Distributed Proxies for Browsing Privacy — A Simulation of Flocks

MARTIN S OLIVIER
University of Pretoria

In previous work we introduced an anonymising proxy scheme — called Flocks — to be used for browsing privacy. Flocks is similar to Crowds in that each proxy randomly decides whether to forward any request it receives to the destination Web server, or whether to forward the request to another proxy. In this manner request chains are formed that hide the details of an originator of a request from the destination server as well as from the various proxies.

Flocks differs from Crowds (and similar Privacy-enhancing Technologies or PETs) because it caches pages that are requested. Unlike related PETs, Flocks is intended for deployment within an organisation. Caching minimises the need for communication between an organisation and external content providers, decreasing cost and potentially increasing access speed. Logging in Flocks is designed to balance privacy with the need to conduct forensic investigations when required (with safeguards to prevent unauthorised breaches of privacy).

Two parameters determine the behaviour of Flocks: α is the probability with which any proxy will forward a request to an external server (rather than to another proxy) and N is the number of proxies in the system. In previous work we analytically determined the impact of these two parameters on privacy and performance aspects of Flocks.

The current paper reports on simulations that were performed to gain deeper insight into the behaviour of Flocks. The simulations confirm the analytic results of our previous work. They also shed light on performance-related issues such as the number and positions of access to external servers, saturation levels and traffic patterns. This information will be useful to decide on appropriate values of α and N from a performance point of view.

The simulations also highlight the problem of overly long chains that will occasionally occur. A simple solution is proposed and tested empirically. The privacy and performance implications of this solution are discussed; it is found that the solution is usable, but has a profound impact on the choice of the α and N parameters.

Categories and Subject Descriptors: H.3.5 [**Online Information Services**]: Data sharing—*Security*; H.2.7 [**Database Administration**]: Security, integrity, and protection; K.4.1 [**Public Policy Issues**]: Privacy

General Terms: Management, Reliability, Security

Additional Key Words and Phrases: Personal privacy, privacy architecture, privacy-enhancing technologies

1. INTRODUCTION

Every evening after a long day's work, John goes to his local pub and buys everybody who is there a beer. After the single beer he goes home to enjoy the rest of the evening with his family. Everybody who knows John, knows him as a friendly, generous person. However, as time goes buy, a computer system also begins to 'know' John: His credit card company computer begins to 'see' that John is spending a substantial amount on liquor every evening — perhaps the equivalent of ten or twenty beers. Before long the computer profiles John as an alcoholic. . .

This simple scenario illustrates one of the problems when computers record humans' actions: because such information is often collected out of context, a different picture may emerge of an individual than the true picture. Truth, however, is not the only criterion to be used when considering the implications of recorded information. Individuals also have the right to control some information about them that should rightfully be considered private — even if it happens to be true. The division of life into public and private spheres dates back to Aristotle [DeCew 2002]; it posits the existence of a domain that is not part of the public. While this dichotomy is overly simplistic, it demonstrates the long-held view that some information is properly private.

Parent "defines privacy as the condition of not having undocumented personal information known or possessed by others" [DeCew 2002]. The merits of this view will not be discussed in the current paper; suffice to say for the moment that the view has received some support.

Against this backdrop, consider the fact that logging implies the 'documentation' of one's actions. A case can be made that one's actions cannot be considered private if logged at all. However, even if one does not accept Parent's view, information generated in an employer-employee relationship is often seen as the property of the employer — especially if the computers used are the property of the employer. And even if the data is not considered as the property of the

Author Address:
M.S. Olivier, Information and Computer Security Architectures (ICSA) Research Group, Department of Computer Science, University of Pretoria, Pretoria, 0002, South Africa; http://mo.co.za.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, that the copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than SAICSIT or the ACM must be honoured. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.
© 2005 SAICSIT

employer, system administrators often have unrestricted access to such log data — and can move the information to the public domain.

One may argue that in this environment only an employee's private Web usage needs to be protected (if private use is tolerated at all). However, even official use of Web resources can reveal much about an individual: Work times and patterns are revealed; the manner in which more complex environments, such as search engines, are used can, in principle, be used as 'psychometric tests', from which personality traits of employees' may be inferred that they do not even know about themselves. Moreover, as indicated at the start of this paper, context is often not recorded: an employee's specific task on a given day may require him or her to use Web resources that may be frowned upon — and will be frowned upon when these logs are looked at years later, when knowledge about the context has long been forgotten.

Flocks is a PET proposed earlier by us [Olivier 2004] that is to be used in such environments. It is based on anonymous proxies that operate similar to those used by Crowds [Reiter and Rubin 1999]: A proxy forwards any request it receives with probability α to the external server; it forwards the request with a probability $1 - \alpha$ to a random (other) proxy. Flocks differs from Crowds because is caches requests at these proxies: if a requested page is already available at a given proxy, the proxy simply retrieves it from the cache (if it has not expired), rather than forward the request to an external server or another proxy. Since Flocks proxies are intended to be deployed in a single organisation, the number of external requests are minimised; this should yield lower communication costs and higher response speeds than Crowds. A second significant difference between Flocks and Crowds is the fact that the former is instrumented for (warranted) forensics — something that can be justified in the given application area [Olivier 2005].

In previous work [Olivier 2004] we identified the two major parameters that influence the behaviour of Flocks: α that has been described above, and N, the number of proxies in the system. In this earlier work we determined the impact of specific choices for these two parameters analytically.

The current paper describes simulations that were performed to gain deeper insight into the operation of Flocks. Simulations can better deal with the complexities introduced by caching than statistical analysis can. Simulations are also able to highlight extreme operating characteristics — and can be used to answer 'what if' questions by subtly changing the behaviour of the system.

The remainder of the paper is structured as follows. Section 2 contains background information. Section 3 describes the simulation programs used. Section 4 compares the observed simulated results with the results expected from our earlier statistical analysis [Olivier 2004]. Next, Section 5 discusses the patterns that emerged for fetching the data from external servers. The observations in the latter two sections impact on the privacy characteristics and the 'cost' of the system. One of the observed undesirable characteristics was (as expected [Olivier 2004]) long trail (or chain) lengths in some cases. While these cases were rare, they are still disconcerting. Section 6 addresses this issue with a minimum impact on the privacy characteristics of the system; however, it is necessary to reconsider the choice of α and N if this solution is used. Section 7 concludes the paper.

2. BACKGROUND

This section describes the background against which the research described in this paper has been conducted. Since this background has not changed since our previous work on Flocks, this section is heavily based on the background provided for an earlier paper on Flocks [Olivier 2005]; in fact, this section is an (almost) verbatim copy of Section 2 of the earlier paper.

Over the years a number of PETs have been introduced. Many of the current ideas were already present in Chaum's 1981 [Chaum 1981] notion of a mix. In the mid 1990s attention turned to specific technologies used on the Internet (and, more specifically), the Web. This focus was based on the realisation that interacting on the Internet often leaves a trail that may be used to learn more about an individual than should be tolerated. The PETs developed in the 1990s were mostly intended to allow the individual to exert control over what information is made known to other parties, by using appropriate intermediaries. These intermediaries could be (fixed) third parties, such as Anonymizer [Caloyannides 2000], Janus (or Rewebber) [Rieke and Demuth 2001] or LPWA [Gabber et al. 1999]. The third parties could also be (randomly or deterministically) selected from a set of available proxies or routers. Such ideas were used in Crowds [Reiter and Rubin 1999] and Onion-routing [Goldschlag et al. 1999].

In more recent times attention has turned to PETs that can be employed inside an organisation to help the organisation protect the information it has collected about individuals. Examples of developments in this regard include Hippocratic databases [Agrawal et al. 2002] and E-P3P [Karjoth et al. 2003; Ashley et al. 2003]. It has been argued that the costs (to the organisation) associated with deploying such a PET will be fully recovered by customer satisfaction — and even lead to increased business opportunities [PrivacyRight 2001; Systems 2000; IBM 2002]. In the case of Flocks a similar case could be made about the benefits employee satisfaction hold for the organisation.

Flocks [Olivier 2004] was introduced as a PET based on technologies such as Crowds, but one intended for deployment within an organisation. In this environment it was intended to minimise external traffic by caching Web pages retrieved from the Internet as far as possible, but yet minimised an administrator's (or even a manager's) ability to breach the privacy of users by just browsing logs (or more actively profiling users). However, a fundamental tenet behind Flocks was the fact that the PET does not reduce users' accountability — and that forensic investigations should be possible where a legitimate reason exists for such an investigation. In essence, Flocks operates as follows. Each user operates a personal proxy and acts as trustee for the logs generated by that proxy. When a user submits a request, it is submitted to this proxy. The proxy serves the request from cache, if possible. Else it forwards the request to the external destination, with probability α. If it does not forward the request to the destination, it forwards it to another proxy that is chosen randomly. In the latter two cases, the result is cached (if possible), once it arrives. Requests from other proxies are dealt with similarly.

Many overviews of PETs have been published [OECD 2002; Goldberg et al. 1997; Froomkin 1996; Seničar et al. 2003]. For a structured view of PETs, see the Layered Privacy Architecture (LaPA) [Olivier 2003].

3. SIMULATING FLOCKS

A number of simulation programs were written for this study. The principle behind all of them is that they simulate requests for a single page over time.

At this point we are assuming that all pages behave in a similar manner: when it is requested, it will eventually be retrieved from the external server or a cached copy will be found. This copy will be returned along the trail of proxies used, and will be available from any of these proxies during subsequent requests. The facts that (a) pages in caches become stale after some time period, and (b) that this time period differs for different pages — even to the extent that some pages cannot be cached — were ignored during the simulations (and were ignored during our earlier analysis [Olivier 2004]). The implications of these restrictions will be considered in later work. We contend that the simulation is accurate for cases where public information is retrieved from the Web. Cases such as Internet banking require each page to be retrieved in any case, and will behave like other pages where only one copy needs to be retrieved. These cases will typically use a single trail of average length, and are not considered further in this paper.

The simulation programs therefore start with an array with a Boolean entry for each proxy in the system. Initially all these entries are set to false, since the page under consideration is not cached at any proxy. A round of requests then start where the page is successively requested from random proxies. For each request, the request is satisfied if the page is cached at that proxy. Else, with probability α, it is assumed that the page is requested from the external server. If this does not happen, the page is requested from another random proxy, where this process is repeated recursively. Once a page is found (whether from an external server or from another proxy) the cache flags for all the proxies on the trail are set to true to indicate that the page is now available from that proxy's cache. After the number of requests per round have been issued, the caches are marked as clear and a next round is begun.

During the process various parameters, such as the number of times the page needs to be fetched from an external server and the saturation (ie the proportion of proxies that have the page cached), are calculated.

To illustrate the process, consider the following output generated from one of the rounds from one of the simulation programs. This was the first round of the simulation data used to determine mean and median trail lengths and the fetch patterns to external servers. For this round the parameters were $\alpha = 0.05$ and $N = 256$.

```
#1,+21,2,1,3,+14,+7,+2,3,6,4,3,5,3,1,2,1,4,+5,8,2,2,6,2,1,8,1,2,3,2,+2,2,2,3,...
```

The `#1` merely identifies the line as round 1. The `21` that follows, indicates that the first request of this round visited 21 proxies (ie it had a trail length of 21). The `21` is preceded by a `+` to indicate that the request resulted in a fetch from an external server. The `21` is followed by a `2`: this means that the second request of the first round visited two proxies. The absence of a `+` indicates that the request was answered from a cached copy.

It was (arbitrarily) decided that each round will consist of 80 requests. This number of requests were indeed enough to observe the trends as more and more requests were sent; in fact the behaviour of the system stabilised long before the 80th request.

The number of rounds also had to be decided on. Table I lists some of the (incremental) key observations after every 100 rounds (for $N = 128$, $\alpha = 0.05$). Again it is clear that most of the values stabilise early in the process. The one significant exception to this is the trail length — in theory very long trail lengths are possible, and it literally is necessary to run the simulation for a 1000 rounds to see the once in a thousand long trail lengths. Obviously, more rounds would have probably led to observation of even longer trail lengths. However, for most purposes, 1000 rounds were more than enough, and this number of rounds were therefore used.

4. EXPECTED RESULTS

The first purpose of the simulations was to confirm the results in our earlier paper [Olivier 2004] where statistics were determined analytically.

Two of the measures determined in the earlier work were the mean and median length of trails. The mean length was determined to be $\nu = \frac{1}{\alpha}$, while the median length was calculated as $\mu = -\log_{1-\alpha} 2$. An optimum level of privacy is achieved when typical trails are neither too long, nor too short: If the trail of an uncommon request is too long, the fact that

Number of	Trail length			Saturation	External fetches	
iterations	Mean	Median	Maximum	Mean	Mean	Median
100	19.78	13	104	74.6%	4.7	5
200	19.24	13	145	74.2%	5.0	5
300	18.87	14	145	74.3%	5.1	5
400	19.79	14	145	74.5%	5.0	5
500	20.07	15	145	74.5%	4.9	5
600	19.69	15	145	74.5%	4.9	5
700	19.49	14	145	74.5%	5.0	5
800	18.89	14	145	74.4%	5.0	5
900	18.92	13	145	74.4%	5.0	5
1000	19.36	14	156	74.4%	5.0	5

Table I. Some incremental simulation observations after every 100 rounds for $N = 128$, $\alpha = 0.05$. Expected trail length mean was 20; expected trail length median was 13.5.

	Mean				Median			
	Expected	$\mathbf{N = 64}$	$\mathbf{N = 128}$	$\mathbf{N = 256}$	Expected	$\mathbf{N = 64}$	$\mathbf{N = 128}$	$\mathbf{N = 256}$
$\alpha = 0.01$	100	98.24	98.30	98.70	68.7	71.5	69	67
$\alpha = 0.05$	20	19.42	19.36	19.75	13.5	15	14	13
$\alpha = 0.10$	10	9.59	10.18	9.73	6.6	7	7	7

Table II. Expected and observed trail length means and medians

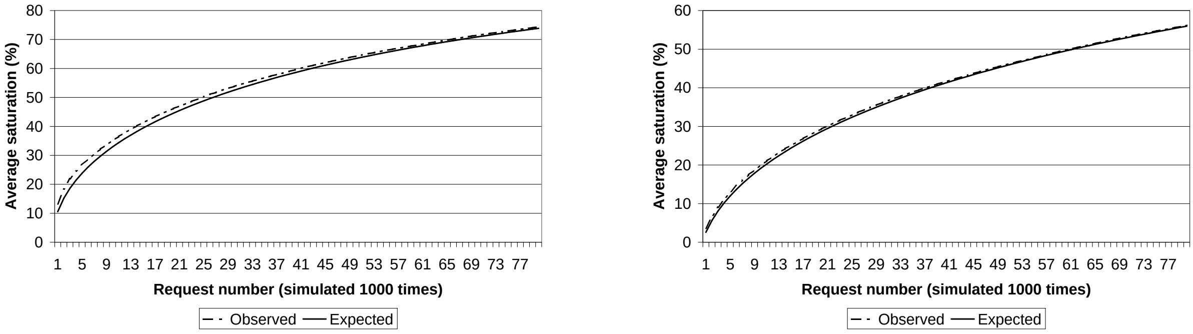

Figure 1. Expected and observed saturation for $\alpha = 0.05$ and $N = 128$ (left) and $\alpha = 0.1$ and $N = 256$ (right)

it has been issued will be obvious from too many proxy logs; if the trail is too short it may be too easy to follow from its point of detection to its origin, thereby violating the anonymity of the requester.

In the simulation the length of the trail of the first request of every round (when the cache is still empty) reflects the typical trail length. Later requests will often ‘link’ to other trails by fetching information that they have deposited in a cache. The effective length of such a ‘joined’ trail will then be the actual length measured, plus the length from the cache to the endpoint of that trail. Table II lists the expected and observed (simulated) means and medians of the first requests for the cases simulated. It is clear that the observed measures are indeed close to what was expected.

In our earlier work we also derived a formula to calculate the expected saturation after a given number of requests have been processed. The *saturation* refers to the proportion of proxies that have the requested page cached. The saturation after request n has been processed was denoted as σ_n. According to our analysis

$$\sigma_n = \sigma_{n-1} + \frac{(\alpha + \overline{\alpha}\sigma_{n-1})\overline{\sigma}_{n-1}}{(1 - \overline{\sigma}_{n-1}\overline{\alpha})^2 N}$$

Figure 1 plots the expected and observed saturation for two cases ($\alpha = 0.05$, $N = 128$ and $\alpha = 0.1$, $N = 256$). The derived formula assumed that $N \gg \mu$. The graphs for all the simulated cases where this applies show similar good correlation as the two depicted in this paper do.[1]

The calculated formula was specifically determined somewhat conservatively: μ was used as an estimate for σ_1 rather

[1] This restriction does not, in general, hold for the $\alpha = 0.01$ cases simulated: for $\alpha = 0.01$, the median is $\mu = 68.97$; simulations were run for $N \in \{64, 128, 256\}$.

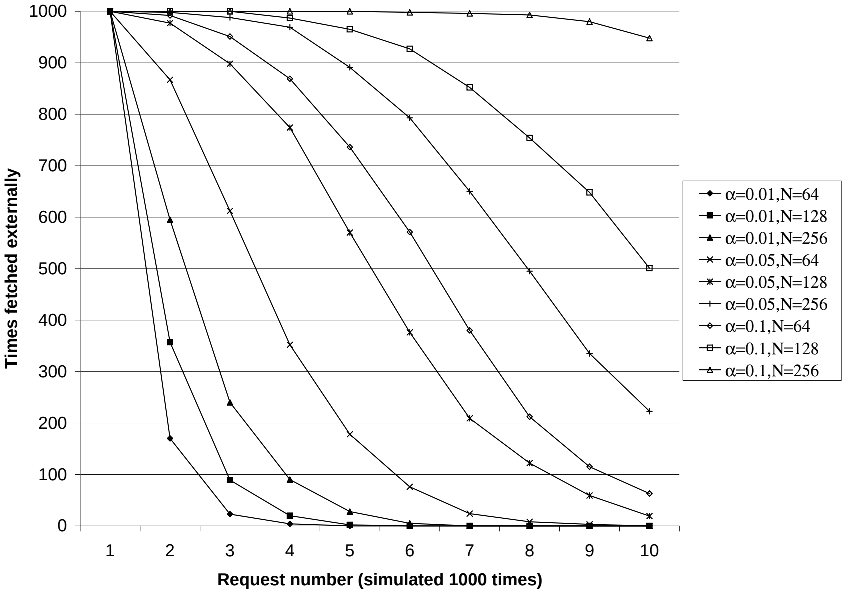

Figure 2. Number of external fetches required per 1000 rounds

than ν to minimise the influence of (rare but significant) very long trails. This is reflected in Figure 1 for $\alpha = 0.05$ and $N = 128$ where the observed saturation is initially marginally higher than the (conservatively) projected saturation. Where $\alpha = 0.1$ and $N = 256$ the significant difference between average and median trail lengths on the one side, and N on the other side, leads to an almost perfect match between the expected and observed saturation levels.

Figure 1 supports the correctness of the earlier calculation of σ_n and the current simulation of it.

5. EXTERNAL FETCH PATTERNS

The intention of the previous section was primarily to validate the previous and current work by comparing results about the same issues determined in vastly different manners. The current section turns the attention to facets of Flocks that can be learnt from the simulation that are hard to derive analytically. More specifically, this section considers external fetch patterns.

Figure 2 shows the points at which pages where fetched from external servers for each of the simulation runs. The figure clearly demonstrates the (inverse) S-curve behaviour of the system: Initially, most requests have to be fetched from the external server. Then, once the saturation level begins to increase, the number of requests that need to be sent to an external server drops rapidly, until all further requests are served from cache. Note the relatively small number of times a page needs to be requested: for many of the simulations run, 10 or fewer external requests were sufficient to get the system to the point where all future requests would be served from cache, or at least to the point where the S-curve's steep downward slope has already begun. Again, this correlates with our findings in the earlier paper: The saturation very quickly increases to the point where almost all subsequent requests are served from cache.

External fetches are not primarily privacy-related: they affect the cost of using the system. The cost for external fetches are significantly more expensive than retrieving a page from cache. In the original work on Flocks the assumption was made that internal requests have no cost (at least, when compared with the cost of requests that are sent to an external server). In a typical organisation that does not use a PET such as Flocks, a single external fetch will be sufficient for information pages irrespective of the number of times clients request them (until the cached copy becomes stale); Figure 2 re-emphasises that — depending on α and N — the communication costs of using Flocks will be multiple times that of the organisation that does not use Flocks. On the other hand, if the organisation relies on a PET such as Crowds, if n requests are issued, all n will result in external fetches. Hence, Flocks has a communication cost significantly lower than that of Crowds.

The next question to address is: At what points are requests sent to external servers? Table III gives part of the answer. It contains the median points at which requests are sent to external servers. Note that the values increase rapidly — up to the point where no further requests are sent to the external server since all requests can be served from cache.

Table III may give the false impression that external requests tend to again occur closer to one another towards the right of the table. This impression is due to the fact that the caches have, for many rounds, been filled during the later stages and no further external requests are necessary. Therefore the rounds that play a role in these later numbers are the ones where

α	N	Median request									
0.01	64	1	15	22	35	-	-	-	-	-	-
	128	1	18	35	52	51	-	-	-	-	-
	256	1	14	29	41	48.5	56	-	-	-	-
0.05	64	1	5	15	24	34	45	41.5	59	70	-
	128	1	3	10	22	30.5	40	46	53.5	57	56
	256	1	2	6	13	20	29	36	45	51	56
0.1	64	1	3	6	13	22	31	38	44	45	50
	128	1	2	5	9	13	20	26	33	39	44
	256	1	2	4	6	8	12	16	20	25	31

Table III. Median points at which the first 10 requests are sent to external servers

α	N	Earliest request										Latest request									
0.01	64	1	2	4	18	-	-	-	-	-	-	1	79	59	72	-	-	-	-	-	-
	128	1	2	3	12	50	-	-	-	-	-	1	78	80	74	52	-	-	-	-	-
	256	1	2	3	7	18	38	-	-	-	-	1	80	80	80	80	69	-	-	-	-
0.05	64	1	2	3	4	5	7	8	17	41	-	1	78	80	80	80	78	79	73	70	-
	128	1	2	3	4	5	7	10	16	17	33	1	72	78	80	80	80	78	80	80	80
	256	1	2	3	4	5	6	8	10	13	17	1	66	77	80	79	80	80	80	80	79
0.1	64	1	2	3	4	5	6	7	9	11	14	1	80	77	78	79	80	80	80	80	79
	128	1	2	3	4	5	6	7	8	10	13	1	78	80	74	77	80	80	80	80	80
	256	1	2	3	4	5	6	7	8	9	10	1	17	31	43	59	71	76	80	80	80

Table IV. Earliest and latest points at which requests are sent to the external server

the caches have not been filled earlier in the process and more requests need to be sent to outside servers later on.

To get a better understanding of the variance inherent in the process, consider Table IV. This table shows the earliest and latest points at which requests have been sent to the external server. Note the many occurrences where all the initial requests have been sent to the external server. In many cases it is possible that a page that is requested, say, five times, will indeed be requested five times from the external server — thereby increasing communication costs significantly over the median case, where such requests will in most cases only lead to one or two actual external fetches. If one considers the *Latest request* part of table IV it becomes clear that for most simulations, cases occurred where the initial requests essentially filled up all caches. This is illustrated by those cases where the second external fetch only occurred when the page was requested the 80th time. The situation where the first requests actually filled the cache is, in fact, not depicted in table IV. For $\alpha = 0.01$ and $N = 64$ out of the 1000 rounds, only 170 required a second external fetch — in 830 cases the first requests did fill all the caches in the system. Since $\alpha = 0.01$ does lead to very long trails, let us rather consider $\alpha = 0.05$. In these cases, 133, 33 and 2 rounds for $N = 64$, $N = 128$ and $N = 256$, respectively, filled all the caches with only one external request. For $\alpha = 0.1$ and $N = 64$, eight rounds filled all the caches from the first request. Neither of the two other values tested for N with $\alpha = 0.1$ could serve all subsequent requests from just the first request.

In some cases, serving all requests using just one external request is good news: Second, third, and subsequent requests establish trails that join one of the former trails. This gives the benefits of a trail hiding the user's identity from the target server and the other proxies in the manner Flocks was designed to work, yet costs very little in terms of external communication. However, in some cases this 'good' characteristic is actually caused by nonbeneficial behaviour: one of the early requests causes such a long trail that the caches in (almost) all the proxies are filled immediately.

The occurrence of first requests that fill all the caches is a reason for concern. Firstly, it is a concern from a performance point of view: A request typically has to be 'bounced' from proxy to proxy many times to fill *all* the caches in the system. While the assumption that internal communication costs are essentially free, such long trails will cause performance degradation; a slow response is unavoidable in such cases. The second concern raised goes to the heart of the purpose of Flocks: If a supposedly private request is indeed 'bounced' back and forth between proxies to the extent that it occurs in (almost) all the proxy caches after being requested once, the request will not only occur in *all* the proxy logs: to fill the caches it will in all likelihood have to visit several proxies more than once. This will cause the PET to make this request more visible to anyone who looks at the log, than it would have been, if no PET were used. While this will happen rarely, the mere possibility of this happening is serious. We consider this in Section 6 below.

Before leaving the topic of saturation, however, consider Table V. Figure 1 only illustrated two cases; Table V gives the saturations after 80 requests for the other cases. While such high saturations at early points are good for performance reasons (since queries can be answered from cache), they also indicate signifcant storage requirements at each of the proxies. This paper only notes this problem and leaves discussion of it for later research.

α	N		
	64	**128**	**256**
0.01	93.18%	81.20%	64.54%
0.05	88.84%	74.44%	58.53%
0.1	87.48%	72.92%	56.13%

Table V. Average saturation levels after 80 requests

α	N			**Median**
	64	**128**	**256**	
0.01	630	856	806	68.97
0.05	116	156	153	13.51
0.1	69	69	69	6.58

Table VI. Maximum observed trail lengths

		Without short-circuit							**With short-circuit**						
Parameters		**Trail length**			**Saturation**		**Ext Fetches**		**Trail length**			**Saturation**		**Ext Fetches**	
α	**N**	**Mean**	**Med**	**Max**	**Mean**	**Max**	**Mean**	**Med**	**Mean**	**Med**	**Max**	**Mean**	**Max**	**Mean**	**Med**
0.01	64	98.24	71.5	630	93.18	100	1.20	1	9.95	10	28	88.24	98.4	3.17	3
0.01	128	98.30	69	856	81.20	100	1.47	1	13.77	13	41	74.26	85.9	3.68	4
0.01	256	98.70	67	806	64.54	97.3	1.96	2	17.90	17	55	59.41	68.8	4.30	4
0.05	64	19.42	15	116	88.84	98.4	3.12	3	8.23	8	30	87.55	98.4	5.07	5
0.05	128	19.36	14	156	74.44	86.7	5.02	5	10.22	9	34	73.12	87.5	6.98	7
0.05	256	19.75	13	153	58.53	69.1	7.61	7	11.85	10	48	57.66	68.4	9.6	10
0.1	64	9.59	7	69	87.47	98.4	5.93	6	6.43	6	23	86.88	96.9	7.4	7
0.1	128	10.18	7	69	72.92	84.4	9.58	10	7.39	6	37	72.00	82.8	11.1	11
0.1	256	9.74	7	69	56.13	65.6	14.56	14	8.00	6	46	55.64	64.4	16.0	16

Table VII. The effect of short-circuiting

6. ON LONG TRAILS

The problem of long trails has been introduced in the previous section from a saturation perspective. However, the problem is already evident from the basic statistics calculated from the simulation data. Table VI shows the maximum observed trail lengths. It can be seen that these maximums were around ten (or even more) times the median trail length for the specific value of α.

The typical random behaviour of the system makes it hard to limit maximum trail lengths. One option is the following: When a proxy gets a request for a page that it does not have cached, it makes a note of the request. When it gets a second request for that page and still does not have the page cached, it obtains the page from the external server. We will refer to this process as *short-circuiting*. Table VII shows the effects of short-circuiting.

It is clear from Table VII that short-circuiting leads to a dramatic decrease in maximum trail lengths — in particular for those cases where the expected average trail length ($\nu = \frac{1}{\alpha}$) is relatively large when compared to N. For these cases the cost in terms of external fetches also increases dramatically (from 1 or 2 for the first 80 requests to 3 or 4). On the other end of the spectrum (typically where $\alpha = 0.1$) the effect on maximum trail length is not that dramatic, but the increase in cost is also less pronounced. The most interesting cases are the middle range ones (where $\alpha = 0.05$). In these cases the maximum trail lengths have indeed been reduced significantly (from 116, 156 and 153 to 30, 34 and 48, respectively), and cost (again in terms of external fetches) has increased by about 50% (from 3, 5 and 7 to 5, 7 and 10, respectively). While these observations have a direct bearing on costs, the mean and median trail lengths also hold implications for privacy. The observed median trail lengths for $\alpha = 0.05$ have changed from 15, 14 and 13 to 8, 9 and 10, respectively. The significance of these changes on privacy clearly depends on N: A change from 15 to 8 for $N = 64$ is clearly more significant than a change from 13 to 10 for $N = 256$: In the larger number of proxies a small change in median trail length will not have a significant impact on the 'visibility' of the request in the proxies. In the former case, a typical request will now only be logged in 8 out of 64 proxies, making it less visible than the original case were it would have been logged in 15 times in the 64 logs (with, perhaps, some duplicates at the same proxy); this decreases visibility of the request and increases privacy. On the other hand, there will now only be 8 intermediaries between the destination and the requester — compared to 15 in the original case. This makes it easier to trace the request and therefore decreases privacy. The optimum balance clearly depends on the specific situation and cannot be prescribed here. What is clear is that short-circuiting has a potential performance increase (in terms of reducing the maximum trail length) as well as a potential cost increase (by increasing the number of external fetches). In a similar fashion, as illustrated, it can benefit or detract from privacy. Its use should thus be carefully considered — it may indeed be beneficial in certain circumstances.

Note that we have not suggested that $\alpha = 0.05$ is ideal; the most significant effects were seen in these example cases for $\alpha = 0.05$. The effect depends on the relative magnitude of $1/\alpha$ compared to N; for other values of N a more significant impact may therefore be seen for other values of α.

Another option, as an alternative to short-circuiting, to consider is the following: Suppose each proxy attaches some "time to live" field to its request; whenever the request is forwarded to another proxy, this field is decremented. When a request arrives at a proxy and this field is zero, the proxy has to obtain an answer: if that proxy does not have the page cached, it has to send it to the external server to get answered with probability 1 rather than α. Clearly the initial time to live has to be chosen randomly, with some maximum. Care also has to be taken in this case: If a proxy receives a request with the time to live equal to the maximum, it knows that the request had to have originated from the predecessor proxy node. Hence a proxy should never (or hardly ever) use this maximum value. However, this enables a proxy to make a similar conclusion when a time to live of one less than the maximum is encountered. Recursively, this leads to problems for any chosen time to live. We therefore do not consider it further in the current paper. We have, however, not explored the use of some large maximum number as initial value (but only to use it rarely). The probability that when such a large number is used as time to live that that particular trail will turn out to be one that was going to be long in any case seems rather remote. This option should therefore be considered in future research.

7. CONCLUSION

This paper described simulations of the Flocks PET. The simulations were used to confirm earlier results that were determined analytically. The paper then proceeded and considered the external fetch patterns as observed during the simulations in greater detail.

The paper also considered the problem of overly long trails. A solution (referred to as short-circuiting) was tested and found useful — but one that has to be tested against the privacy requirements of a given situation.

Future work will investigate other solutions to the long trail problem — such as the solution alluded to at the end of Section 6. Future work will also attempt to better characterise the S-curve observed for external fetch patterns. In addition, future work will consider the impact of pages that become stale because they have been cached for too long and therefore needs to be refetched. The potential high storage requirements of Flocks proxies mentioned in Section 4 has also been left for future research. A prototype implementation is also planned to study the effects of some observations (such as the cost assumption for internal communication) in a more realistic environment.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Research Foundation under Grant number 2054024. Any opinion, findings and conclusions or recommendations expressed in this material are those of the author and therefore the NRF does not accept any liability thereto.

REFERENCES

AGRAWAL, R., KIERNAN, J., SRIKANT, R., AND XU, Y. 2002. Hippocratic databases. In *28th Int'l Conf. on Very Large Databases (VLDB), Hong Kong*.

ASHLEY, P., HADA, S., KARJOTH, G., AND SCHUNTER, M. 2003. E-P3P privacy policies and privacy authorization. In *Proceedings of the ACM workshop on Privacy in the Electronic Society*. ACM Press, 103–109.

BROWN, I. AND LAURIE, B. 2000. Security against compelled disclosure. In *Computer Security Applications 16th Annual Conference (ACSAC '00)*. IEEE, New Orleans, LA , USA, 2–10.

CALOYANNIDES, M. A. 2000. Encryption wars: Shifting tactics. *IEEE Spectrum 37,* 5, 46–51.

CHAUM, D. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM 24,* 2, 84–88.

DECEW, J. Summer 2002. Privacy. In *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed. http://plato.stanford.edu/archives/sum2002/entries/privacy/.

FROOMKIN, A. M. 1996. Flood control on the information ocean: Living with anonymity, digital cash, and distributed databases. *University of Pittsburgh Journal of Law and Commerce 395,* 15. http://www.law.miami.edu/~froomkin/articles/oceanno.htm.

GABBER, E., GIBBONS, P. B., KRISTOL, D. M., MATIAS, Y., AND MAYER, A. 1999. Consistent, yet anonymous, web access with LPWA. *Communications of the ACM 42,* 2 (February), 42–47.

GOLDBERG, I., WAGNER, D., AND BREWER, E. A. 1997. Privacy-enhancing technologies for the Internet. In *IEEE COMPCON '97*. IEEE, 103–109.

GOLDSCHLAG, D. M., REED, M. G., AND SYVERSON, P. F. 1999. Onion routing. *Communications of the ACM 42,* 2 (February), 39–41.

IBM. 2002. Privacy in a connected world. White paper, IBM. May. http://www-1.ibm.com/industries/government/doc/content/bin/private.pdf.

KARJOTH, G., SCHUNTER, M., AND WAIDNER, M. 2003. Platform for Enterprise Privacy Practices: Privacy-enabled management of customer data. In *Privacy Enhancing Technologies: Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers*, R. Dingledine and P. Syverson, Eds. Springer.

OECD. 2002. Inventory of privacy-enhancing technologies (PETs). Report DSTI/ICCP/REG(2001)1/FINAL, Working Party on Information Security and Privacy, Organisation for Economic Co-operation and Development.

OLIVIER, M. S. 2003. A layered architecture for privacy-enhancing technologies. *South African Computer Journal 31*, 53–61.

OLIVIER, M. S. 2004. Flocks: Distributed proxies for browsing privacy. In *Proceedings of SAICSIT 2004 — fulfilling the promise of ICT*, G. Marsden, P. Kotzé, and A. Adesina-Ojo, Eds. Stellenbosch, South Africa, 79–88.

OLIVIER, M. S. 2005. Forensics and privacy-enhancing technologies — logging and collecting evidence in Flocks. In *First Annual IFIP WG 11.9 International Conference on Digital Forensics*. National Center for Forensic Science, Orlando, Florida, USA. Accepted for presentation.

PRIVACYRIGHT. 2001. Control of personal information — the economic benefits of adopting an enterprise-wide permissions management platform. White Paper. `http://www.privacyright.com/info/economic.html`.

REITER, M. K. AND RUBIN, A. D. 1999. Anonymous web transactions with Crowds. *Communications of the ACM 42,* 2 (February), 32–48.

RIEKE, A. AND DEMUTH, T. 2001. JANUS: Server anonymity in the world-wide web. In *Conference Proceedings EICAR International Conference,* U. E. Gattiker, Ed. 195–208.

SENIČAR, V., JERMAN-BLAŽIČ, B., AND KLOBUČAR, T. 2003. Privacy-enhancing technologies — approaches and development. *Computer Standards & Interfaces 25,* 147–158.

SYSTEMS, W. 2000. User managed privacy: A new approach for addressing digital privacy and personal information on the Internet. White Paper. `http://www.wave.com/technology/PrivacyWhitePaper.pdf`.

M. S. Olivier, “Distributed proxies for browsing privacy — a simulation of Flocks,” in *Research for a changing world — Proceedings of SAICSIT 2005*, J. Bishop and D. G. Kourie (eds.), White River, South Africa, 104–112, September 2005.

©SAICSIT
Source: http://mo.co.za