasi-Newton search [13], a method for minimizing mean square differences, to find parameters fitting the empirical data. These are approximations of the empirical data, not necessarily governing distributions.

For those unfamiliar, the coefficient of determination, termed r^2, is a measure of correlation: it represents the fraction of variance explained by the proposed model. For example, an r^2 of 0.90 indicates that the model accounts for 90% of the variance in the data, signaling a very good fit.

4.1. Message size

Figure 4 shows a cumulative distribution function (CDF) of message sizes. Since the distribution is symmetric under a log scale, we deduce that message sizes are log-normally distributed with parameters $\mu_l = 7.63$, $\sigma_l = 1.68$ ($r^2 = 0.934$). The median message size is 1901 bytes with first and third quartiles of 758 and 5092 bytes, respectively. The smallest message was the empty message, where the largest message was 44.8 MB.

Figure 5 shows a log–log complementary distribution (LLCD) of message sizes. The LLCD is the natural logarithm of the complementary CDF against log-scaled message size. Linear behavior is seen from the 70% quantile (message sizes of 4 KB and greater) and onwards. This indicates a significant Pareto-like heavy tail [14] with $\alpha = 1.83$ ($r^2 = 0.911$).

The power-law heavy tail is in contrast with other work. Gomes *et al.* [6] found that message sizes can be represented by log-normal distributions at the body *and* at the tail. However, our analysis shows that the tail more closely follows a power law; a log-normal tail provides a poorer fit ($r^2 = 0.811$). Bertolotti and Calzarossa [4,5] failed to analyse any heavy-tailed characteristics.

A Pareto distribution with $0 < \alpha \leq 2$ indicates that the distribution has infinite variance. Thus, assuming a log-normal distribution of message sizes when designing systems will erroneously lead to systems where heavy-tailed message length could cause resource misallocation. To correctly model e-mail traffic, we advocate a piece-wise model with a log-normal body ($\mu_l = 7.63$, $\sigma_l = 1.68$) censored at 4 K (0.70 quantile) with a Pareto tail ($\alpha = 1.83$).

Most bytes in the message spool come from large messages, as Figure 6 attests.

DOI: 10.1002/spe

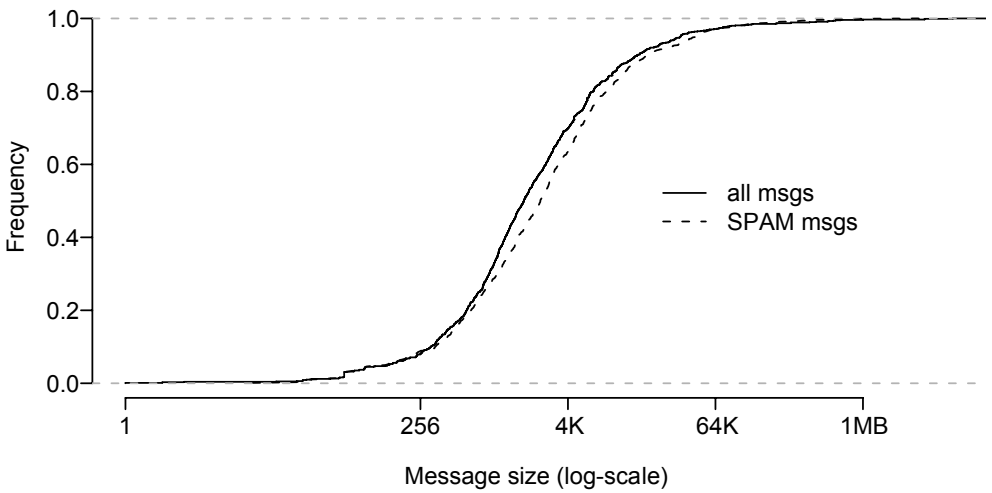

Figure 4. Message size CDF

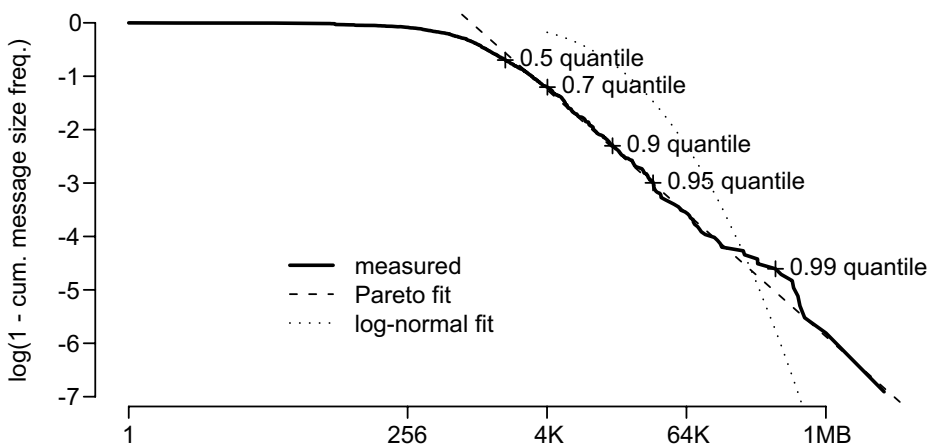

Figure 5. A LLCD of messages sizes.

DOI: 10.1002/spe

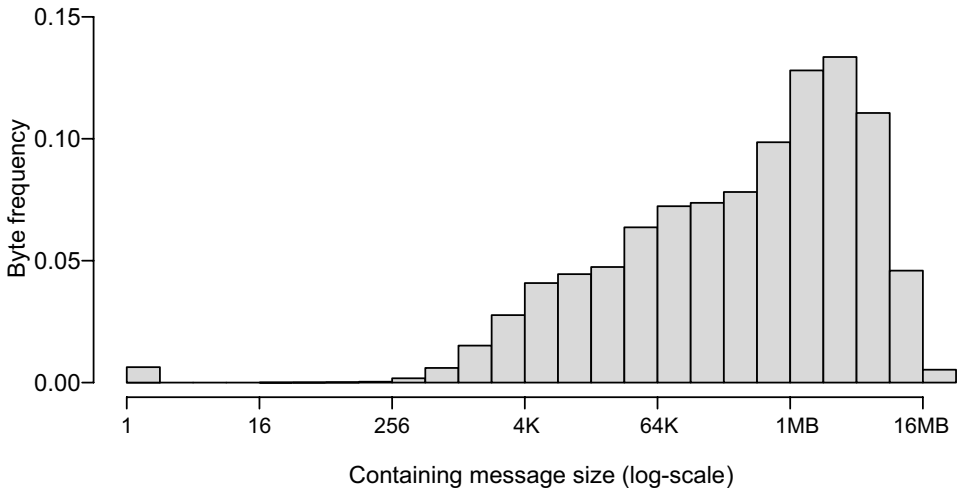

Figure 6. A histogram of bytes by message sizes.

4.2. Unsolicited commercial e-mail (spam)

To measure spam, we run SpamAssassin 2.43 with its default configuration on each message. A message is considered spam if its hit-count threshold, an arbitrary measurement of the likelihood that a message is spam, reaches above a certain level. For the default configuration, this is 5. In our study, about 20% of messages were categorized as spam. This number is low as SpamAssassin was not updated during the tracing period; spam filters must continually update their heuristics as spam messages are designed to bypass them.

The spam message size is also log-normally distributed, as shown in Figure 4, but, interestingly, a spam message is typically larger than a non-spam message ($\mu = 8.28$, $\sigma = 1.18$, $r^2 = 0.945$). The median spam message size is 3814 bytes with first and third quartiles of 1794 and 8006 bytes, respectively. Spam message sizes also exhibit a Pareto-like heavy tail with $\alpha = 1.86$ ($r^2 = 0.971$) from about 4 KB onwards.

The result that spam messages are larger than non-spam messages contradicts results presented in Gomes *et al.* [6], where they discovered that spam messages are typically much smaller than non-spam messages. This discrepancy may be due to an artifact of the short 8 day trace period in their study or because of our long trace lacking a spam heuristic update.

4.3. Receivers and senders

Figure 7 shows the rank of the number of messages received per user. Notice that most messages are received by a few users, following a Zipf distribution with $\theta = 0.753$ ($r^2 = 0.961$). Figure 8 shows the number of messages ranked against a sender's address. Ignoring the messy tail, the distribution is Zipf

DOI: 10.1002/spe

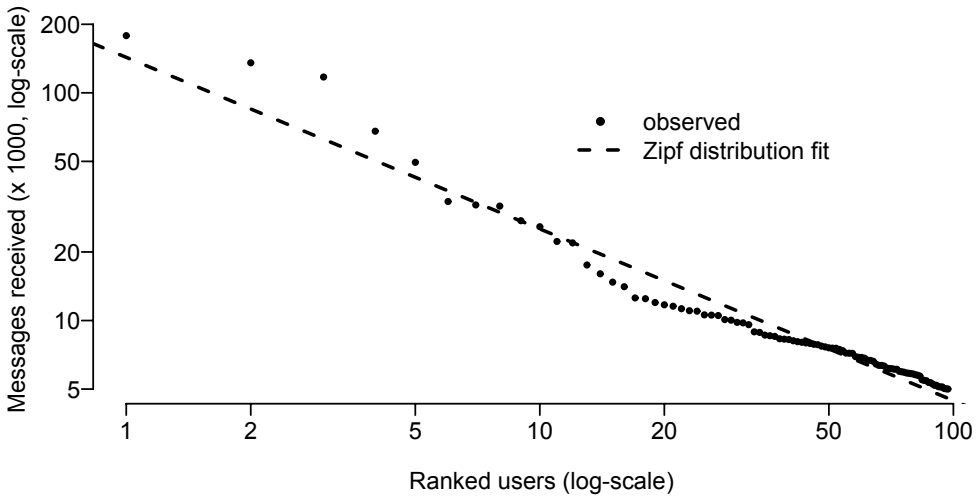

Figure 7. The number of messages received by each user.

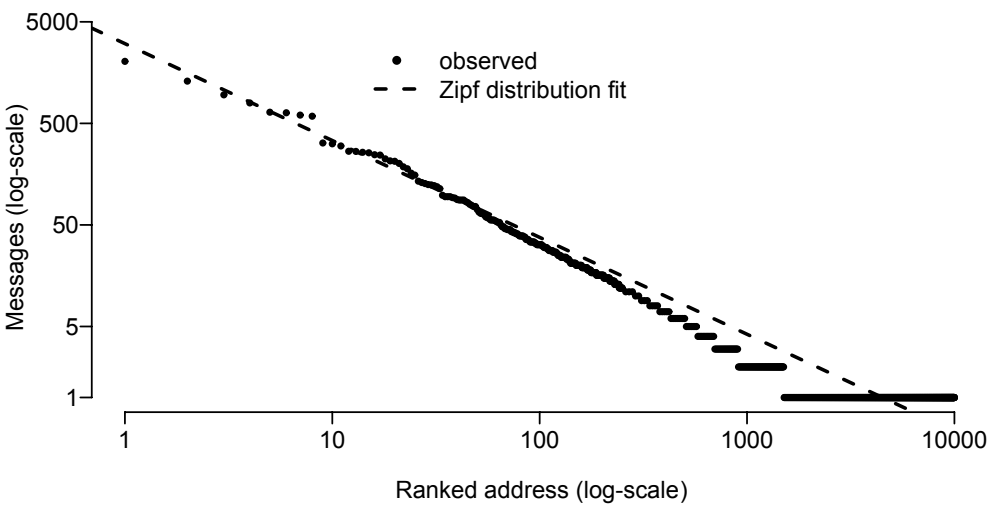

Figure 8. Incoming messages received by address.

DOI: 10.1002/spe

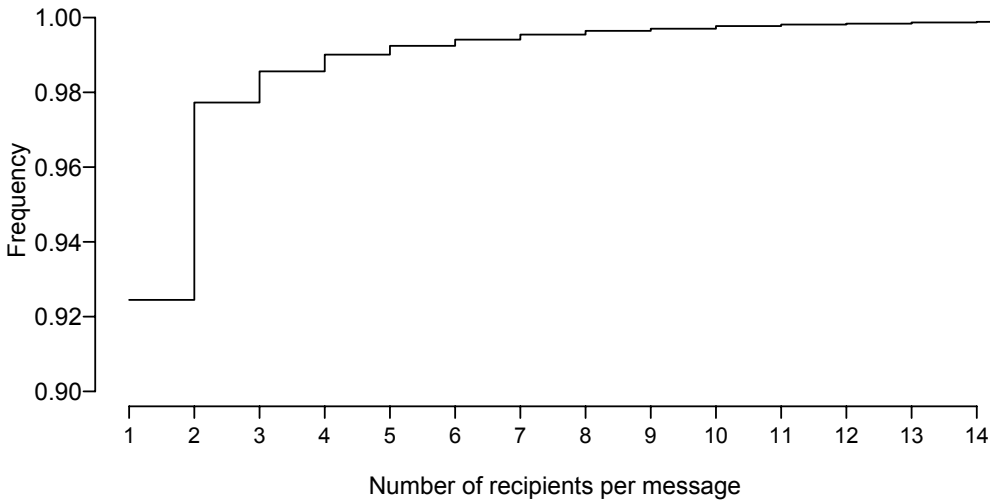

Figure 9. The number of recipients of a message.

with $\theta = 0.860$ ($r^2 = 0.957$). The tail represents one-time senders as well as many forged addresses from which spam is sent.

Figure 9 shows the number of recipients per message; most messages are delivered to a single recipient. This is unsurprising as group communication messages using mailing lists are replicated individually to each user.

These results are consistent with those presented in previous work.

4.4. Time-varying nature

Figure 10 shows the amount of mail transfer over the tracing period. Since mail traffic was observed at an educational institution, most messages are sent during the weekdays, with the weekends experiencing a precipitous drop in traffic (shown with the original curve). The dip in the trace during the months of December 2002 and January 2003 represents the decreased volume of mail processed during the winter holidays.

Figure 11 shows a typical day of mail transfer. We observe that most mail transfer occurs during the workday, especially during the lunch period and late in the evening.

Estimating a stationary distribution of interarrival times requires care. As our dataset exhibits strong non-stationary trends—daily and weekly variations as depicted in Figure 10—we need to estimate an interarrival distribution based on a segment in which message arrivals are well modeled as stationary. We chose an 8 hour segment on 1 December 2002 for this purpose. Figure 12 shows that message interarrival times, in seconds, closely follow an exponential distribution with parameter $\lambda = 0.132$ ($r^2 = 0.977$). The mean message interarrival time is about 7.6 seconds. Interarrival times also show

DOI: 10.1002/spe

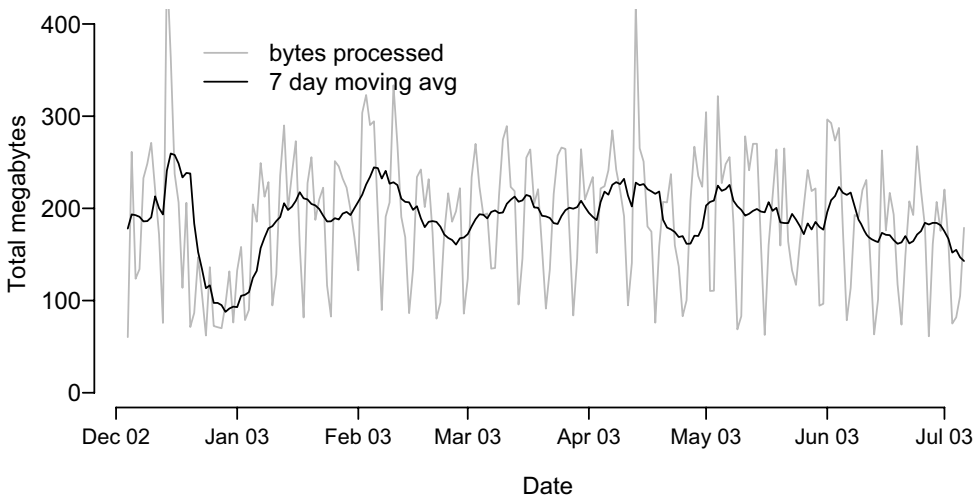

Figure 10. The amount of mail received, per day, during the tracing period.

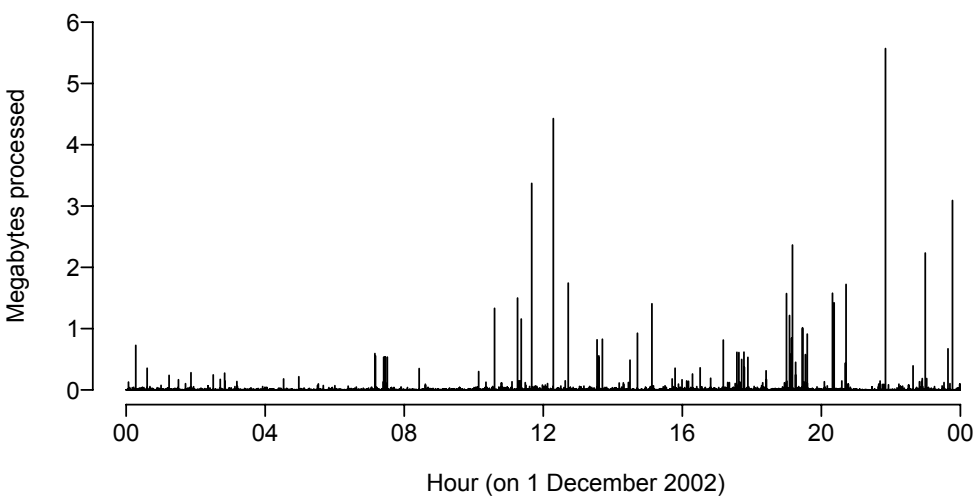

Figure 11. Mail transfer over a typical day.

DOI: 10.1002/spe

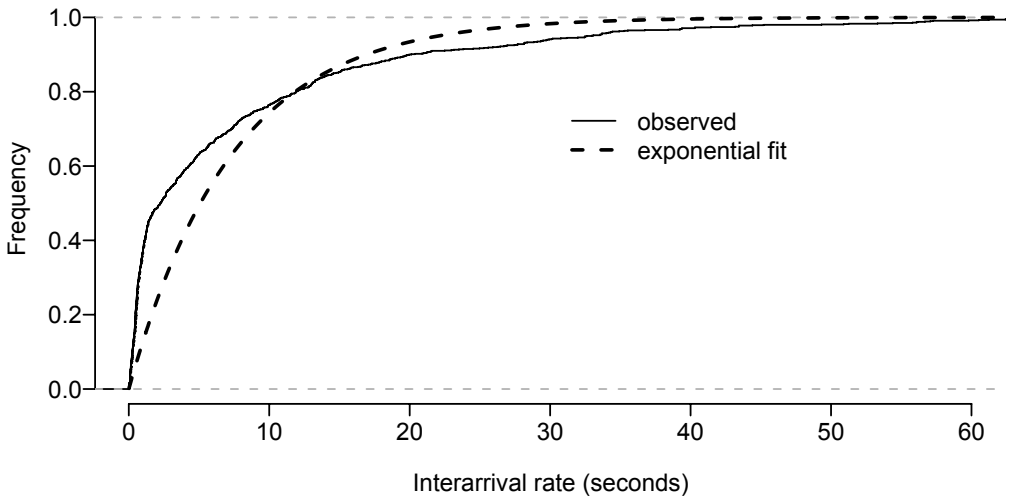

Figure 12. Message interarrival times over an 8 hour segment on 1 December 2002.

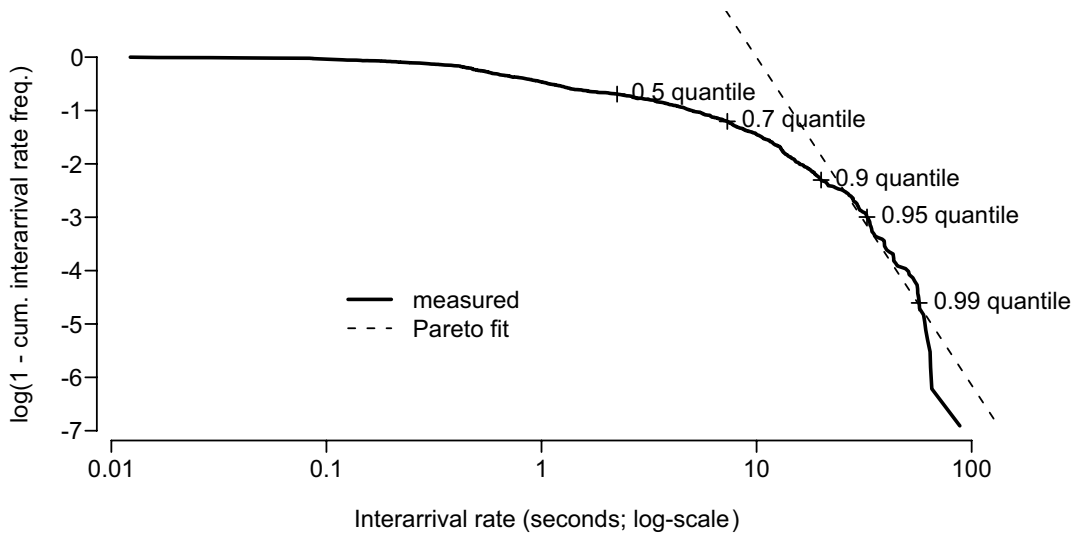

Figure 13. Message interarrival time LLCD over an 8 hour segment on 1 December 2002.

DOI: 10.1002/spe

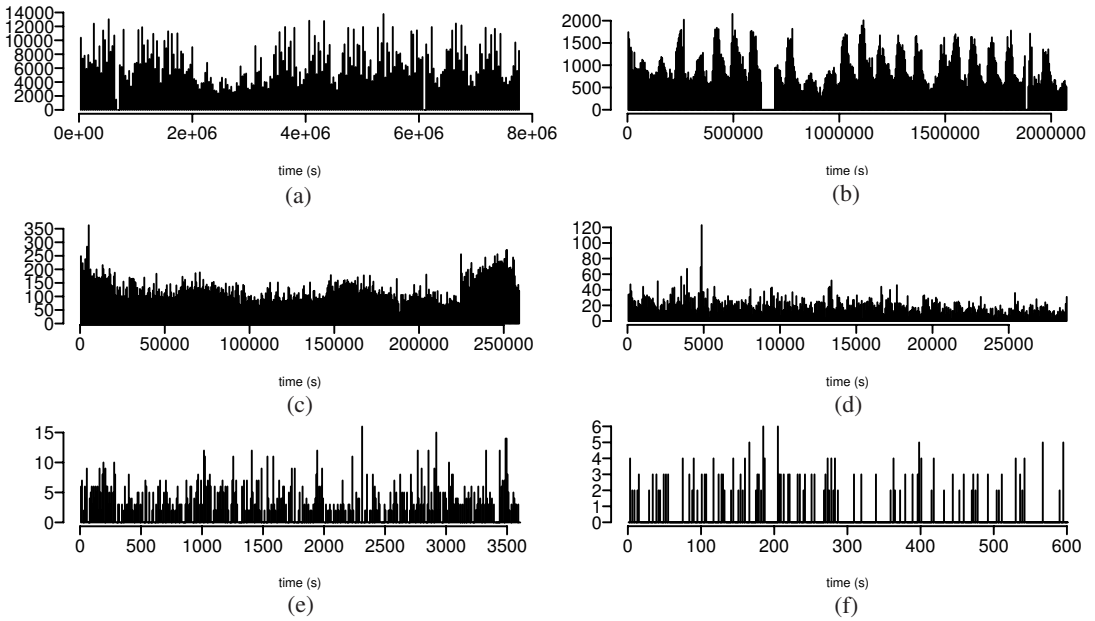

Figure 14. The message processing rate, in messages received per second, shown at different time scales. The x-axis represents time, in seconds, and the y-axis shows the number of messages processed during the time scale: (a) 32 768 second buckets (3 months); (b) 4096 second buckets (24 days); (c) 512 second buckets (3 days); (d) 64 second buckets (8 hours); (e) 8 second buckets (1 hour); (f) 1 second buckets (10 mins).

evidence of a Pareto power-law tail after 20 seconds (0.9 quartile) as shown in Figure 13. The Pareto tail can be estimated with $\alpha = 3.66$ ($r^2 = 0.915$).

Figure 14 shows the number of messages received across timescales of six different orders of magnitude. The x-axis represents time, in seconds, and the y-axis shows the number of messages processed during the timescale. Starting in the lower-right with 1 second, each subsequent plot's time resolution increases by a factor of eight. The most obvious feature of these plots is that they appear bursty at all timescales. Furthermore, there is no characteristic size of a burst: at every timescale there are highly bursty periods separated by smaller bursty periods. This bursty nature manifests itself in transient congestion at mail servers.

A notion of burstiness is characterized by self-similarity [15–17]: any section of the data has the same statistical properties as any other with such a timeseries exhibiting bursts, extended periods greater than average, at a wide range of timescales. While Figure 14 provides pictorial evidence of the self-similar nature of message processing, a robust quantitative measure is required. The degree of self-similarity is defined via the *Hurst exponent*, H. A data source is self-similar if $0.5 < H \leq 1$, with increasing H indicating a greater extent of burstiness.

Graphical methods of determining the Hurst exponent—detrended fluctuation analysis (DFA) [18], a periodogram [15,17,19], and a variance–time plot [15–17,19]—are shown in Figure 15 for a subset of

DOI: 10.1002/spe

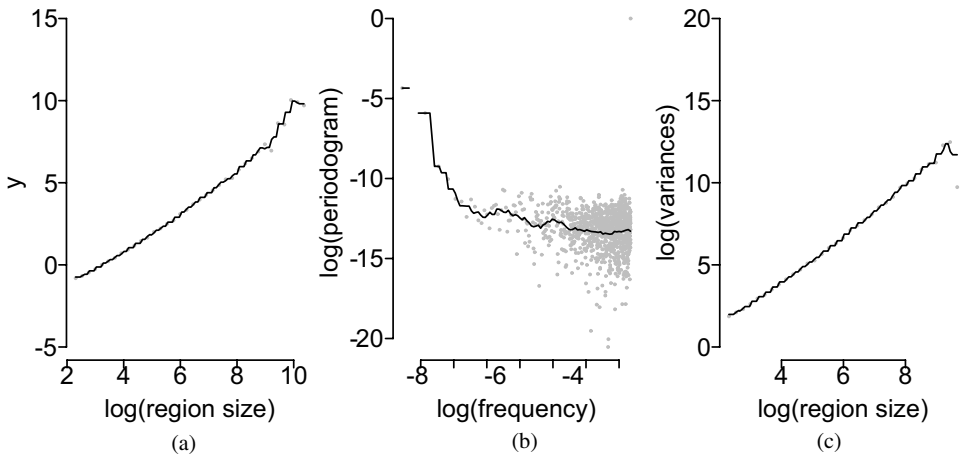

Figure 15. (a) DFA ($H = 0.664$), (b) periodogram ($H = 0.672$) and (c) variance–time ($H = 0.724$) plots of the message processing rate.

8 hours of the trace. These techniques produce estimated Hurst exponents of 0.644, 0.672 and 0.724, respectively. We thus estimate the Hurst exponent of the message processing rate to be $H \approx 0.68$, indicating that e-mail traffic is indeed self-similar. These results indicate that a dynamically optimizing storage system for e-mail could exploit the quiet periods, rearranging data to provide increased throughput during busy periods.

The evidence of a Pareto heavy tail in the interarrival time distribution interarrival times is consistent with Bertolotti and Calzarossa [4,5], but in contrast to Gomes *et al.* [6], who found exponential arrival times with no existence of heavy tails. A strictly exponential arrival time distribution implies Poisson process modeling of SMTP connection arrivals is adequate. Self-similarity, however, relies crucially on the heavy-tailed property of interarrival times [17]. To correctly model e-mail traffic, one can superimpose heavy-tailed ON/OFF processes, where ON times correspond to the delivery of a message by a client and the OFF times correspond to periods when that client is idle.

4.5. Interdepartmental mail

Messages that are sent and received within our institution, considered interdepartmental mail, account for about 42% of all messages (see Figure 16). Interdepartmental mail accounts for 24.7 GB or about 48.9% of all bytes transmitted (see Figure 17). There does not appear to be a correlation between interdepartmental mail and any associated content type.

4.6. Content type

Each message can be composed of several MIME components or parts. Most messages, 77%, consist of only the textual message body, 20% of messages contain only one attachment and 2% contain

DOI: 10.1002/spe

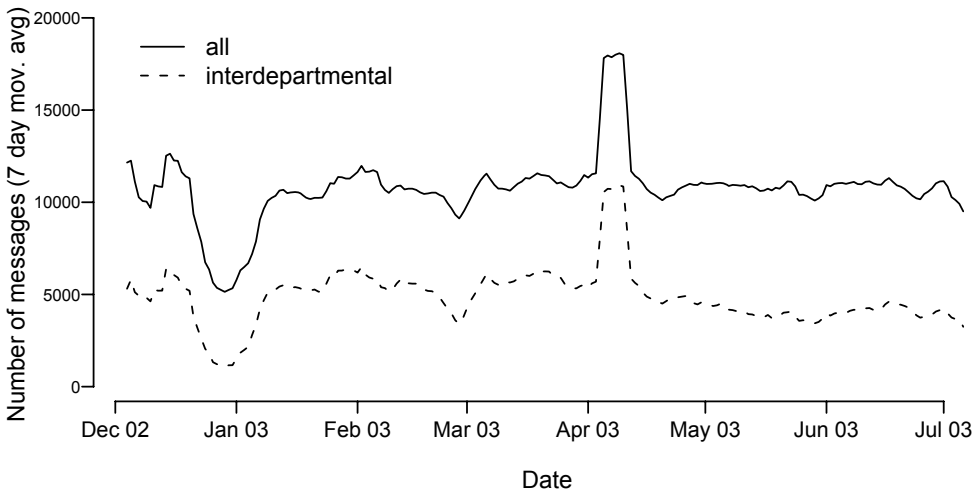

Figure 16. Interdepartmental mail transfer by number of messages transmitted (7 day moving average).

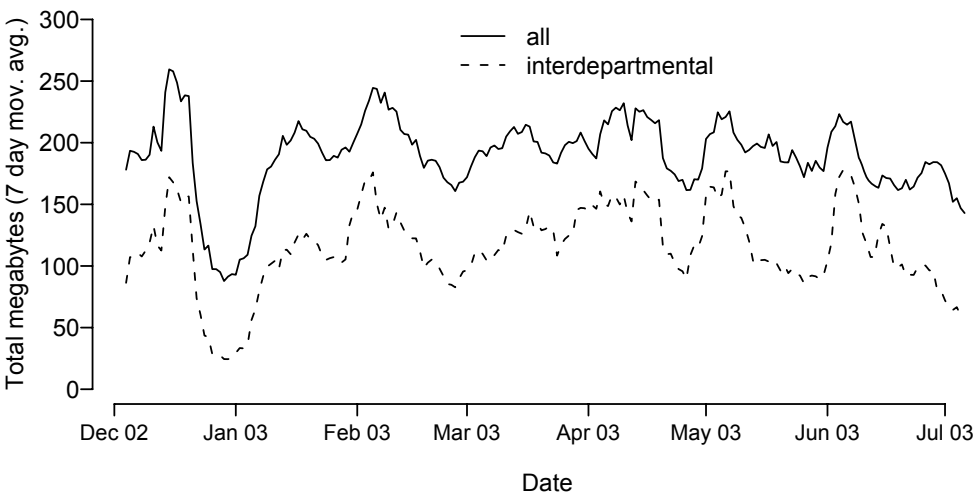

Figure 17. Interdepartmental mail transfer by number of bytes transmitted (7 day moving average).

DOI: 10.1002/spe

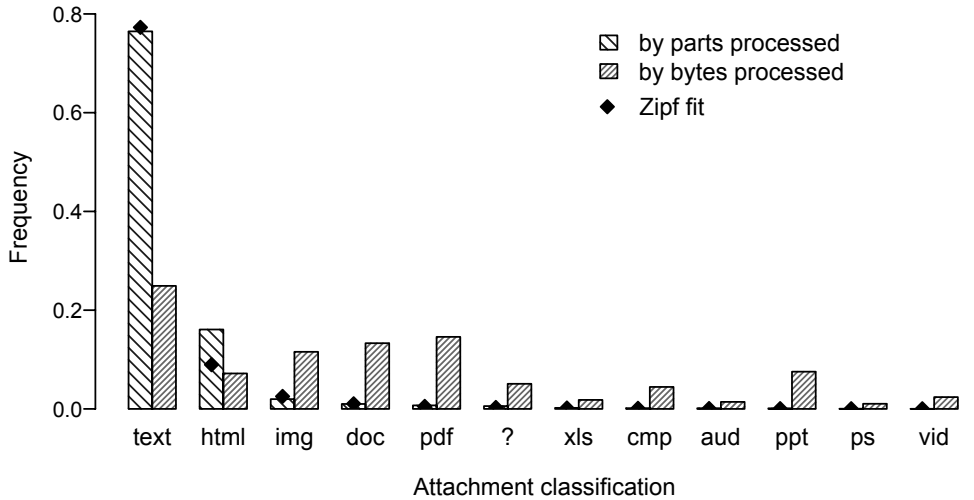

Figure 18. Distribution of message parts and bytes across different content types. `vid` refers to video files (ASF, MPEG etc.); `cmp` is compressed files (gzip, zip etc.).

two attachments. Three or more attachments account for less than 1% of all messages. This is unsurprising as we expect most messages to be correspondence.

Figure 18 shows that while attachments account for a small proportion of the total number of MIME parts seen, they do account for a significant fraction of delivered bytes. Classifiable but obscure file types—those that accounted for less than 0.5% of attachments in terms of frequency count and delivered bytes—were omitted.

The diamond markers in Figure 18 represent a Zipf distribution fitted onto the content frequency with $\theta = 3.11$ ($r^2 = 0.974$). The fit is close with the exception that HTML content is under represented. We believe the reason for this under representation is that some mailing software automatically attaches an HTML version of the plain text message to each outgoing message. Thus, the extra HTML content is not representative of actual user content creation, but an artifact of a software 'feature'.

4.7. Duplicate message content

Of the 2.85 million messages collected, about 2 million of them have at least one message that is exactly identical. This, largely due to multiple recipients of a message and announcement mailing lists that replicate messages to every user, accounts for a total of 3.2 GB, or 6.3% of the total bytes seen.

Using content-based indexing (Section 3.2), we counted 30.9 GB of unique data, or 61.1% of the total bytes seen. The remainder, 19.6 GB, are redundant. Content-based indexing is able to capture attachment forwards. Figure 19 shows the fraction of byte uniqueness by content type.

Audio and video content generally contain many redundant bytes as these are likely to be shared. Images are generally created by the end user and are unique. Word, Excel and Powerpoint documents

DOI: 10.1002/spe

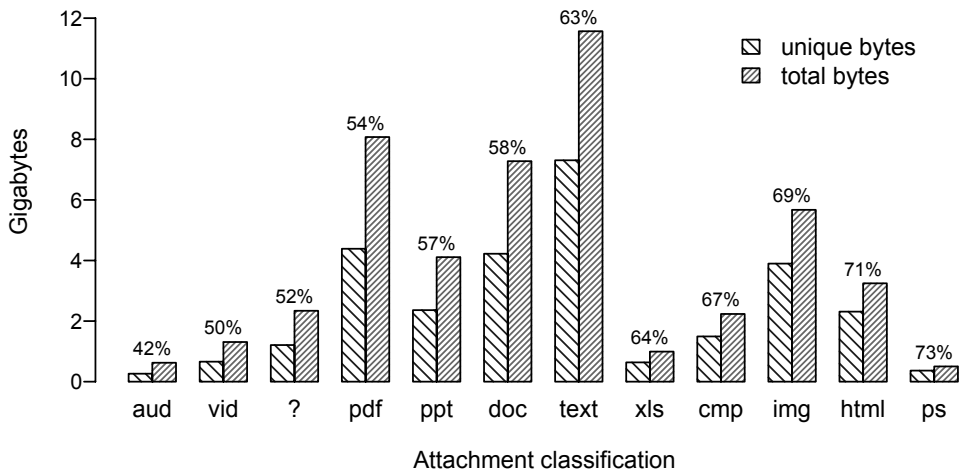

Figure 19. Fraction of unique bytes by content type.

represent a similar amount of redundancy. However, it is unclear why Postscript and Acrobat would be at the opposite ends of the spectrum and why HTML content is fairly unique.

Surprisingly, there are a sizable portion of attachments, labeled by `?` in the figure, that are unclassifiable by their MIME type or file name. We generally believe these consist of executables exchanged between users and mis-tagged by a user's mailer.

4.8. E-mail as a sharing mechanism

Earlier we hypothesized that e-mail is being used as a file-transfer mechanism. Despite the availability of distributed file systems, we believe users often turn to other mechanisms to coordinate data sharing. This is especially true for small sites and for *ad hoc* collaboration between a few users, where the administrative burden of setting up a distributed file system is often significant. In this case, we believe that sending files through e-mail is seen as the easiest way to exchange data. An interesting question is whether e-mail is indeed being used as an out-of-band file transfer mechanism.

Any time one user sends another a file, they form a bilateral sharing relationship. Figure 20 shows the fraction of a user's received messages that contain non-textual, non-HTML, non-SPAM attachments. The average user at our institution has about 14% ($\sigma = 0.09$) of their mail containing shared files, indicating that there is a non-trivial amount of out-of-band data sharing.

4.9. Content similarity

Some of these shared data are redundant as identical content is sent to multiple recipients and because, we hypothesize, small edits are performed in a group collaboration setting. In the latter case, a file evolves by users performing small edits on a document and propagating it to the group.

DOI: 10.1002/spe

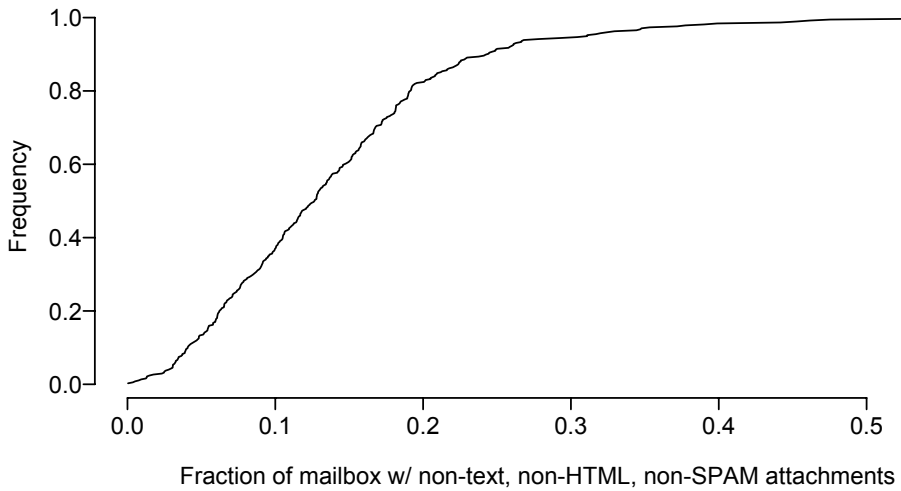

Figure 20. CDF of the fraction of a user's received messages that contain non-textual, non-HTML attachments.

Ideally, to determine whether this hypothesis is true, we could have classified content similarity by comparing messages within a thread using string matching algorithms. However, privacy requirements prevent us from keeping actual message payloads. Alternatively, we use content-based indexing to delineate a message's payloads into chunks where we can match up repeated payloads but cannot reconstruct them.

Matching up similar payloads is fairly straightforward. For two payloads, with multisets of chunk identifiers $\mathcal{A}$ and $\mathcal{B}$, respectively, the intersection of the sets is the payloads' common data and the difference of $\mathcal{A}$ from $\mathcal{B}$ is the new data added. We define the *payload delta* as the number of bytes that are new expressed as a fraction of the new payload size. Note that the differences between two payloads is at the level of granularity of a chunk, which is of variable size.

To see whether small edits on a document do occur in a group collaboration setting over e-mail, we compute the payload delta between successive temporal messages within a thread—versions—that contain Microsoft Word, Excel or PowerPoint attachments. The fraction of changed bytes is computed on only the attachment alone, not the entire message containing the attachment. Determining message threads in a non-privacy intrusive way is difficult: we chose to group messages with the same recipients and with at least one chunk in common as a thread. The latter condition biases results in favor of similarity, but its omission creates an unworkable set of disassociated messages.

Figure 21 shows a CDF of the fractions of bytes changed per version. Although these data prevent us from making any conclusive statements about collaboration, we find that roughly 30% of versions have some similarity between them. Intuitively, this leads us to believe that there is more than a non-trivial amount of 'ping-pong'-style collaboration via e-mail.

DOI: 10.1002/spe

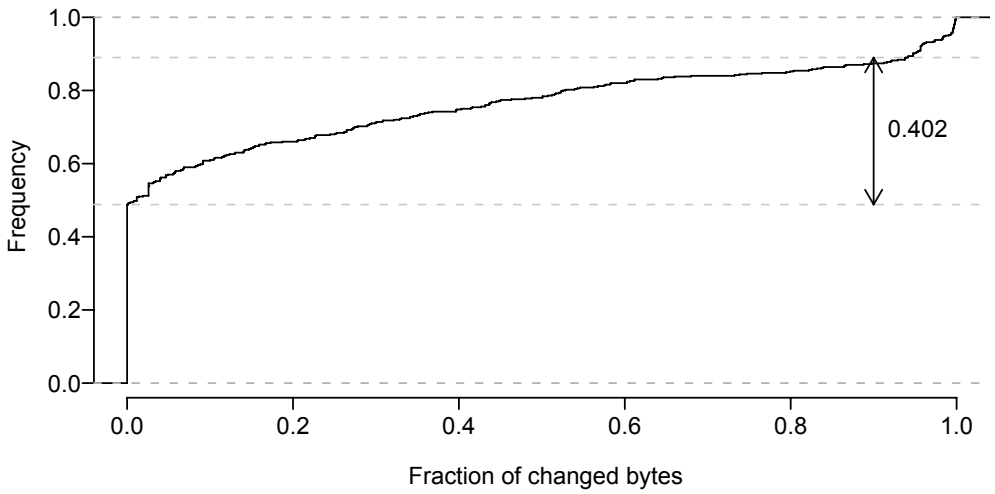

Figure 21. Fraction of bytes changed over versions of Microsoft Word, Excel and PowerPoint attachments.

4.10. Content temporal locality

Temporal locality, generally referring to the notion that an object recently seen has an increased probability of being seen among its peers, consists of two components: popularity—how frequent an object occurs compared to its peers—and correlation—where a causal relationship may exist between two or more objects [20]. Correlation is meaningless when dealing with message content as it is irrelevant whether identical messages follow each other in the reference stream: they are simply messages destined to more than one recipient. We focus on popularity.

In Web system's analyses [20], the reference stream is shown to be concentrated among a small fraction of all objects referenced; this popularity skew is usually captured through fitting a Zipf-like distribution. Chunks provide a mechanism for uniquely identifying content within e-mail that is traditionally unaddressable. However, capturing popularity skew by fitting the chunk reference stream to a Zipf-like distribution yields little value: messages that are replicated to many users will merely have their chunks deemed more 'popular'. For example, an announcement message accidentally replicated twice to every user will be at the apex of this curve. Instead, our notion of popularity should capture the longevity, if any, of content.

To that end, we define the lifetime of a chunk as the time from which it is first seen to the time it is last seen. Figure 22 shows that the majority of chunks, approximately 87%, are unique and are never seen again. Within the first 200 seconds, 92% of chunks are seen and are never to be seen again. This is likely because mailing list messages and spam sent as separate connections to different users are received during this time frame. These results conclude that there is very little content longevity.

When delineating mail based on its departmental or non-departmental status, there is no difference in their lifetime distributions. Furthermore, while we expect that most long-lived chunks are spam, there

DOI: 10.1002/spe

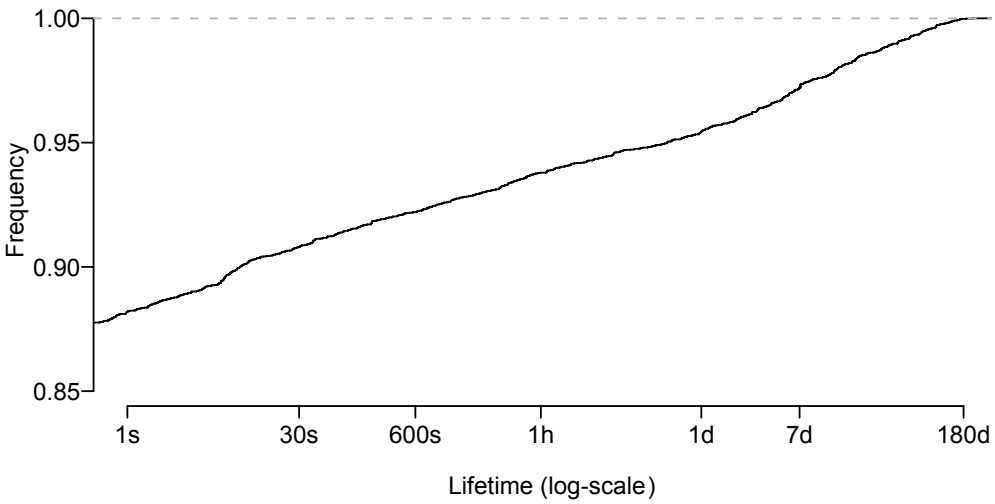

Figure 22. Chunk lifetime CDF.

does not appear to be any correlation between the lifetime of a chunk and whether it is spam. To avert filters, spam messages are continually subtly changed, defeating our content-detection mechanism.

Our intuition had suggested that there would be some content longevity; that users would forward attachments for some reasonable duration. However, the data show that redundant data, and particularly long-living redundant data, are less prevalent than we initially hypothesized.

4.11. Summary

Message size. Message sizes closely followed a censored model with a log-normally distributed body and a heavy Pareto-like tail, with evidence of infinite variance. The median message size is 1901 bytes with it being 3814 bytes for spam. While most messages are small, most bytes come from large messages.

Receivers and senders. Few users receive most of the messages, mimicking a Zipf distribution. Messages also arrive from addresses according to a Zipf distribution. Most messages have a single recipient.

Time-varying nature. Mail traffic is very bursty as indicated by the high degree of self-similarity. The inter-arrival times of messages follow an exponential distribution with a Pareto heavy tail.

Interdepartmental mail. About 42% of messages and 49% of all messages bytes are destined for recipients within the department.

Content type. Plain text accounts for most message payloads, but images and document types account for the most bytes received. Attachment content types are Zipf distributed.

DOI: 10.1002/spe

Duplicate message content. 70% of all messages have at least one other message that is exactly identical to it. Using content-based indexing, we found that 26.5% of all message bytes received are redundant.

E-mail as a sharing mechanism. There is evidence that e-mail is used as a file sharing mechanism. The median user at our institution has 13% of their incoming messages contain a file attachment.

Content similarity. There is some evidence that 'ping-pong'-style group collaboration occurs over e-mail.

Content temporal locality. There is very little longevity of content within an e-mail system, indicating minimal temporal locality.

5. IMPROVING POSTMARK

The Postmark benchmark [22] is designed to measure file system performance in small-file Internet server applications such as electronic mail. It creates a large set of continually changing files and measures the transaction rates for a workload of many small reads, appends and deletes.

Postmark initially creates a pool of files and then performs transactions on them. Each transaction consists of two subtransactions.

1. Creating a new file within the pool or deleting an existing file in the pool.
2. Sequentially reading an entire file or appending some data to a file. Here, Postmark assumes an mbox-style mail spool, where all messages are stored sequentially in a single file, delimited by a special token.

Each subtransaction operates on a file uniformly selected from the pool and each operation within a subtransaction is equally likely to occur. That is, the probabilities of creating a new file, deleting an existing file, reading a file or appending to a file are all equal to 0.25. These operations attempt to provide a first-order approximation of mail server traffic.

The initial pool size and the number of transactions is chosen to be very large to minimize the effects of caching and transient effects. Postmark is an as-fast-as-possible benchmark; there are no idle periods. The measure of Postmark is the number of transactions per second.

Some of Postmark's characteristics counter conclusions drawn in this paper. Postmark does *not* synchronously write to disk. This is necessary in a mail environment as messages must be permanently stored. All of the comparisons to Postmark are made to a version of Postmark that performs synchronous updates by calling `fsync` after each create or append transaction.

Postmark uses a uniform distribution from 500 to 10 000 bytes to draw the amount of data to append. Each file does not exceed the high watermark of 10 000 bytes. However, message sizes follow a log-normal distribution with a heavy tail (see Section 4.1). In addition, an artificial limit on the file size produces additional incorrect modeling. For example, a large append will cause additional capacity misses within the disk cache that would not be modeled with exclusively small appends.

Furthermore, Postmark uses a uniform distribution to pick candidate files for appending, assuming that each user is equally likely to receive mail to their mailbox. However, few users receive the most mail, following a Zipf distribution (see Section 4.3).

A modified version of Postmark, called *Postmark-m*, for mail, incorporates the above changes.

DOI: 10.1002/spe

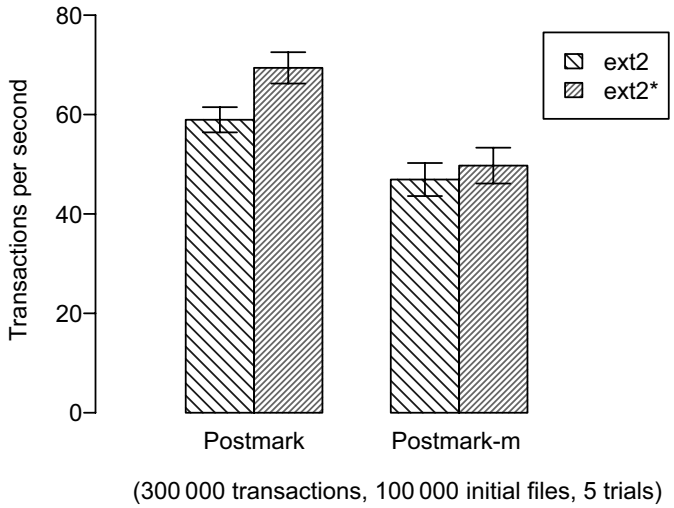

Figure 23. Comparison of Postmark and Postmark-m under the standard ext2 file system and ext2*, a modified version of ext2 that preallocates files.

Failing to correctly model the heavy-tailed nature of appends in Postmark causes resource misallocation under real e-mail workloads. We ran Postmark and Postmark-m on standard ext2 [21] and a modified version of ext2, referred to as *ext2**, under Linux 2.4. The ext2* version is equivalent to ext2, except that during file creation the file system attempts to preallocate 20 disk blocks (10 240 bytes) contiguously for the file—the high watermark file size of Postmark.

We ran five trials of each version of Postmark against ext2 and ext2* for 300 000 transactions with 100 000 files. The results are shown in Figure 23. In raw performance, Postmark-m is about 20% slower than Postmark under ext2.

Worse, the artificial nature of Postmark's high watermark causes it to perform about 18% better under ext2* as all files are contiguous on disk. A two-sample T-test between the file systems yields p-values of 0.0005 and 0.24 for Postmark and Postmark-m, respectively. This indicates that, for any reasonable confidence level, Postmark statistically diverges under ext2* while Postmark-m does not. The conclusion that ext2* is a better file system candidate for e-mail servers is erroneous.

The heavy-tailed characteristics and Zipf candidate file selection of appends cause files to grow and also become fragmented, causing additional seeks and rotational latency delays in reading a file. Also, since seeks are a dominating performance bottleneck, a large append causes additional capacity cache misses, flushing many small updates out of the cache. This results in re-seeks to retrieve those updates.

Figure 24 shows read transaction time CDFs for Postmark and Postmark-m under ext2 and ext2*. The median read transaction time for Postmark is 12.9 ms under ext2 and only 0.87 ms under ext2*. The ext2* version is able to effectively pigeon-hole each file into contiguously allocated space. For Postmark-m, the median read transaction times are 10.2 ms and 7.8 ms for ext2 and ext2*, respectively. We believe Postmark-m more accurately reflects actual mail server operation.

DOI: 10.1002/spe

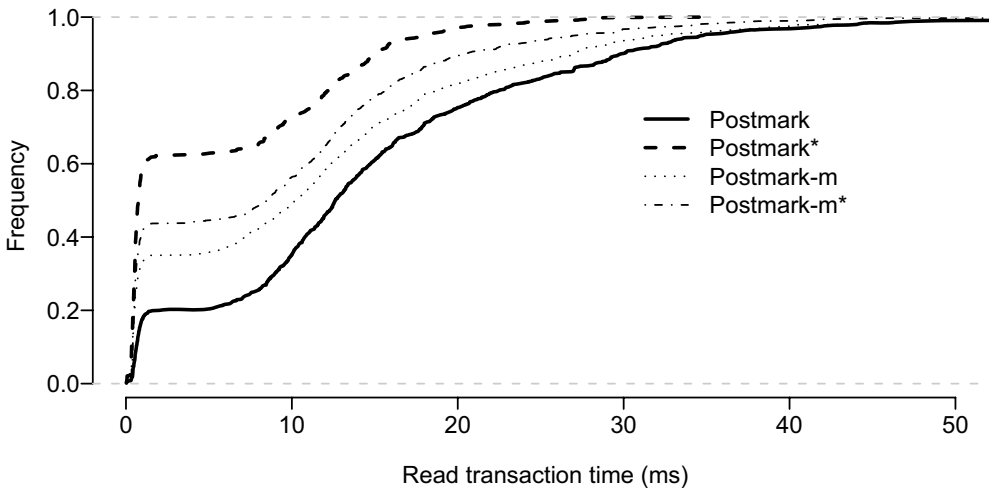

Figure 24. Read transaction time CDFs for Postmark and Postmark-m, under ext2 and ext2*.

6. CONCLUSION

The main contribution of this paper is a large-scale study of e-mail. We collected data over a seven month period, instrumenting about 2.85 million messages. We have supplied analytical distributions and possible explanations for several e-mail parameters of interest, furnishing a data point for e-mail patterns. We have found analytical distributions for message sizes, senders and receivers and burstiness. We have confirmed that message sizes have a heavy tail and offer evidence of extreme variance. Heavy tails were also found in message interarrival times along with a high degree of burstiness or self-similarity. Furthermore, we have analysed attachments, finding their content type, redundancy, version similarity and temporal locality.

These results allow better modeling of e-mail workloads in benchmarking and systems engineering. We have shown that Postmark's assumptions about file system behavior are incorrect and have provided a new version of Postmark, Postmark-m, that more accurately describes the nature of mail server access patterns.

REFERENCES

1. Fallows D. Email at work: Few feel overwhelmed and most are pleased with the way email helps them do their jobs. *Technical Report*, Pew Internet & American Life Project, December 2002.
2. Saito Y, Bershad BN, Levy HM. Manageability, availability and performance in Porcupine: A highly scalable, cluster-based mail service. *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP'99)*, Kiawah Island, SC, December 1999. ACM Press: New York, 1999; 1–15.
3. Standard Performance Evaluation Corporation. SPECmail2001 mail server benchmark architecture white paper, January 2001.

DOI: 10.1002/spe

4. Bertolotti L, Calzarossa MC. Workload characterization of mail servers. *Proceedings of the 2000 Symposium on Performance Evaluation of Computer and Telecommunication Systems*, Vancouver, BC, 16–20 July 2000. SCS Press, 2000.
5. Bertolotti L, Calzarossa MC. Models of mail server workloads. *Performance Evaluation* 2001; **46**(2–3):65–76.
6. Gomes LH, Cazita C, Almeida JM, Almeida V, Meira W Jr. Characterizing a spam traffic. *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement (IMC '04)*, Taormina, Italy 2004. ACM Press: New York, 2004; 356–369.
7. Postel J. Simple mail transfer protocol. *Request for Comments 821*, August 1982.
8. Crispin M. Internet Message Access Protocol—version 4rev1. *Request for Comments 2060*, December 1996.
9. Myers J, Rose M. Post office protocol—version 3. *Request for Comments 1939*, May 1996.
10. Borenstein N, Freed N. MIME (Multipurpose Internet Mail Extensions) part one: Mechanisms for specifying and describing the format of Internet message bodies. *Request for Comments 1521*, September 1993.
11. Manber U. Finding similar files in a large file system. *Proceedings of the USENIX Winter 1994 Technical Conference*, San Fransisco, CA, 17–21 January 1994. USENIX Association: Berkeley, CA, 1994; 1–10.
12. Rabin MO. Fingerprinting by random polynomials. *Technical Report TR-15-81*, Center for Research in Computing Technology, Harvard University, 1981.
13. Nocedal J, Wright SJ. *Numerical Optimization*. Springer: New York, 1999.
14. Hernández-Campos F, Marron JS, Smith FD, Samorodnitsky G. Variable heavy tailed durations in internet traffic, part I: Understanding heavy tails. *Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02)*. IEEE Computer Society Press: Los Alamitos, CA, 2002; 43.
15. Creorovella M, Bestavros A. Self-similarity in world wide web traffic: Evidence and possible causes. *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Philadelphia, PA, May 1996. ACM Press: New York, 1996; 160–169.
16. Gribble SD, Manku GS, Roselli D, Brewer EA, Gibson TJ, Miller EL. Self-similarity in file systems. *Proceedings of the 1998 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*. ACM Press: New York, 1998; 141–150.
17. Leland WE, Willinger W, Taqqu MS, Wilson DV. On the self-similar nature of ethernet traffic. *ACM SIGCOMM Computer Communication Review* 1995; **25**(1):202–213.
18. Peng CK, Buldyrev SV, Havlin S, Simons M, Stanley HE, Goldberger AL. Mosaic organization of DNA nucleotides. *Physical Review E* 1994; **49**:1685–1689.
19. Taqqu MS, Teverovsky V. On estimating the intensity of long-range dependence in finite and infinite variance time series. *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, Adler RJ, Feldman RE, Taqqu MS (eds.). Birkhäuser: Boston, MA, 1998; 177–217.
20. Fonseca R, Almeida V, Crovella M, Abrahao B. On the intrinsic locality properties of web reference streams. *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, San Francisco, CA, 30 March–3 April 2003. IEEE Computer Society Press: Los Alamitos, CA, 2003.
21. Card R, Ts'o T, Tweedie S. Design and implementation of the second extended filesystem. *Proceedings of the 1st Dutch International Symposium on Linux*, 1994.
22. Katcher J. Postmark: A new filesystem benchmark. *Technical Report 3022*, Network Appliance, October 1997.

DOI: 10.1002/spe