Performing Replacement in Modem Pools

Yannis Smaragdakis

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332
yannis@cc.gatech.edu

Paul Wilson

Department of Computer Sciences
The University of Texas
Austin, Texas 78712
wilson@cs.utexas.edu

Abstract

We examine a policy for managing modem pools that disconnects users only if not enough modems are available for other users to connect. Managing the modem pool then becomes a replacement problem, similar to buffer cache management (e.g., in virtual memory systems). When a new connection request is received, the system needs to find a user to "replace". In this paper we examine such demand-disconnect schemes using extensive activity data from actual ISPs. We discuss various replacement policies and propose CIRG: a novel replacement algorithm that is well suited for modem pools. In general, the choice of algorithm is significant. A naive algorithm (e.g., one that randomly replaces any user who has been inactive for a while) incurs many tens of percent more "faults" (i.e., disconnections of users who are likely to want to be active again soon) than the LRU algorithm, which, in turn, incurs 10% more faults than CIRG. For good replacement algorithms, the impact can be significant in terms of resource requirements. We show that the same standards of service as a system that does not disconnect idle users can be achieved with up to 13% fewer modems.

1 Introduction

A pool of modems is a time-shared resource: there are typically many more potential users than can simultaneously connect. The most common instance of modem pools is in telephone-modem-based Internet Service Providers (ISPs), where modems accept user connections over the telephone network. When all modems are occupied, no more connections are allowed and the users attempting to connect receive a busy signal. Although ISPs strive to avoid busy signals, this is not always feasible. The most common line of defense is to keep a fixed ratio of subscribers to modems, with a value of 10:1 sometimes considered safe for avoiding busy signals. This policy, however, is hard to maintain consistently (e.g., as usage patterns change in response to marketing actions[1]) and cannot always protect fully against busy signals. As an added measure, some ISPs try to discourage subscribers from constant modem use by setting usage limits and applying surcharges for exceeding them. An additional widespread practice is to proactively disconnect users who have been idle for some fixed time interval or users who have been connected continuously for some period of time. Recently, Douglis and Killian [DK99] improved over fixed idle timeout policies with *adaptive* proactive disconnections of users. Their technique varies the idle time threshold for disconnecting a user, based on the user's past activity patterns.

Although all of the above techniques are valuable for certain scenarios, they do not acknowledge that, in the most common case, the cost of allowing a user to stay connected is a function of the current load. ISPs typically suffer a cost for prolonged usage *only* when the modem pool utilization has reached capacity.[2] In other words, it is commonly of no cost to ISPs to allow users to stay connected when modems are available. The cost of telephone lines is usually fixed for ISPs, regardless of usage. Also, the cost of local phone calls in the US is commonly fixed (or zero) for users, regardless of call duration. To the best of our knowledge, the only policy in use that somewhat relates usage limits and current load is the variable timeout policy: some ISPs have shorter timeouts for pre-defined "peak hours" (e.g., 6pm to midnight). Nevertheless, more sophisticated schemes are easy to implement and, as we argue, inconvenience users much less.

In this paper, we examine the case of treating a modem

[1] In one well-publicized case, a change in America Online usage pricing caused many users to stay connected longer, allegedly resulting in a barrage of busy signals that prompted a lawsuit by dissatisfied customers [Ano97].

[2] This is not to say that ISPs have no incentive for limiting usage even when capacity is not reached. For instance, ISPs could limit usage expecting that the market will support higher prices for increased usage. This is orthogonal to the concerns of this paper.

pool as a replacement buffer. That is, users are not disconnected until all modems are occupied. When a new user attempts to connect with all existing modems occupied, there are two possible outcomes: either there are no "idle" users currently and the new user will be denied service (busy signal) or one of the idle users will be *replaced* (i.e., disconnected in favor of the new user). Strictly speaking, this approach is not easy to implement exactly, because busy signals are generated by the telephone network, not the ISP. Nevertheless, as we will explain, the policy can be easily approximated closely in an actual modem pool.

The interesting question in replacement scenarios is how to choose the user to replace. We examine three different replacement algorithms: LRU, CIRG, and RANDOM. The LRU algorithm replaces the user who has been inactive the longest. The CIRG algorithm (for Conditional Inter-Reference Gap) is novel and is inspired by Phalke's work on inter-reference gaps for access prediction in virtual memory systems [Pha95]. In intuitive terms, CIRG attempts to recognize access patterns for individual users and should be a good fit for the modem pool domain. Finally, the RANDOM algorithm selects any idle user for replacement. In Section 3 we discuss in detail the replacement problem for modem pools. The problem is related but different from replacement in other settings (e.g., virtual memory systems). The differences in the problem and in the expected access patterns (e.g., no *spatial* locality) guide the design of replacement algorithms and we explain what the characteristics of a good replacement algorithm should be.

For our experiments we used three extensive traces of user activity, which cover several distinct circumstances. The results of our simulations show that our approach is much less inconvenient to users than proactive disconnections after a fixed period of inactivity. Similar to the Douglis and Killian work [DK99], our metric for "inconvenience" counts "faults", that is, disconnections of users who are likely to want to be active again soon. In these terms, disconnecting users only when the modem pool is full results in an order of magnitude fewer faults than a fixed idle timeout policy. More interestingly, however, we show that the choice of replacement algorithm is important. RANDOM has a much higher cost (by over 20% and up to 1000%) than LRU. LRU, in turn, performs worse than CIRG, often by 10% or more. (Both LRU and CIRG turn out to be very good predictors of future idle times.) This shows that the naive approach of disconnecting any idle user when the load is high is far from ideal.

It is an interesting challenge to further quantify the benefits of our approach (or of any other approach to disconnecting idle users). The difficulty is that disconnecting users may encourage them to set up their connection so that it appears to be permanently active. For this reason, although we present results for a wide range of values, we concentrate on a range that guarantees a quality of service similar to that of the trace collection environment. (The natural assumption is that when users are not annoyed, they will not change their behavior.) We show that, for CIRG, good quality of service can be maintained with up to 13% fewer modems than a no-disconnections policy. This result shows that when there are not enough modems (e.g., due to a surge in subscriber numbers) the impact can be softened significantly using our technique.

Our experiments are interesting, however, regardless of the actual economics and current practices in the ISP business. What we are really demonstrating is the kind of activity patterns that the combination of human users and modern Internet tools (e.g., browsers, email clients, etc.) is expected to exhibit. In particular, we analyze which techniques work well for predicting future idle times. These results may be applicable to more contexts than telephone-modem-based ISPs—practically in every case a time-shared resource is accessed interactively by Internet users and we want to predict future inactivity times.

2 Studied Workloads

We describe early on the workloads we studied, so that we can use concrete examples in the subsequent discussion.

The first of our traces is the one collected by Douglis and Killian and used in their study [DK99]. This trace consists of the activity over a week in May 1998, polled every 30 seconds, of 100 users of a modem pool maintained by AT&T Labs. The modems in the pool served as the uplink of an asymmetric connection (the downlink was a cable connection). The setup where this trace was collected is interesting because it uses static IP addresses. This makes disconnects less undesirable: all TCP connections can stay open until the next modem connection. Thus, the cost of disconnecting a user is primarily in terms of the delay of reconnection. As a consequence, the modem pool itself has an aggressive 15 minute idle timeout (which would probably be too short for general ISPs). A few users have no inactivity timeout (after their request) and can stay connected indefinitely. An interesting characteristic of this setup is that all users have a dedicated phone line. A direct consequence is that users cannot be denied service but also that they have less of an incentive to disconnect explicitly. In fact, the trace we studied contains no disconnection information. Douglis and Killian report that 15 users appeared active at all times. (Actually this number depends on what is

defined as "at all times": for a higher inactivity tolerance, 19 users appear active at all times.)

Our other two traces are from Telesys—the Internet Service Provider for the University of Texas community. In September 1998, Telesys was the largest ISP in the Austin, Texas metropolitan area (we are not aware of more recent data) with over 3,000 modems, serving over 34,000 subscribers [Aca98]. (The total number of subscribers seems to be artificially high: unusually many subscribers did not use their accounts at all during the two periods of our study. The inflation in the number of subscribers is probably due to the low cost of Telesys for its restricted user base—many Telesys users can afford to subscribe even though they rarely use the service.) Our Telesys traces contain explicit disconnect information, and users are allowed to stay connected indefinitely. (Telesys does not disconnect idle users despite a stated policy of a 2-hour idle timeout.) Telesys has not had busy signals due to limited capacity in the past few years [Aca98], hence no users were denied service during our trace collection.

We collected the two traces of Telesys activity by repeatedly polling the modem servers in two-minute intervals. The traces are from periods of significantly different activity. The first was collected in a period of low activity (June 26 to July 6, 1999, which includes the 4th of July holiday) with 18,086 distinct users accessing the system in this time period. The maximum number of simultaneously connected users was 2,151, and 5 users were connected at all times. In all, there were over 315,000 connections (i.e., instances of users connecting and disconnecting). The second trace was collected in a high activity period (November 1 to 8, 1999). 21,221 distinct users accessed the system in that time with a maximum of 3,024 users connected simultaneously. There were over 347,000 connections in total and 11 users stayed connected throughout the week-long tracing period. The number of connected users as a function of time for the Telesys traces is plotted in Figure 1.[3]

An interesting issue concerns users who stay permanently connected (termed "workaholics" in the Douglis and Killian study, with the term originating in Barbará and Imieliński [BI94]). If "workaholics" are connected regardless of disconnections (e.g., due to periodic tasks using the network) they should be included in the study. If they are active only because they try to "beat" the disconnection policy of the tracing environment, they should be excluded. This issue is significant only for our first trace (from AT&T Labs) because of the idle timeout that was in place when the trace was collected, and because of the large percentage of "workaholics". In the Douglis and Killian study, "workaholics" were excluded from the experiments. The reason was that after interviews with individual users, Douglis and Killian concluded that their constant activity was due to programs used as a response to the 15-minute idle timeout. (That is, these users ran periodic tasks to simulate activity and avoid automatic disconnections.) Similarly, we excluded "workaholics" from our first trace. Nevertheless, we have also performed all experiments with the full trace and concluded that the inclusion of "workaholics" does not fundamentally affect our results or our conclusions (absolute counts may change but relative differences are practically the same).

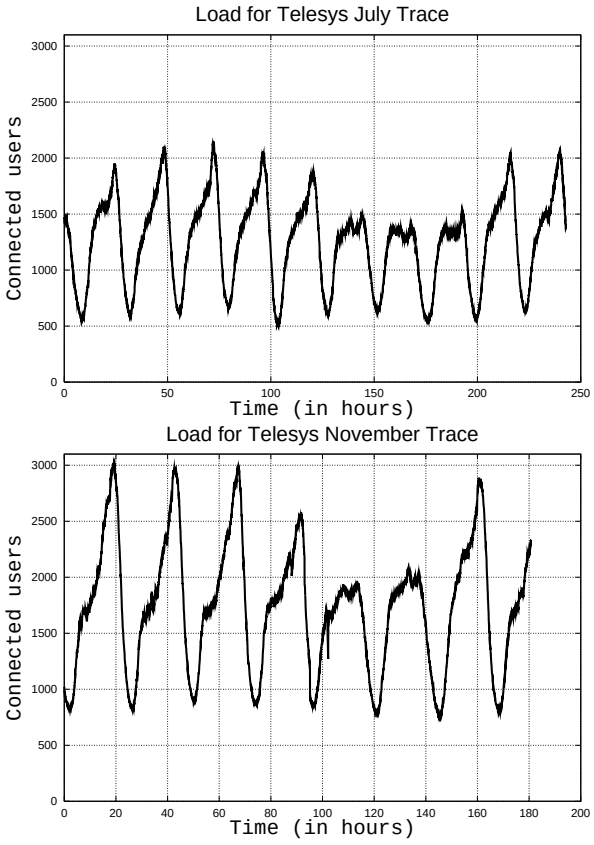

Figure 1: Number of connected users as a function of time. The first plot shows three "low usage" days because of the 4th of July holiday.

3 Replacement Algorithms for Modem Pools

The replacement problem in buffer management is a general and well-studied problem. In this section we

[3] No similar data can be plotted for the AT&T Labs trace since no disconnection information is available. The set of connected users for that trace depends on the disconnection policy used.

identify the special characteristics of the replacement problem in modem pools and appropriate replacement algorithms for the modem pool domain.

3.1 User Activity Patterns

The primary difference of the modem pool setting relative to a general replacement setting is the possibility of denial of service (that is, busy signals). Whereas replacement in general buffer management is based on always satisfying the incoming request, in modem pools it is better to sometimes deny requests. The reason is both subjective (because active users will get very annoyed at losing a resource that they claimed first) but also often objective: with dynamic IP address assignment, disconnections are costly as all existing TCP connections will be terminated. Overall, there is no compelling reason to always satisfy a request for connection in modem pools, unlike, for instance, buffer management in memory hierarchies. Hence, the modified replacement problem that we study, includes a "minimum inactivity threshold" parameter. A user can be replaced only if she has been idle for at least this much time. If no user can be replaced, incoming connection requests are denied.

The purpose of a replacement algorithm is to recognize regularities in the reference patterns and predict correctly which entities can be replaced with minimum cost. Thus, the unique characteristics of the modem pool domain influence the choice of a replacement algorithm. A characteristic of this domain is that there is no concept of *spatial locality*, or, in general, any way to associate the activity of two different users. "Spatial locality" refers to the observation that once an entity becomes active, other entities that are close to it in a well-defined space are likely to also become active soon. For instance, in virtual memory systems once a page is referenced, pages next to it, either in the address space, or in the recency space,[4] are also likely to be referenced soon. No such inference can be drawn in the case of modem pools. Individual users are quite likely to have no interactions with one another and there is rarely reason for their inactivity patterns to be correlated. The lack of spatial locality means that replacement algorithms that may be successful in other domains are unsuitable for modem pools. For example, recently proposed algorithms for virtual memory replacement, like SEQ [GC97] (which detects regularities in the address space) or EELRU [SKW99] (which detects regularities in the recency space) are not appropriate for modem pool replacement.

The question then becomes, what kind of regularities are there in modem activity? Clearly, activity patterns are mainly dictated by human users but also Internet tools (e.g., email clients, browsers, etc.). It is interesting to examine whether strong regularities exist. For this, we consider the distributions of idle times before a user becomes active for the three traces of our study, shown in Figure 2.

There are a few observations we can make. First, as expected, there is strong *temporal locality* exhibited by all traces: the number of times a connection is reactivated generally decreases very rapidly as a function of the idle time. A second observation is that there is evidence of programmatic (i.e., periodic) behavior in all traces. In the AT&T Labs trace, strong spikes of activity appear at 5 and 10 minutes of idle time. Additionally, strong activity is observed after 14 minutes of idle time, which is immediately before the inactivity timeout of 15 minutes for this service. Further activity, however, is evident for idle times (16 minutes) slightly *longer* than the disconnection timeout. This suggests that some users may have set their systems to reconnect immediately after a non-user-initiated disconnection. This is one of the features that may interact with our simulations and are discussed further in Section 4.1. The Telesys traces also exhibit periodic activity, for instance, every 10, 20, and 30 minutes. Activity after exactly 5 or 15 minutes of idle time could also be strong for the Telesys traces, but due to the trace granularity (2 minutes) it cannot be distinguished very well.

We see, therefore, that the AT&T Labs and Telesys workloads show evidence of both human and programmatic activity. Recall that "workaholics" (i.e., users who tend to be permanently connected) were excluded from the AT&T trace. Hence, it is reasonable to believe that the periodic activity observed is not just a response to the inactivity timeout of the system. Such periodic activity is significant (for instance, periodic stock price updates are often more important to home traders than any interactive traffic).

Overall, and quite expectedly, Figure 2 shows that human-produced activity has excellent temporal locality: a connection receiving only human input tends to stay inactive once it becomes inactive. Most of the activity seen after some inactivity seems to be programmatically produced because it coincides with easily identifiable time intervals. A good replacement algorithm should be able to perform well for both kinds of activity.

3.2 The CIRG Algorithm

The CIRG replacement algorithm (for *Conditional Inter-Reference Gap*) is a novel (to our knowledge) algorithm, loosely based on the Inter-Reference Gap model proposed by Phalke [Pha95]. The term "inter-reference

[4] The recency space is the total ordering of pages according to how recently they were last referenced.

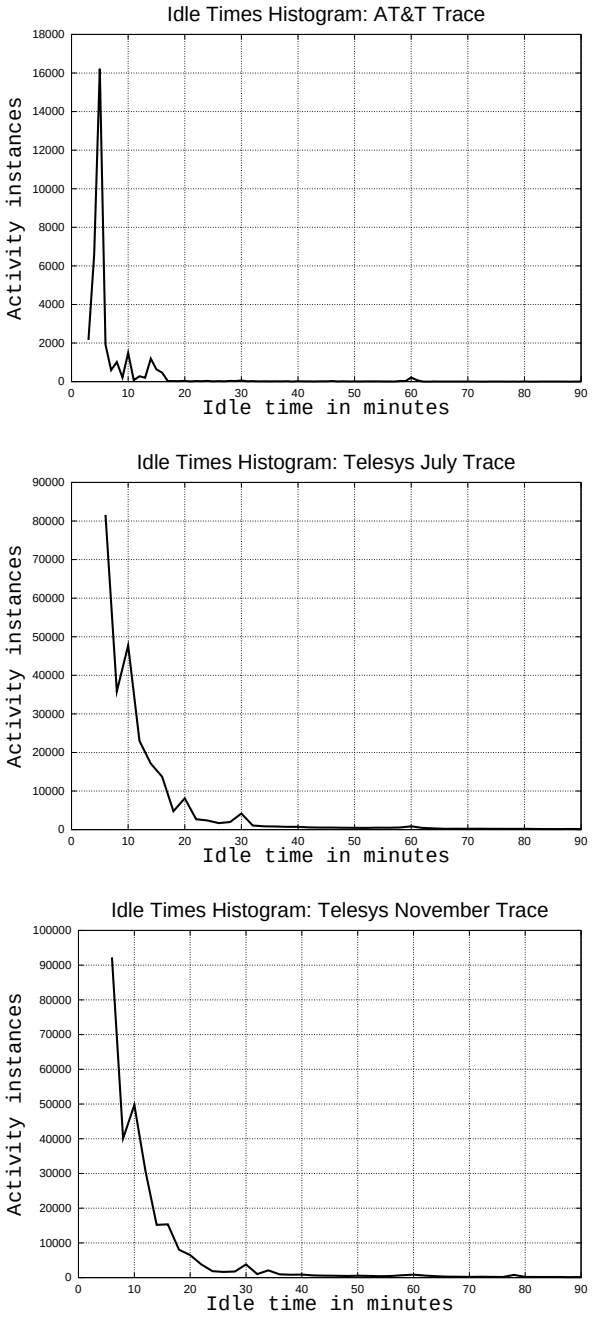

Figure 2: Idle time distributions for the three traces of our study. The left ends of all traces rise quickly and are excluded for clarity.

gap" is a synonym for idle time in the user activity domain. The CIRG algorithm keeps per-user information about the most recently encountered idle times and uses it to predict future idle times. In particular, for each user, CIRG keeps a list, which we call $idle$. (We will later generalize this to multiple lists.) The first element of the list, $idle(1)$ is the user's idle time before the user last became active. For $i > 1$, the i-th element of the list, $idle(i)$, is the total idle time for a user the last time the user was idle longer than $idle(i-1)$ time units. For example, the contents of the $idle$ list could be:

```
2, 5, 13, ...
```

(with minutes as the time unit) meaning that the user was idle for 2 minutes before she last became active. The last time the user was idle for more than 2 minutes, she stayed idle for 5 minutes. The last time the user was idle for more than 5 minutes, she stayed idle for 13, and so on.

Using the $idle$ list and the current idle time, the CIRG algorithm computes how long a user stayed idle the last time she was idle for as long as now.[5] This is used as an estimate of the total idle time until a user will next become active. In our example, given a current idle time of 7 minutes, CIRG will predict that the user will stay idle for another 6 minutes (for a total of 13). The user with the highest predicted future idle time is the one to be replaced.

This scheme can easily be generalized to multiple lists.[6] That is, with k lists, CIRG can compute how long a user stayed idle the last k times she was idle for as long as now. A reasonable estimate of future idle time will then be the average of the predicted values from each of the k lists. To keep the list lengths short, we can quantize the idle times values (e.g., by rounding all idle times to the closest multiple of 2 minutes).

We have experimented with the CIRG algorithm in several different replacement settings. Overall, CIRG is, as expected, good for detecting regular patterns in behavior. CIRG is not an ideal algorithm for some settings (e.g., virtual memory replacement) because it keeps

[5] If the user has never been idle for as long as now, a reasonable guess for the total idle time is twice the current idle time (i.e., an LRU-like estimate can be employed: the user's predicted future idle time is the same as her past idle time).

[6] The procedure for maintaining k lists, $idle_1, \ldots, idle_k$ is simple but we outline it here for completeness. Each list has an update operation that takes an idle time parameter. The update operation for the entire data structure (all lists) calls the update operation for the first list, $idle_1$, with parameter t, when the user becomes active after being idle for time t. The update operation for list i examines all elements of list $idle_i$ in order from the beginning of the list. For each element $e = idle_i(j)$, if $e < t$, the element is removed from list i and if $i < k$, the update operation is called on list $i+1$ with parameter e. Finally, t is inserted in list $idle_i$.

statistics in terms of time differences between accesses. (As we explain in [SKW99], inter-reference time is not a reliable metric of locality for modern programs because they may access vastly different amounts of memory in the same amount of time.) Nevertheless, we expected CIRG to be appropriate for the modem replacement domain, which deals with human users and real-time idleness.

An interesting aspect of CIRG is that, because it keeps statistics per user, it can recognize an arbitrary number of different periodic patterns. Thus, CIRG is a good algorithm for dealing with *multiple* different regular patterns. Such patterns could occur because of Internet tools pulling information (e.g., email clients downloading messages every 10 minutes, browsers updating the displayed stock prices page every 5 minutes, etc.). Patterns could also occur because of user habits (e.g., the user usually takes 5-10 minute breaks).

4 Experimental Setup

Given the activity traces for the studied workloads, we simulated different disconnection policies and modem pool sizes. In this section, we discuss the assumptions behind our experiments and the metrics used to evaluate different approaches.

4.1 Validity of Simulations

Our experiments consist of simulations of modem pools with different numbers of modems and different disconnection policies than those in place during tracing. Nevertheless, the actual user activity might have been different if the actual simulated policies were in effect. For instance, one common user reaction to strict idle timeout policies is to use programs that simulate activity so that the connection appears to be always in use.

This interaction of policies and user decisions is something that no predictive study involving human users can compensate for. Nevertheless, although this issue can certainly affect the exact results of our experiments, it should not affect the overall picture and our conclusions. There are two reasons for this. First, the most important of our observations are made under conditions that guarantee a quality of service similar to that of the tracing environment: busy signals are extremely rare and disconnections of users who want to be active again soon are very few. The second reason is more subtle but very important. The most likely user reaction (if any) to inconvenient disconnections is to simulate constant activity so that no disconnection occurs. This penalizes all disconnection policies equally per user (the user simply cannot be disconnected, and occupies a modem as long as she wants). Thus, a policy that causes more user inconvenience can be expected to be penalized more than a policy that causes less inconvenience (because more users will attempt to avoid disconnections in the former case). Therefore, even though the absolute values of our metrics may change, their *relative differences* for different disconnections policies will only be *accentuated* by the user's conscious choices. Hence, we believe that our results will hold true when actually employed.

Apart from the case of a conscious user choice, however, there are cases when programmatically generated activity may interact with our simulations. This means that the actual behavior might not be identical to the simulated one if the studied workloads were indeed handled with the disconnection policy under simulation. The differences are expected to be insignificant, but we discuss them here so that our assumptions are in the open. There are two kinds of systematic activities that may be affected by a different disconnection policy and may appear in our traces.

1. The user may have set her dial-in program to reconnect on non-user-initiated disconnections.
2. The user may have set her dial-in program to *not* dial in automatically every time an application attempts to make a TCP connection.

The potential problem in case 1 is dual. First, the observed behavior in our trace may include automatic reconnections after disconnections. This is behavior that would not have occurred if a different policy had not caused the disconnection. This affects only our AT&T Labs trace, which was collected in an environment with a 15 minute idle timeout. Nevertheless, the activity at 16 minutes of idle time (see Figure 2) is not high enough to suggest that this may be a significant interference. The second possible problem with case 1 is that new activity will appear after every disconnection, but this activity is not evident from the original trace. Therefore, all our simulations should be viewed as accurate under the assumption that users do not automatically reconnect on disconnect. Even in the opposite case, however, this behavior is likely to affect all simulated policies equally.

The potential problem with case 2 is that during tracing we may have missed some behavior because of the disconnection policy that was in place in the tracing environment. That is, the user might have been active, had she not been previously disconnected. (The issue clearly is relevant to programmatic activity, not interactive user activity—the user would explicitly connect if she wanted to use the network interactively.) This should not be a problem—if the behavior was not important enough to occur in the traced system, it was probably not important

enough to occur in any simulated systems. It should also be noted that our traces reflect all activity that originated from a user. We have no way of telling if some of this activity would not have occurred if the user had previously been disconnected. Hence, our simulations represent the behavior of a system where users do dial in automatically every time they are disconnected and an application tries to access the network (e.g., every time their email client needs to download messages).

4.2 Metrics

Our first metric of performance of a disconnection policy is, expectedly, the number of busy signals that the policy incurs for a given number of modems. Nevertheless, this number depends primarily on the number of users who can potentially be disconnected and the latter number is almost the same for all replacement policies. Our approach is to have an inactivity threshold (or *timeout*), t_1. Any user who has been idle for more than t_1 seconds can potentially be disconnected. The difference between different policies is in *which* user (or users) actually do get disconnected. Thus, the number of busy signals is similar across different policies that all have the same value of t_1. (The numbers are not identical because the set of users that can be potentially disconnected depends on which users were disconnected in the past.) Therefore, the number of busy signals is a good indicator of the overall quality of service, but it is not a good differentiator of the various disconnection policies.

Our other performance metrics are identical to those used in the work of Douglis and Killian. These metrics attempt to measure user "inconvenience" due to disconnections. The approach consists of setting a threshold t_2. If a user is disconnected and does not become active again within t_2 seconds, then the disconnection is deemed successful and is termed a *soft fault*. If, however, the user becomes active within t_2 seconds, the disconnection is considered a *hard fault* (or just *fault*). (Hard faults were called "bumps" in the Douglis and Killian study, a term borrowed from the disk spin-down domain [DKB95].) The *severity* of a hard fault is a linear measure of how close the user's idle time after disconnection was to the threshold t_2. (That is, severity is a linear function of idle time after disconnection, such that an idle time t_2 incurs severity 0, and an idle time 0 incurs severity 1.) Two metrics that we used for evaluating disconnection policies are the number of hard faults and the severity of hard faults. Nevertheless, the results using these two metrics were very similar, with the fault severity slightly accentuating the differences between disconnection policies. Since the two metrics yield very similar results, as well as due to lack of space, we will only use the number of hard faults as a metric in the following section.

The reason for considering only a few of the faults to be significant has to do with the perceived inconvenience of disconnections by the user. A user is wrongly disconnected if her network connection is idle but she is actively using the machine and expects to be connected (e.g., the user may be writing an email reply off-line, which she will later send). In this case, the user will notice the disconnection and reconnect explicitly. This should be counted as a fault by the system because the user was inconvenienced and the lack of network connectivity was immediately noticed. On the other hand, if a user is idle and stays idle after a disconnection, the user was probably truly inactive. In this case, the inconvenience of reconnecting is much less and the user is likely to consider the disconnection justified.

Distinguishing between hard and soft faults may seem strange to readers used to replacement problems. We believe, however, that this metric is truly appropriate for the domain. Additionally, none of the results we present change qualitatively if we consider the total number of faults as a metric. That is, if disconnection policy A is better than disconnection policy B using the number of hard faults as a metric, then A is also better than B using the number of total faults as a metric. What changes is the quantification of how much better A is.

5 Experimental Results

We performed several experiments with different disconnection policies, which we outline below. For all the experiments shown here, *we set threshold t_1 to be equal to t_2*. This means that a user can be disconnected only if idle for more than t_1 seconds, and the disconnection will be a hard fault if the user's total idle time is not at least $2t_1$ seconds. The results obtained with $t_1 = t_2$ are fully representative of all results we have observed for different values of t_1 and t_2 (i.e., we observed no systematic deviation when we experimented with $t_1 \neq t_2$).

For the CIRG algorithm, we used three lists of past idle times (i.e., the value of the parameter k of Section 3.2 is 3) and the predicted idle time was the average of the predicted time using the three lists. We also performed experiments with one and five lists and the results were very similar.

5.1 Replacement vs. Fixed Timeouts

It is, in a sense, unfair to compare a replacement policy with a policy that proactively disconnects users even when the modem pool is not fully utilized. A replacement policy is always going to inconvenience fewer users

while incurring the same number of busy signals. It is, however, interesting to quantify how much better the replacement approach is in a practical setting. Since fixed-idle-threshold disconnection policies are widespread, we show the results of a single experiment below, just as a point of reference. We will not, however, insist on such comparisons any further in the rest of the paper. If keeping users connected has no cost, the replacement approach clearly results in much less inconvenience for users.

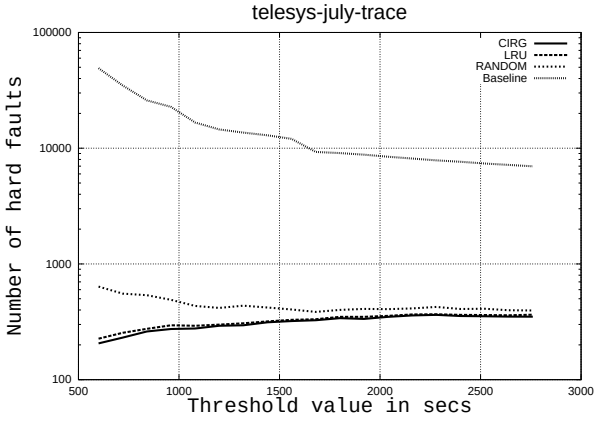

Figure 3: Log plot of hard faults for three replacement policies and a fixed idle threshold policy.

Figure 3 shows the numbers of hard faults incurred when simulating a fixed-idle-threshold disconnection policy and three replacement policies for our first Telesys trace. The simulated modem pool has 1,936 modems (this number is 90% of the maximum number of users who are connected simultaneously in this trace). The threshold value, $t_1(= t_2)$, varies from 600 to 2700 seconds (10 to 45 minutes). (Recall that t_1 is the minimum idle time before a user becomes eligible for disconnection.) For all values of the threshold, the replacement algorithms incur at least one order of magnitude fewer hard faults.

In examining Figure 3 (as well as figures that follow) it may seem awkward that the number of hard faults can rise for higher thresholds. Recall, however, that the figures shown are produced for $t_1 = t_2$. When the threshold t_1 increases, t_2 also increases, thus causing many soft faults to be considered hard faults.

5.2 Comparison of Replacement Algorithms

A more interesting experiment concerns the comparison of different replacement algorithms. The first question to be answered is whether a simple replacement algorithm, like RANDOM, is good enough. RANDOM corresponds to the simple idea of replacing any user who has been idle for more than t_1 seconds. The point of reference for the comparison is the LRU algorithm—a common benchmark in replacement problems. LRU replaces the user who has been idle the longest (as long as this is more than t_1 seconds, in our case). The LRU algorithm has been used in replacement settings ranging widely (e.g., from virtual memory to web caching [ASA+95]). The next step is to see whether a specialized algorithm can perform better than LRU. As we will see, the CIRG algorithm meets this test.

Fixed Threshold, Variable Modem Pool Size Results. Figure 4 shows the results of simulations for all three replacement policies and a wide range of modem pool sizes. The value for threshold t_1 (and, consequently, t_2) is set to 600 seconds (10 minutes). We will later examine how our results change under different threshold values.

The ranges of modem pool sizes that we examine contain all reasonable values for practical applications, given each workload. That is, for the low end of the studied range, the workload incurs too many busy signals or too much user inconvenience due to disconnections. For the high end of the studied range, the workload incurs practically no hard faults or busy signals. For the Telesys traces, the chosen range begins at around 70% of the maximum number of simultaneously connected users in the trace.

As can be seen from Figure 4, CIRG and LRU are significantly better than RANDOM in terms of hard faults incurred. The difference ranges from a few tens of percent to over 1000%. Even more importantly, the relative difference is very significant for large modem pools, which are the ones that are going to be encountered in practice. For instance, it would be quite realistic to handle the workload of the July Telesys trace with around 1,900 modems (this setup would suffer around 15 busy signals in 10 days of use). For 1,906 modems, however, LRU incurs 370 hard faults, CIRG incurs 341, and RANDOM incurs 926 faults. Thus, in terms of user inconvenience, RANDOM may not be a good choice for a practical setup. Additionally, we can see that CIRG outperforms LRU. LRU systematically incurs 5% or more, and sometimes over 20% more faults than CIRG.

The second observation we can make in Figure 4 concerns the overall quality of service. We see clearly how a disconnection policy helps maintain a low busy-signal count with fewer modems. Combined with a replacement algorithm that yields a low number of hard faults, the result is a policy that guarantees good service under heavy demand. For instance, the July Telesys workload can be handled using CIRG with 1,875 modems while incurring only 100 busy signals and 500 hard faults dur-

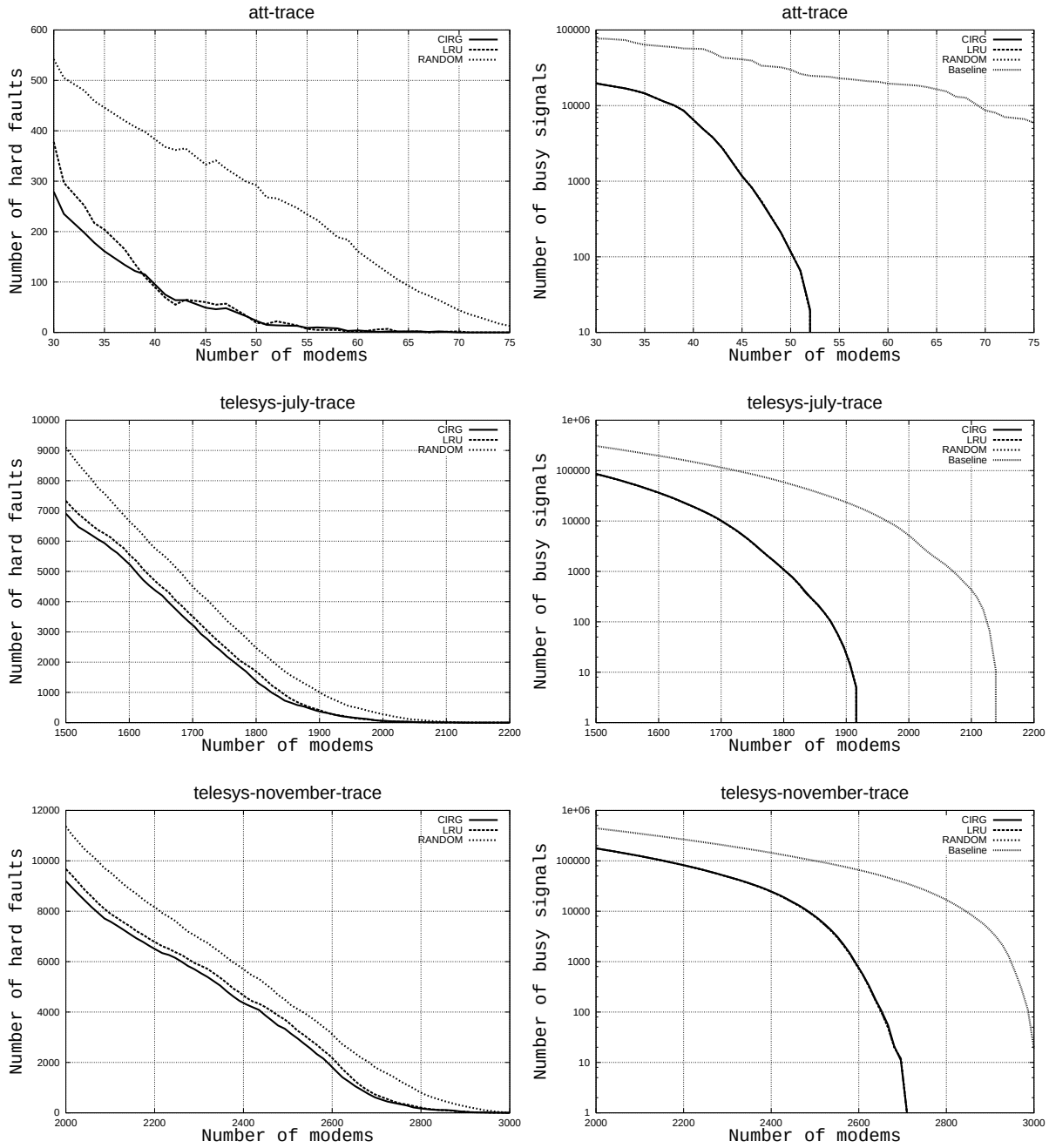

Figure 4: Plots of hard faults (linear scale) and busy signals (log scale) for all traces and a variable number of modems. The value for threshold t_1 is 600 seconds (10 minutes) and $t_2 = t_1$. The baseline in the busy signal plots is the number of busy signals incurred with no disconnection policy in place (for the AT&T trace, no user would get explicitly disconnected so this baseline is hardly meaningful). For the busy signal plots, often all three curves for RANDOM, LRU, and CIRG coincide and cannot be distinguished.

ing the course of 10 days (recall that this trace contains over 315,000 connections and user-initiated disconnections). This is 13% fewer modems than would be needed if the system did not disconnect idle users. For comparison, RANDOM needs 1,957 modems to reach 500 hard faults.

Fixed Modem Pool Size, Variable Threshold Results. Figure 5 shows how the numbers of hard faults and busy signals vary for different values of the threshold t_1. The modem pool sizes simulated are 52 modems for the AT&T Labs trace, and 1,936 and 2,721 modems for the Telesys July and November traces, respectively. These modem pool sizes are 90% of the maximum number of modems needed simultaneously for the Telesys traces. For all three traces, the modem pool size is such that few busy signals are incurred for a 600 second threshold. Thus, the sizes are such that they could very well be used in practice to handle these workloads.

For the AT&T Labs trace, the values of t_1 examined are between 300 and 900 seconds (5 and 15 minutes, respectively). Since the environment where the trace was collected had a 15 minute inactivity timeout, these values seem reasonable. The values are certainly more aggressive than what would be expected from a typical ISP, but since the setup uses static IP addresses, the cost of a disconnection is small (i.e., only the reconnection delay and not the termination of open sessions). For the Telesys traces, we experimented with threshold values ranging from 600 seconds (10 minutes) to 2700 seconds (45 minutes). These are more realistic settings for a general purpose ISP where disconnections are more costly due to the dynamic IP address assignment.

As we observe in Figure 5, the number of hard faults may increase for higher threshold values, but not dramatically. LRU and CIRG remain the best predictors of future idle time, with CIRG performing slightly better. RANDOM performs significantly worse than both for all settings that incur a tolerable number of busy signals (e.g., below 1,000 for the Telesys traces). The number of busy signals incurred by all policies increases, expectedly, for higher threshold values. By increasing the minimum inactivity threshold for disconnections (t_1), the number of users who can potentially be disconnected decreases rapidly (as seen in the histograms of Figure 2).

We can see from Figure 5 that with 10% fewer modems than the maximum needed simultaneously, we can get a low number of busy signals and hard faults, for high threshold values—over half an hour for the July Telesys trace. The November Telesys trace has worse locality and quickly incurs many busy signals for threshold values above 20 minutes. This is to be expected: users of Telesys who stay idle long, are likely to have dedicated phone lines for modem connections. Nevertheless, the increase of Telesys usage between the Summer and Fall trace is mainly due to students. The percentage of students who can afford dedicated phone lines is lower than the corresponding percentage of faculty and staff. Thus, it is natural to have proportionally fewer users who are idle for long in the November trace. We believe that the July trace is more representative of typical ISP subscribers' behavior than the November trace, but have taken no steps towards verifying this.

Soft Faults Finally, we present in Figure 6 the numbers of soft faults incurred by all the studied policies. The first column presents the soft faults under constant threshold t_1 and variable modem pool size. The second column presents the numbers of soft faults under constant modem pool size and variable threshold. All parameters are the same as for the corresponding plots in Figures 4 and 5. As we explained earlier, the number of soft faults is not an accurate indication of the performance of a policy—soft faults represent correct predictions that lead to successful disconnections. Nevertheless, we show the soft fault measurements for completeness. The reader can refer to these plots to confirm that the predictions made by CIRG and LRU were very often accurate—the number of soft faults is usually many times higher than the number of hard faults.

6 Implementation Considerations and Applicability

The idea advocated in this paper is to consider disconnecting users only if a modem pool is fully occupied. Following this approach, we studied different algorithms for picking users to disconnect. Unfortunately, neither modem servers[7] nor modems have native support for the policies we describe. Fortunately, however, the approach is very easy to implement.

The most straightforward implementation would be one that does not strictly perform replacement but keeps a fixed number of modems unoccupied, as long as users with idle time more than t_1 exist. That is, for a small number n, if the number of available modems drops below n and there are users idle for more than t_1 seconds, the system will disconnect one of these users and repeat the process. If there are no users idle for more than t_1 seconds, then all modems can be occupied and other connections will get a busy signal. This implementation

[7] There is no established term for the devices that manage and implement Internet protocols over multiple serial ports. Depending on the exact functionality and marketing decisions of each maker, these are called *modem servers*, *remote access servers*, *terminal servers*, or *communications servers*

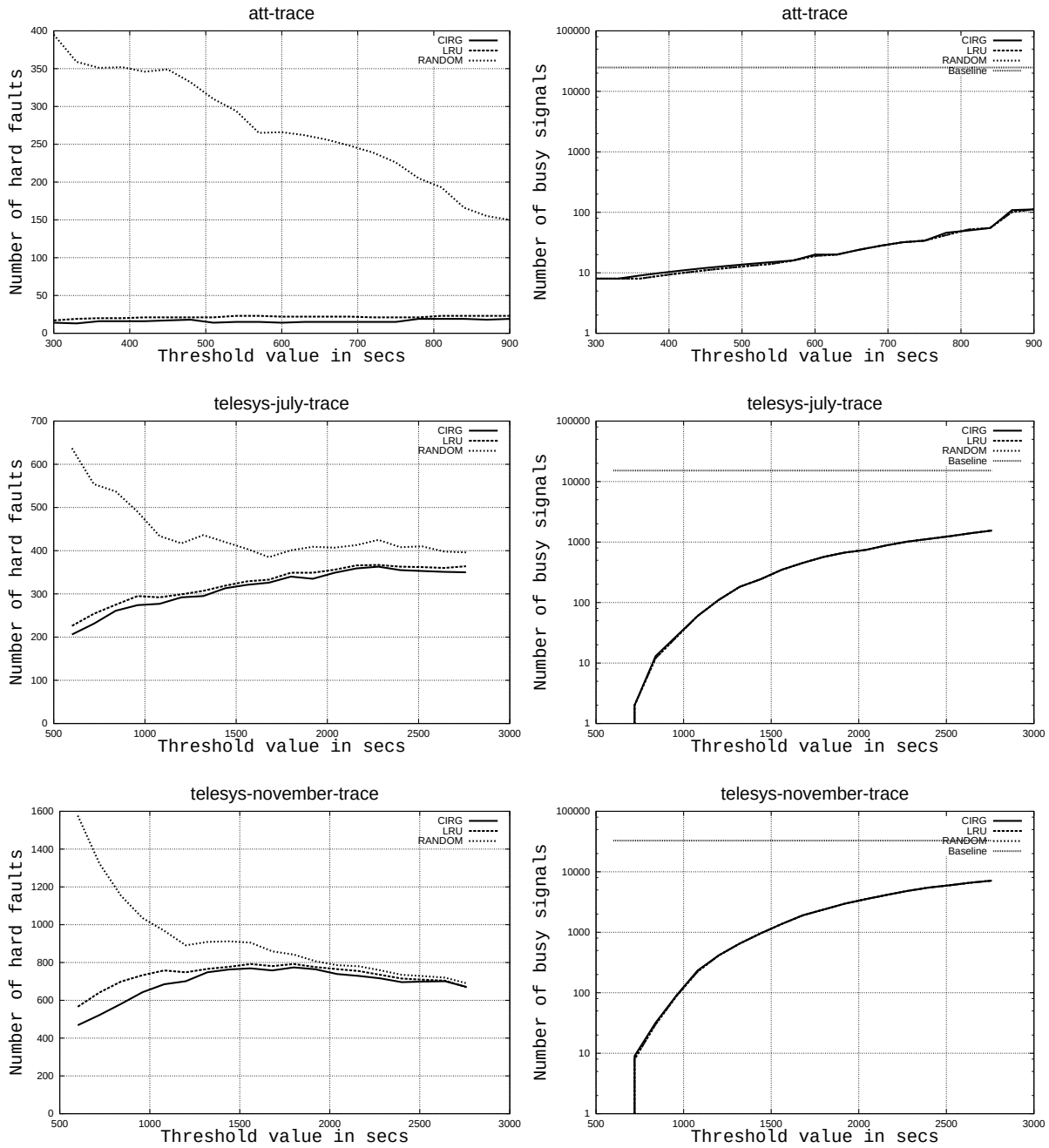

Figure 5: Plots of hard faults (linear scale) and busy signals (log scale) for all traces under variable threshold t_1 (and $t_2 = t_1$). The number of modems for each workload is fixed to a value that yields few busy signals for a 10 minute threshold (for the Telesys workloads, this is 90% of the maximum number of simultaneously connected users in the trace). The baseline in the busy signal plots is the number of busy signals incurred with no disconnection policy in place. For the busy signal plots, often all three curves for RANDOM, LRU, and CIRG coincide and cannot be distinguished.

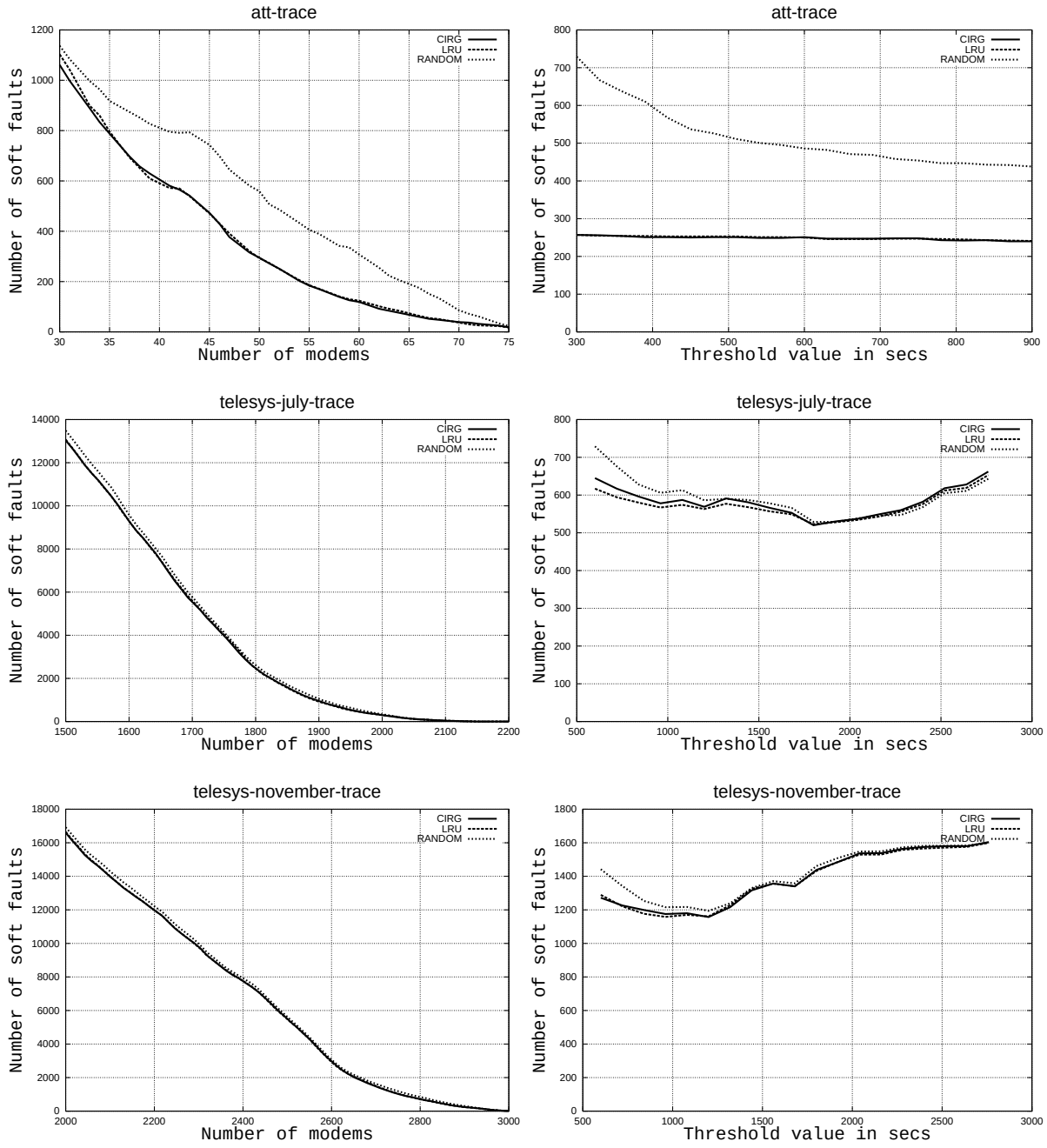

Figure 6: Plots of soft faults for all traces under variable number of modems (first column) and variable threshold (second column). The parameters not shown on the plots are the same as those used in Figure 4 for the first column and in Figure 5 for the second column. The LRU and CIRG curves often coincide and cannot be distinguished.

works well because it does not penalize the system in cases of heavy load that cannot be alleviated with disconnections (all modems can be used). In case of a load that could be lightened with disconnections, it only penalizes the system by keeping n modems available. The number n could be quite small relative to the pool size. For the Telesys pool of over 3,000 modems, a value of $n = 20$ would be reasonable (see below for the rate of connections and disconnections in the Telesys traces).

Additionally, what makes a sophisticated modem disconnection policy easy to implement is that the data and decisions involved are not significant for modern machines. For the Telesys modem pool (which is among the largest unified pools encountered in practice) one has to manage up to a few thousands of modems at any time and a total number of users in the low tens of thousands. Handling replacement policy data structures with this many entries is a simple matter. Even for CIRG and LRU, the more "costly" policies among the ones we studied, updating the data structures and selecting a user to disconnect was, at most, a matter of milliseconds. The total memory required was less than 2MB for CIRG and less than 100KB for LRU at any time. Furthermore, the input data change at human-time rates. Typically, 5 to 10 connections or disconnections per minute were observed in the Telesys trace. As we saw in our experiments, our polling interval of 2 minutes was sufficient for obtaining data such that accurate predictions can be made.

In fact, the problem is computationally simple enough that even a centralized remote implementation is sufficient. (This is certainly not the only option but we discuss it here because of its simplicity.) That is, a remote workstation can be periodically polling all the terminal servers and sending messages that will initiate user disconnections. Many modern communications servers support SNMP (see [CFSD90] and [MR91] for the protocol and the relevant MIB entries), so both the polling and the disconnection commands can be sent remotely over the Internet. Alternatively, a centralized implementation with small proxies that will perform the disconnections at every server seems to be a simple option.

To see how feasible this is, during our trace collection, we polled the over 100 Telesys terminal servers remotely over the Internet using the "finger" command (which uses the *Finger user information protocol* [Zim91]). This method is clearly inefficient because the protocol is not optimized for periodic polling and because the information we needed was less than 5% of the total transmitted data. Nevertheless, our polling took around 50 seconds when done serially and around 15 seconds when done with one process per terminal server (the vast majority of processes finished within 3 seconds but a couple took longer). Although we have no way of analyzing the delay, it is reasonable to assume that it is primarily due to delay of processing at the terminal server and secondarily due to network latency. The former can be minimized with a less inefficient polling protocol. The latter could be reduced if our machine storing the trace was at closer network proximity of the servers. Nevertheless, even the 15 seconds taken for a remote, inefficient poll are perfectly acceptable—user statistics will not have changed significantly in this time.

7 Proactive Disconnections

The main idea explored in this paper is that of performing user replacement in modem pools. The general problem we are addressing, however, is that of predicting future idle times for user connections to the Internet. Thus, our mechanisms could find application in other domains. A prominent opportunity appears in the case of proactive user disconnections. This is the problem studied by Douglis and Killian [DK99]. Their adaptive timeout technique aims at reducing the total connect time across users. Proactive disconnects are interesting when either the service provider or the user is charged more for longer connection times. (As Douglis and Killian admit, "Examples of this in the general ISP market are rare, but some services that function effectively as ISPs do observe this property.") The tradeoffs involved in such a setting are interesting. For instance, there may be a connection initiation fee for users, so a disconnection should be performed only if the charge of staying connected is higher than the charge of connecting anew.

The formulation of the proactive disconnection problem is similar to that of the replacement problem. There are two thresholds, t_1 and t_2. t_1 is the minimum idle time before a user is eligible for disconnection. t_2 is the minimum user inactivity time after an automatic disconnection, for the disconnection to be considered successful. ("Successful" may mean cost-effective, psychologically acceptable, or both.)

It is interesting to consider a proactive disconnection algorithm based on the same predictions of future idle time as those made by our replacement algorithms. We have implemented an algorithm called *CIRG-proactive* that uses the same information as the CIRG replacement policy to estimate the future idle time for every user and disconnects any user with an expected future idle time that is longer than t_2. For instance, assume a value of 5 minutes for t_2, and a user who has been idle for 6 minutes and has an $idle$ list:

```
2, 5, 13, ...
```

(these values in minutes). The user will be disconnected,

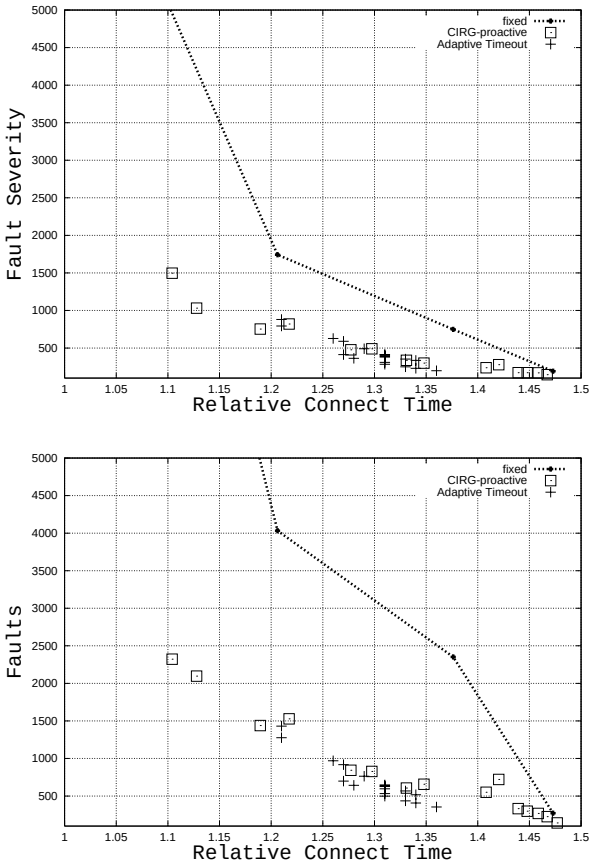

Figure 7: Fault severity and absolute number of faults vs. relative connect time for three policies: a fixed timeout policy, CIRG-proactive, and the Douglis and Killian technique ("adaptive timeout"). Connection times are normalized with respect to the connection time of an optimal, off-line policy.

since her expected future idle time is 7 (for a total idle time of 13).

We have performed experiments with CIRG-proactive and compared it to the Douglis and Killian technique. We should emphasize that our results are preliminary. The reason is that we have not re-implemented the Douglis and Killian mechanism. Hence, we could not examine its effects on our long traces (the Telesys traces). Instead, we use the measurements already computed by Douglis and Killian for the AT&T Labs trace and compare them to the performance of CIRG-proactive.

Figure 7 presents the results of our comparison. It shows two scatter plots: one of fault severities and connection times and another of absolute fault counts and connection times. Both show results for a fixed timeout policy, CIRG-proactive, and the Douglis and Killian technique applied to the AT&T Labs trace, with 15 workaholics excluded. Connection times are normalized with respect to the connection time of an optimal, offline (i.e., clairvoyant) policy. That is, the optimal policy would encounter 0 faults for a relative connect time of 1. The value of t_2 is 5 minutes and the values of t_1 for the fixed timeout policy are 2, 5, 10, and 15 minutes. (These values are chosen to be identical with those used by Douglis and Killian in their study.) The values of t_1 for CIRG-proactive are all integral minute values from 1 to 15 minutes. 18 data points for the Douglis and Killian technique are plotted, each for a different combination of adaptivity (multiplicative or additive) and different parameters. As we are comparing to the approach as a whole, we do not distinguish between the different flavors of the Douglis and Killian technique (for this, the reader is referred to [DK99]).

We can see from Figure 7 that CIRG-proactive performs on average just as well as the Douglis and Killian policy and they both perform significantly better than fixed timeouts. Nevertheless, the values for the Douglis and Killian approach are clustered together in the connect-time/user-inconvenience space, while CIRG-proactive offers a wider range of options in the tradeoff between user inconvenience and total connect time. It is worth noting that CIRG-proactive offers this wide range of options with only a single parameter that can be tuned (the t_1 minimum idle time for disconnection) while the Douglis and Killian approach has several degrees of variability (additive vs. multiplicative adaptivity with two numeric parameters for each variant). The wide range of CIRG-proactive values means that we can argue that CIRG-proactive is strictly better than a fixed timeout policy: For each fixed timeout point in the plot, we can find a CIRG-proactive point that is below it and to its left. That is, for each fixed timeout setting, we can find a value of t_1 for CIRG-proactive so that it incurs both a lower fault severity (or fewer faults) *and* a lower total connect time. The Douglis and Killian technique provides no such guarantee for the values shown.

8 Conclusions

In this paper, we examined a disconnection policy for modem pools that performs replacements: users are disconnected only if not enough modems are available for other users to connect. Our idea is certainly not revolutionary but it is nicely evolutionary. If any disconnection policy based on inactivity is to be used, a replacement scheme works best and is easy to apply in practical settings. In our study, we examined several replacement algorithms, and compared their performance in terms of busy signals and user inconvenience.

Perhaps more importantly, however, this paper studied the regularities that arise in Internet users' inactivity times. Based on our analysis, we proposed the CIRG algorithm and showed that it is a very good predictor of future idle times. Our hope is that the main elements of the CIRG approach will also find applications in other domains.

Acknowledgments

We would like to thank Fred Douglis and AT&T Labs for supplying us with the first of the traces used in our study. We would also like to thank the anonymous referees for their extensive comments. Finally, Don Batory's careful reading helped improve the paper.

References

[Aca98] Academic Computing and Instructional Technology Services. Telesys update, September 1998. `http://www.utexas.edu/cc/newsletter/sep98/telesys.html`.

[Ano97] Anonymous. Steamed customers sue AOL for busy signals. *Information Week*, January 15, 1997.

[ASA+95] Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Stephen Williams, and Edward A. Fox. Caching proxies: Limitations and potentials. In *Fourth International WWW Conference*, 1995.

[BI94] Daniel Barbará and Tomasz Imieliński. Sleepers and workaholics: Caching strategies in mobile environments. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1–12, 1994.

[CFSD90] J. D. Case, M. Fedor, M. L. Schoffstall, and C. Davin. RFC 1157: A simple network management protocol (SNMP), May 1990.

[DK99] Fred Douglis and Tom Killian. Adaptive modem connection lifetimes. In *Proceedings of the 1999 USENIX Annual Technical Conference*, pages 27–41, Monterey, California, June 1999. USENIX Association.

[DKB95] Fred Douglis, P. Krishnan, and Brian Bershad. Adaptive spin-down policies for mobile computers. *Computing Systems*, 8(4):381–413, 1995.

[GC97] Gideon Glass and Pei Cao. Adaptive page replacement based on memory reference behavior. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, 1997.

[MR91] K. McCloghrie and M. T. Rose. RFC 1213: Management information base for network management of TCP/IP-based internets: MIB-II, March 1991.

[Pha95] Vidyadhar Phalke. *Modeling and Managing Program References in a Memory Hierarchy*. PhD thesis, Rutgers University, 1995.

[SKW99] Yannis Smaragdakis, Scott Kaplan, and Paul Wilson. EELRU: Simple and effective adaptive page replacement. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, 1999.

[Zim91] D. Zimmerman. RFC 1288: The Finger user information protocol, December 1991.