Adaptive Modem Connection Lifetimes

Fred Douglis Tom Killian

AT&T Labs–Research, Florham Park, NJ, USA

July 15, 1998

Abstract

Internet Service Providers sometimes go to great lengths to minimize dial-up connection times, in order to make the best use of limited resources. Typically they disconnect users after a fixed period of complete inactivity, such as 10–15 minutes. We propose adaptive time-out policies that take past history into account, and we evaluate some of these policies using a trace from a production environment. We find that adaptive policies can reduce cumulative connection times and average simultaneous usage by about 10–20% compared to a conservative fixed threshold, in exchange for a moderate increase in the number of disconnections that inconvenience the user.

1 Introduction

In computers, as in real life, using resources often comes at a cost in proportion to the duration of use. That cost can typically be reduced by discontinuing use temporarily—assuming that the resource will not be needed for a period of time—and then using the resource again in exchange for some possible start-up overhead. Often, the decision to discontinue use is based on a fixed time-out interval: one waits for the resource to be idle long enough that one can assume it will continue to be idle a while longer, long enough to avoid antagonizing the user. It must also be idle long enough to amortize any start-up overhead and result in a net gain from discontinuing use.

One category of resource use with variable timeouts is anything that consumes energy on a battery-operated device such as a mobile computer. Examples include spinning down the disk [3, 2, 7], putting the CPU in a low-power or reduced-power state [9], and suspending the display. In the communications domain, an example of this tradeoff has been demonstrated in the IP-over-ATM area. One can use a variety of algorithms to decide when to relinquish a virtual circuit that is being used to transmit a sequence of IP datagrams [5]. An algorithmic approach that is common to many of these systems has been termed a "random walk," in which a parameter varies over time in response to past history [4].

Modems are another type of limited resource, with some interesting properties that make straightforward approaches somewhat problematic. With most telephone-modem-based ISPs, a computer connects to the Internet via a pair of modems, one owned by the user and one owned by the ISP. The ISP provides a temporary IP address using PPP or some other protocol. If the modem connection is terminated, the computer is not guaranteed to get the same IP address the next time it connects. This means that proactively disconnecting the modem will likely terminate any existing TCP connections using the older IP address.

Furthermore, even if the IP address is fixed, there is a significant delay in reestablishing a PPP connection: we have measured 30–40s for dialing, training, and PPP negotiation. This delay can be annoying to the user, and it can also cause application-level or system-level timeouts to occur. This might result in a transient error (for instance, downloading a Web page) or a more serious one (such as terminating a telnet session, forcing it to be reestablished and losing any state in it).

At the same time, constant use of an ISP's modem is generally discouraged. One way to discourage use is an economic disincentive: AT&T, MCI, and other ISPs recently changed their "unlimited" Internet Access to have a limit of 150 hours per month before surcharges are applied. This limit was imposed because a small fraction of the user community would use the system virtually non-stop, forcing the ISPs to continually increase capacity or risk having other customers encounter busy signals.

Another way to limit modem use is to proactively disconnect "idle" users and suffer the consequences. It appears that many ISPs will disconnect a completely idle user after some fixed interval that varies in the range of about 10–60 minutes. But users do not like to suffer the 30–40s delay reconnecting to the ISP after being reconnected, nor the possibility of a busy signal, nor the possible loss of TCP sessions that are open but inactive and which get reset after a hangup. A simple solution, from their perspective, is to run a daemon that uses the modem periodically, more often than the usual timeout. Checking electronic mail is an obvious example of such a daemon. Since completely periodic use, such as checking mail every 5 minutes, could be detected and compensated for, a more sophisticated approach would be to access the modem at somewhat random intervals for as long as the client wishes to keep the modem connection alive. As a result, some ISPs drop connections after an extended period of activity, such as a day, regardless of ongoing use.

Our goal was to see whether another approach to disconnecting idle modems might reduce modem usage without significantly inconveniencing users. Here we apply an "adaptive" timeout technique, which was previously applied to the domain of disks [2], to the domain of modems. The basic approach is the same: an ISP would start with some timeout interval, and then vary that timeout interval for each modem over time based on usage patterns. A shorter interval will generally reduce modem usage but be more susceptible to making a mistake, i.e. disconnecting a modem at a time when it will be used again too soon to make disconnecting it worthwhile. One decreases the timeout when the previous interval successfully predicted a long enough idle interval, and increases it when the idle interval proved too short. One can also simultaneously track *patterns* of bursts of activity, for instance accesses for just a few seconds every 15 minutes, in order to predict the next idle interval and disconnect immediately. We will elaborate on these approaches, and quantify the outcomes, later in this paper.

The environment in which we apply adaptive modem timeouts is particularly conducive to proactive hangups, despite the problems mentioned above. This is because it uses static IP addresses, so a modem can be disconnected and later reconnected without affecting running applications if the applications do not attempt communication during the downtime. If they do communicate, they must have a long enough timeout to tolerate the reconnect delay.

Trace-driven simulations indicate that varying the timeout adaptively has significant benefits over relatively short fixed timeout intervals (2–10 minutes). For instance, all the adaptive algorithms we studied resulted in many fewer bad choices about when to hang up the modem than a short fixed threshold, with at most the same cumulative connection time. The longest fixed threshold we considered, 15 minutes, substantially drops the number of bad choices by comparison to the adaptive and shorter fixed thresholds, in exchange for more connect time.

The rest of this paper is organized as follows. The next section discusses the metrics we use to evaluate the costs and benefits of modem disconnection. Section 3 discusses our target environment and the traces we collected from it. Section 4 discusses several different approaches to varying the timeout threshold. In Section 5, we describe the experiments we performed, the results of which appear in Section 6. Section 7 concludes.

2 Metrics

In deciding when to disconnect a modem, one must balance the inconvenience to the user against the benefit to the ISP due to reclaiming the modem.

2.1 Inconvenience

From the user's perspective, each time the modem disconnects there is potentially some inconvenience. Waiting several seconds (typically on the order of a half-minute) for reconnection is a mild annoyance if the user hasn't used the network for a long time, but it would be a greater problem if:

- the idle time was very brief, or
- the reconnection adversely affected running processes (such as terminating a telnet session).

Occasional annoyances are probably fine, while frequent ones would be intolerable and should be avoided.

Our passive monitoring of the modem pool won't tell us when a connection is terminated, so we focus on idle time. In our initial experiments, we use a parameterized idle-time threshold. If the modem has been idle that long after being disconnected, then disconnecting was desirable. If it has not, then disconnecting was undesirable. In the adaptive disk spin-down work [2] on which we model our system, undesirable disk spin-downs were referred to as "bumps," and we adopt that terminology here. We use a default of 5m of inactivity after a disconnect as the required idle time to avoid a bump, and then compare this with a more conservative 10m threshold.

In [2], a bump was considered a binary event: either a bad spin-up occurred, or it didn't. The study alluded to considering some bumps as more egregious than others. We apply that reasoning here by counting bumps either as binary events or as fractional numbers. In the former case, we charge the same amount (1) any time the idle-time threshold is not met. In the latter case, we charge $1 - \frac{I}{T}$, where I is the idle time since disconnect and T is the threshold. Thus a reconnect immediately after a disconnect is charged 1 unit, whereas a reconnect just before the threshold T is crossed is charged almost nothing. One can view these two values as a count of the total *number* of bumps and the *severity* of the bumps.

2.2 Benefits

Benefits to the ISP accrue when a modem (and the phone line to which it is attached—there is effectively a one-to-one ratio) is used less. In the event that an ISP is charged in proportion to connect time, the total *connect time* across all users is relevant. (Examples of this in the general ISP market are rare, but some services that function effectively as ISPs do observe this property. For instance, AT&T has a corporate network for employees with a toll-free number, and it pays per-minute charges which are in turn charged back to the users of the dial-up service.)

Another benefit is the potential ability to reduce the total number of modems in the ISP (or conversely, to serve more users with the same number of modems). We consider both the *average simultaneous user count* and the *maximum simultaneous user count*. The average is an indication of how one could provision the modem pool to satisfy demand most of the time while rarely running out of resources. The maximum shows how to provision the modem pool in order never to run out of resources at all. Another useful metric might be the 90th or 95th percentiles, rather than the mean, but we have not yet considered percentiles other than 100%.

3 Environment and Traces

This study was performed in the context of the Speedy Asymmetric Intranet Link (SAIL) project in AT&T Labs. SAIL uses cable modems to transmit data at high-speed to home users, with the "upstream" link over a 28.8 kbps telephone modem. (This configuration is typical for cable modems, with only about 20% of cable plants supporting two-way communication [6].) The upstream link uses the Point-to-Point Protocol (PPP) [8], with dynamically assigned IP addresses. IP endpoints, however, deal only with the downstream cable-modem addresses, and these are statically assigned. Using static downstream addresses enables the modem pool to disconnect a user after a relatively short period of inactivity, usually 10–15 minutes, without normally impacting connections beyond the dial-up overhead. It takes 30–40s to reconnect over the telephone modem; the cable modem is virtually always accessible.

Figure 1 depicts the architecture of SAIL. SAIL connects AT&T Labs–Research with homes throughout the northern part of New Jersey, by distributing data over several cable head-ends. The upstream connections come in through modem pools at or near the cable head-ends and are backhauled into the AT&T Labs network. Downstream data flows over a collection of T1 lines and

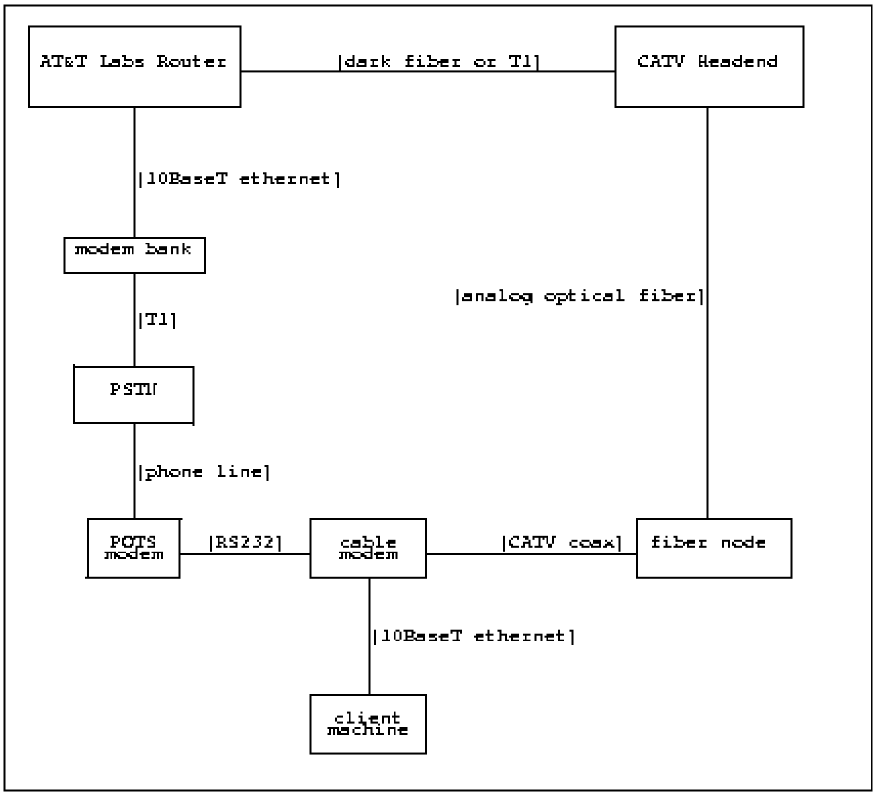

Figure 1: SAIL architecture.

optical fiber to four regional cable television headends.

We used two sets of traces. The first was a packet-level trace that was collected in November, 1997 in one of the modem pools. While activity could be detected by the existence of a packet to or from a particular host, a voluntary disconnection of the modem would be indistinguishable from an idle connection. We used this trace initially as a proof-of-concept, but felt that a live system would not want to make decisions on a packet-by-packet basis. We do not discuss the first trace further in this paper.

The second trace was collected over a one-week period in May, 1998 by periodically polling each of the modem pools for activity. The modems would report which users were active, and how many bytes had been transferred since the most recent connection. Activity could be inferred by a change in the byte-count, while disconnection could be detected by the complete absence of a particular userid in the report. The polling granularity was set at 30 seconds, a somewhat arbitrary choice that was based on the balance between the desire not to load the modems unnecessarily and the desire for current statistics. With a 30-second granularity, one could not tell whether a newly detected connection was established just after the previous poll (i.e., 30s previously), just before the current poll, or anytime in-between. We take the conservative approach of "charging" a connection from the earliest point when it may have become active, which we approximate as 29s before the current polling interval.

Although the script that gathered the latter trace was capable of actually disconnecting modems, it acted primarily in a non-intrusive capacity. The reason for this was two-fold. First, acting on live connections permits one to apply only one policy, since after disconnecting a modem, one cannot wait a minute or two and disconnect it again. In practice, the modems use a fixed-threshold policy with a 15-minute timeout in the vast majority of cases and an infinite timeout in a few cases where users previously complained about unwanted disconnections. This would limit us to implementing policies that would disconnect in at most 15 minutes, since anything longer would have the real system "beat us to the punch" in disconnecting the user.

The second reason was a fear that disconnecting live users might upset them if we were too aggressive. We wanted to simulate the effect first. The sole exception to this "hands off" approach was the modem connection of one of the authors, which was disconnected using one of the simple adaptive schemes described below. Over the one-week collection interval, his modem was disconnected by the script just over 200 times, and 43 of them (21%) were deemed by the script to be premature: the subsequent idle time wasn't long enough. However, completely subjectively, at no time was the idle time since the previous access so short as to prove particularly annoying.

4 Variable Timeouts

In contrast to the fixed timeout intervals currently in use, we consider two types of variable timeout intervals, which are complementary. Adaptive timeouts use an interval that fluctuates over time as a function of past history. Predictive timeouts use a small window of past history to predict when the next access will follow the same pattern. In addition, we consider an off-line optimal timeout algorithm as a baseline against which to compare other approaches.

4.1 Adaptive Timeouts

Our adaptive algorithm attempts to keep the number of undesirable disconnects low, relative to total connect time. At any given time, each modem has a timeout T_m associated with it. If the modem has been idle for T_m seconds, it is disconnected. The next time the modem is used, the idle time since the disconnect, I, is compared against a minimum idle time M. If $I \geq M$, then the disconnect was "acceptable" and the threshold T_m is reduced. Otherwise, it was a "bump," and T_m is increased.

A key question with this approach is how much to adjust the threshold. We use the same approach as with adaptive disk spin-down [2], permitting both additive and multiplicative modifiers along with minimum and maximum values. A given policy specifies a starting threshold T_s, how to adjust on a acceptable disconnect or a bump, and a range. If the adjustments are additive, then we add a fixed amount on a bump, or subtract on an acceptable disconnect. We repeat the terminology of [2], using α_a or α_m to adjust on a bump (the subscript a denotes an additive adjustment and the subscript m denotes a multiplicative adjustment), while β_a and β_m apply in the case of an acceptable disconnect. As with disk spindown, experience suggests that one should decrease the threshold slowly on success and increase it more rapidly on failure.

Multiplicative adjustments tend to affect the threshold more rapidly. For instance, we may multiply by 1.4 on a bad disconnect and divide by 1.1 on an acceptable one. The threshold will nearly double on back-to-back bumps.

The range is used as a sanity check. By preventing the threshold from falling below some minimum (e.g., 1 minute), we avoid entering a state in which a disconnect might occur through a perfectly normal gap between packets. By keeping it from rising above some maximum (e.g., 15 minutes), we prevent an adaptive algorithm from becoming strictly worse than the most conservative fixed threshold we would apply. (In addition, since the original trace applied a maximum, the trace would indicate a disconnect even if the replay would have kept the connection intact.)

4.2 Predictive Timeouts

In studying access patterns, it became apparent that some modems were used at regular intervals. For example, a host might check mail or perform some other sort of "keep-alive" at 15-minute intervals, and with a 15-minute fixed timeout, the modem might either never disconnect or repeatedly disconnect just moments before the next access. It seemed desirable to detect these patterns and disconnect quickly after each burst of activity, under the assumption that the next activity would not occur for many minutes. One must use a history buffer to look at some number of past accesses for a pattern; in our case we looked at the past 5 active and idle periods.

Being too aggressive in detecting these patterns could of course be a mistake. If one were to assume that because every recent interval of activity was brief (a few seconds), the next interval will be equally brief, then normal activity might also trigger a disconnect during a momentary lull. This situation is analogous to the case of a very short adaptive threshold, and is addressed the same way: a separate minimum idle time is established. In our simulations we used a 2-minute threshold, meaning that when periodic activity was detected, the timeout threshold would be dropped temporarily to 2 minutes if it were not already below that point.

What if the prediction is wrong? A wrong prediction would mean that past recent history was not a good predictor of the next access. We increment a counter, and stop predicting for a client if the count of incorrect predictions exceeds a threshold (currently 2 mistaken predictions). In that event, the adaptive modifiers to the threshold still apply, but we don't shortcut the threshold just because a pattern seems to develop.

4.3 Off-line Optimal

Given a trace, it is possible to look ahead to the next access and disconnect the modem if and only if the next access will be M seconds in the future. (In fact, assuming the time cost of reconnecting is C, one could take this a step further and reestablish the connection C seconds before the next access [2], but we do not consider that here.) We simulated the off-line optimal threshold as a way to compare different algorithms and evaluate any additional room for improvement.

In the case of connect times, we use the total connect time for the off-line optimal as a baseline, and report other connect times as a ratio by comparison to the optimal. In general, a 2-minute fixed threshold has less cumulative connect time than the optimal (because the optimal requires a 5-minute period of inactivity), in exchange for an exceptionally high number of bumps. The same holds true of the average and maximum number of modems in use over time, which are also normalized to the optimal.

The number of bumps is zero for the off-line optimal, so a ratio does not apply. We use absolute counts, displayed on a log scale, with the zero value for the optimal case represented by a small nonzero number.

4.4 Required State

With fixed thresholds, the per-modem state at the ISP is minimal: just the last time when there was activity, and a per-modem timeout if that is configurable for each modem. Using an adaptive threshold, one needs the per-modem timeout, as well as a global or per-modem set of parameters for adjusting the timeout. One also must note not only the last time of activity but also the time when a modem disconnected, in order to determine whether a bump has occurred. Predictive timeouts, which use a buffer of historical information, require several idle time and active intervals for each modem. In no case is the per-modem state more than a few tens or hundreds of bytes per modem, however.

5 Experiments

The fixed-threshold policies used disconnect thresholds of 2, 5, 10, and 15 minutes. When reported individually, these are designated by *fixed*N, for a value of N. In our graphs, the fixed policies are generally connected by a dashed line to highlight them by comparison to the adaptive policies, and because one can usually interpolate between two fixed thresholds. The adaptive policies used additive or multiplicative modifiers within ranges that approximated the fixed-threshold policies. When clustered, these are grouped by the type of modified and the range within which they vary. When shown separately, they are designated by a string of the form $\mathbf{S}$-β_a+$\alpha_a\mathbf{m}min\mathbf{M}max$ for additive modifiers, or $\mathbf{S}/\beta_m{*}\alpha_m\mathbf{m}min\mathbf{M}max$ for multiplicative ones. S is a starting threshold in minutes; all experiments reported here started with a 5-minute threshold.

The values for α and β appear in Table 1(a), and the ranges among which they varied appear in Table 1(b). Note that the values for β_m in the table refer to an amount by which to divide the current threshold.

In addition to varying the modifiers and ranges, we varied the definition of a bump and the inclusion or exclusion of "workaholics," as described next.

5.1 Workaholics

Some clients are essentially always active. This may be because they are actively using the network, getting useful work done. In other cases the communication is "busy work" that is specifically intended to keep the connection alive. Either way, no method that is intended to modify the disconnection threshold is going to affect these clients. The more "workaholics" there are, the less

Additive		Multiplicative	
α_a	β_a	α_m	β_m
5.00	-1.00	1.2	1.1
5.00	-2.00	1.4	1.1
3.00	-2.00	1.4	1.2

(a) Adjustment values. The left two columns list the additive values studied in this paper, while the right two list the multiplicative values. Additive times are in minutes.

Starting value	Minimum value $T_{\min}$	Maximum value $T_{\max}$
5	1	15
5	5	15
5	5	30

(b) Ranges of the disconnect threshold studied in this paper, in minutes.

Table 1: Parameters for adaptive spin-down (times in seconds). The cross-product of the sets of parameters was used to drive the simulations; that is, each of the 3 combinations of (α_a, β_a) and 3 combinations of (α_m, β_m) in Table (a) is used with each of the 3 sets of values in Table (b), giving 18 sets of adaptive parameters to drive the simulator.

overall benefit might accrue from modifying the behavior of the non-workaholics. (Barbará and Imieliński [1] refer to the latter as "sleepers.")

We identify workaholics by noting those modems that accumulated a total connection time of at least 80% of the entire trace, using the off-line optimal disconnection threshold. The simulator reports statistics across all modems and also excluding those modems that are identified during the off-line optimal run as workaholics. The number of workaholics is partially dependent on the definition of a bump; in our trace, with a 5-minute bump threshold, 15 hosts were so identified, and the number increased to 19 with a 10-minute threshold.

How should workaholics be treated? A workaholic that would be active even if modem disconnections were unintrusive should be considered in all results, because it will limit the overall benefit available if new policies were deployed. One that is that artificially busy *because* of the inconvenience of modem disconnections should be factored out. Our experience in discussing modem activity with their owners has suggested that the latter case is far more common, and we therefore exclude workaholics from consideration in this paper unless stated otherwise. We consider workaholics further in Section 6.4.

6 Results

We present results that compare adaptive and fixed thresholds, given a fixed definition of what constitutes a bump and aggregating across all users. We then compare these results to a set of

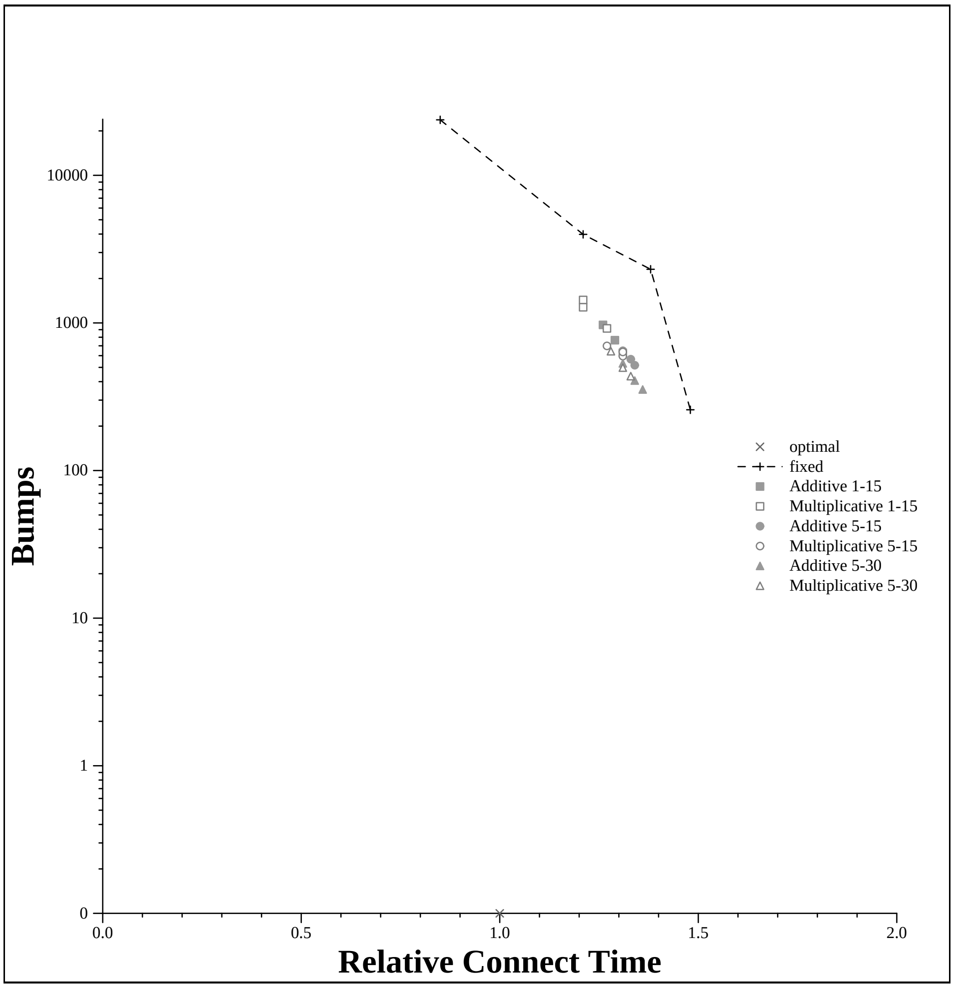

Figure 2: Total bump count, 5-minute bump threshold, excluding 15 workaholics.

simulations with a more conservative definition of a bump. Next we consider the effect of increasing the maximum number of prediction errors. We then include the effect of workaholics, and finally we look at variations of the adaptive thresholds among two individual users.

6.1 Adaptive versus Fixed Thresholds

From the standpoint of user annoyance, the number of times the user must wait for a new connection is of great importance. However, the longer since the user last communicated, the more willing the user is to wait to reinitiate contact. Different users may have varying levels of willingness to suffer an initial delay. As a baseline, we consider a case where a disconnect is considered a bump if the modem isn't idle for at least 5 minutes after the disconnection. We vary this threshold in the next subsection.

The first set of figures shows the cumulative effect across the entire user population of approximately 100 modems, but excluding the activity of 15 workaholics. Figure 2 shows the total number of bumps encountered, on a log scale, by comparison to the total connect time. Connect time is itself relative to the off-line optimal case, as discussed above, which appears on the bottom axis at the relative value of 1. The fixed 2-minute timeout uses slightly less connect time but encounters an unacceptable total of over 20,000 bumps over a one-week period. The other fixed points encounter far fewer bumps, but only the 15-minute threshold encounters fewer bumps than the various adaptive algorithms. Compared to the 5-minute threshold, some of the adaptive algorithms encounter close to half an order of magnitude fewer bumps at no cost in connect time, while the rest reduce the bumps further in exchange for more connect time. Compared to the 10-minute threshold, virtually all the adaptive points are both below and to the left, i.e., encounter both fewer bumps and less connect time.

The fixed 15-minute threshold is an interesting case. Compared to the adaptive point that is farthest to the bottom-right of the graph, it uses 9% more connect time and encounters 27% fewer bumps. The relative weights of the need to reclaim modems and the desire not to inconvenience the user would determine whether the simpler fixed threshold would be more desirable.

Focusing on the differences between additive and multiplicative thresholds, we see that the additive policies within a range of values are consistently below and to the right of the multiplicative ones. (Refer to the solid polygons of a particular shape, comapred to the open ones.) This means that the additive policies are connected longer but encounter fewer bumps. This is unsurprising since the multiplicative ones react to bumps more aggressively; the same phenomenon was noted in the domain of spinning down a disk [2].

"Bump severity" weights the bumps by the extent to which they missed the threshold. Figure 3 shows the severity-versus-connection plot comparable to Figure 2. Most of the same comments apply, but the adaptive points with the lowest severity are closer to the 15-minute fixed threshold. Here, the fixed threshold reduces the severity by just 10% in exchange for the same 9% increase in connect time.

Figure 4 plots the average number of modems in use at each 30-second collection point, relative to the optimal, against the same "relative connect time" in the preceding figures. As one would expect, the more total time used by an algorithm, the more likely additional modems will be

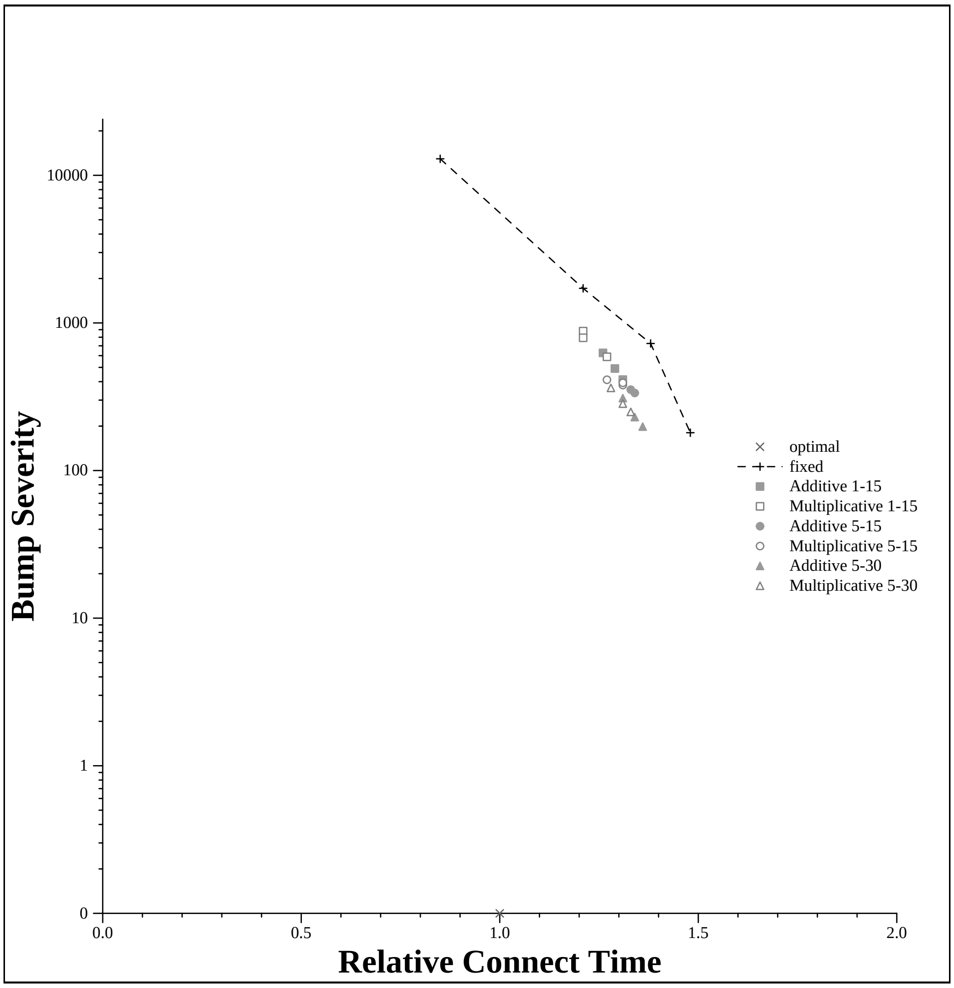

Figure 3: Bump severity, 5-minute bump threshold, excluding 15 workaholics.

in use in parallel. The 15-minute fixed threshold uses 23% more modems on average than the optimal, whereas the adaptive algorithm with the fewest bumps uses just 13% more, equivalent to a reduction of 8% from the 15-minute threshold. The 2-minute threshold uses much less than the off-line optimal, but of course it encounters too many bumps to be of practical interest.

Figure 5 is similar to Figure 4, but with the maximum number of simultaneous modems instead of the average. With this trace and configurations, virtually all the thresholds except the fixed 2-minute one hit the same maximum. The 2-minute threshold had a lower maximum, as one would expect. The 15-minute and a couple of the adaptive thresholds, including the one that is closest to the 15-minute threshold in bumps, had a maximum that was one modem greater than the other thresholds.

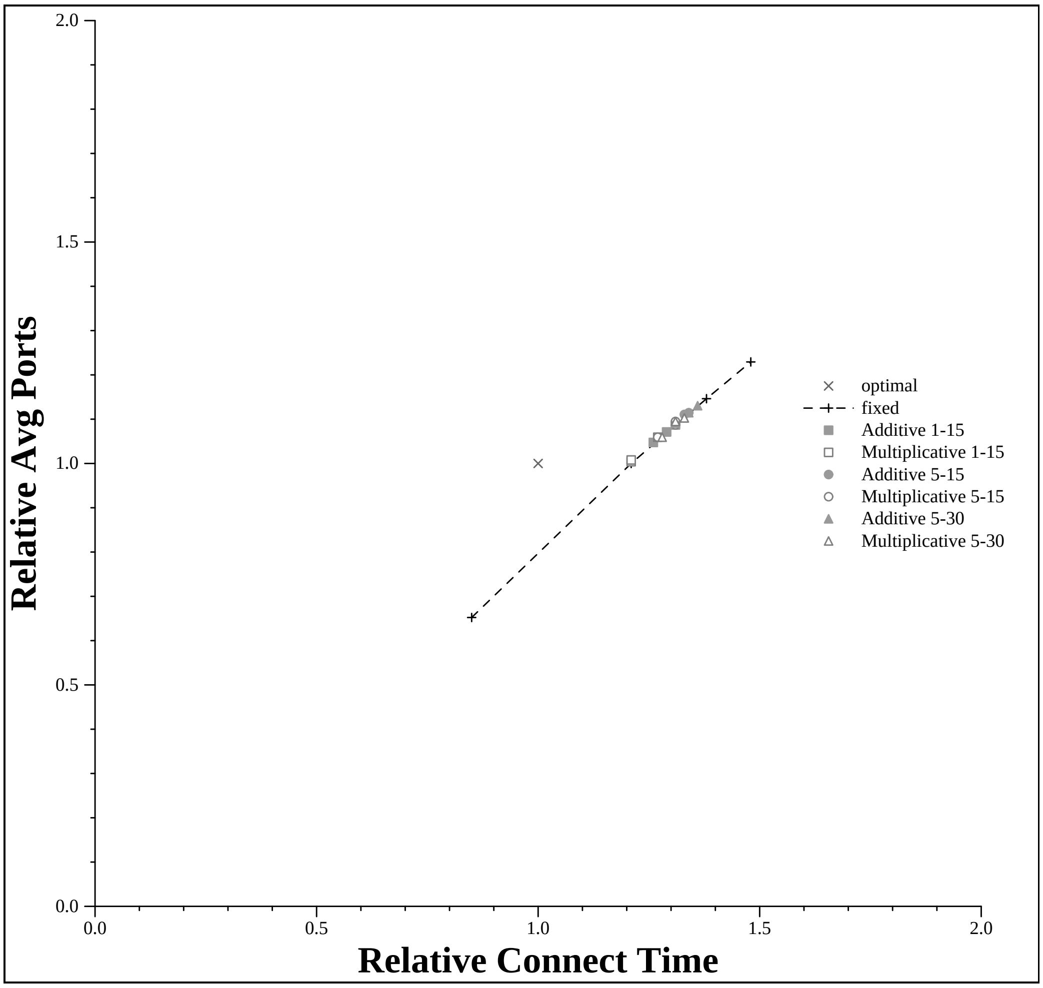

Figure 4: Mean modem usage, 5-minute bump threshold, excluding 15 workaholics.

6.2 Varying the Bump Threshold

Figures 6–8 present simulation results corresponding to Figures 2–4, but with a bump if the idle period is less than 10 minutes (rather than 5).[1] Note that although most numbers are directly comparable because they are relative to the optimal policy for that definition of a bump, the total counts of bumps are absolute counts. Since they exclude different numbers of workaholics, they are not directly comparable, and one should instead consider the trends within each graph.

Focusing just on the number of bumps (Figure 6, compared with Figure 2), the overall shape of the fixed-threshold curves and the relative positions of the adaptive-threshold points are similar. There are some notable distinctions, though:

[1]The figure corresponding to Figure 5 is omitted due to space limitations, but is virtually the same.

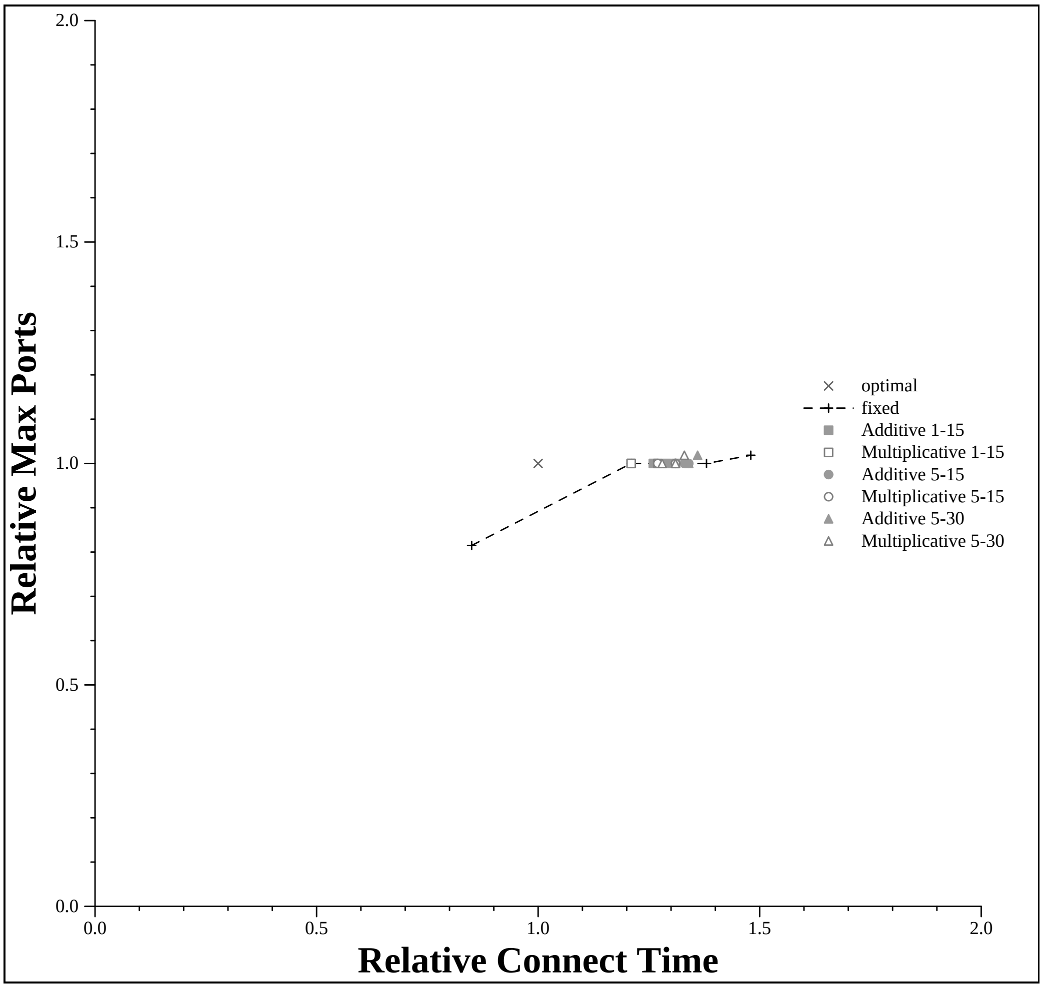

Figure 5: Maximum modem usage, 5-minute bump threshold, excluding 15 workaholics.

- The 2-minute fixed timeout improves on the total connect time, relative to the 5-minute bump threshold. This is because the optimal in the case of a 10-minute bump threshold disconnects significantly less often. The total bump count decreases, but only because of the four additional workaholics that are excluded in the 10-minute bump case. Counting all modems, the number of bumps increases by 11%, which is to be expected given the more stringent threshold for a bump yet a consistently low disconnect threshold.

- While in both figures the adaptive points are consistently below the line formed by the fixed-threshold points, the best adaptive point in the 10-minute bump case is closer to the 15-minute fixed threshold case than for the 5-minute bump. The adaptive policy with the fewest bumps increases the total bump count by less than 4% compared to the 15-minute fixed threshold,

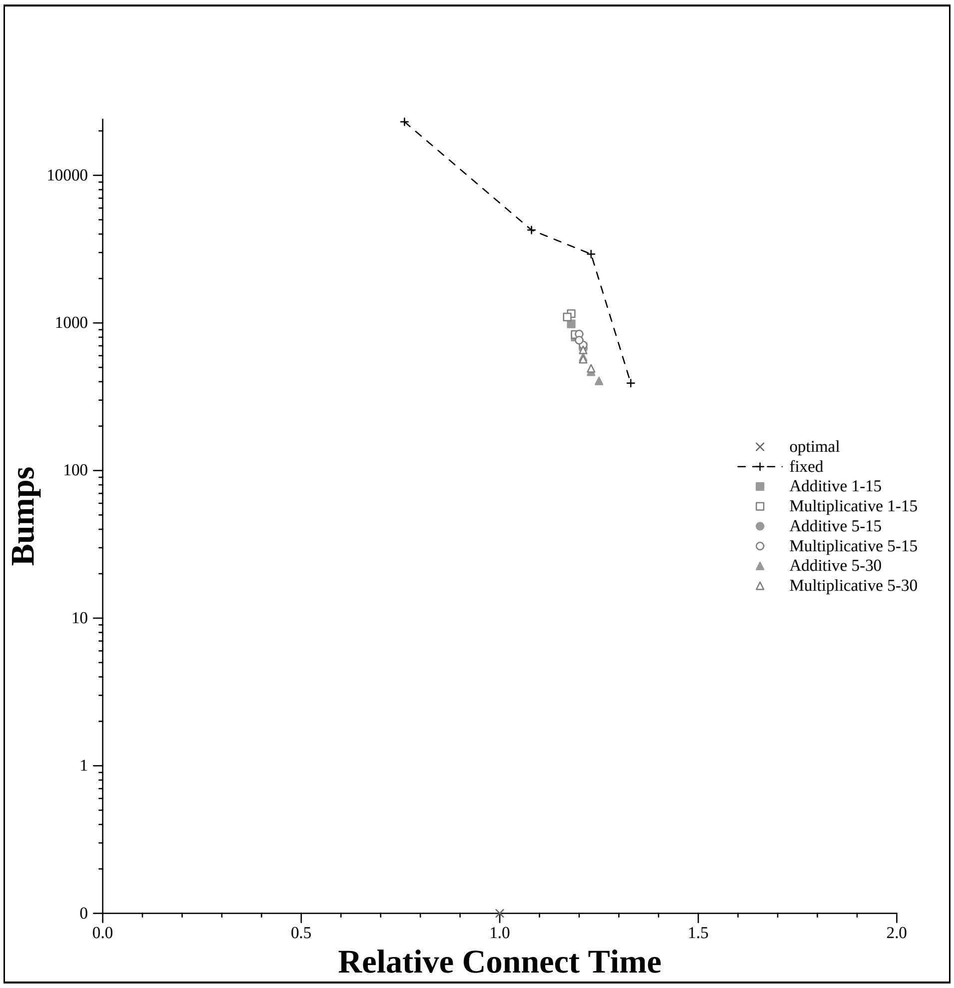

Figure 6: Total bump count, 10-minute bump threshold, excluding 19 workaholics.

and reduces connect time by 6%. When bump severity is considered, the adaptive policy actually reduces overall severity by 4% compared to the 15-minute threshold.

The severity for the fixed 5-minute (Figure 3) and 10-minute (Figure 7) thresholds is virtually identical: it actually increases by about 1% with the higher threshold, while simultaneously increasing overall connect time by 14%. This is because waiting 10 minutes and then disconnecting is more likely to hit network activity soon thereafter than waiting 5 minutes. For instance, a user who checked mail every 12 minutes would encounter a bump with a fixed threshold of either 5 or 10 minutes and a need to be idle for another 10 minutes, but the reconnect would occur earlier in the cycle for the 10-minute disconnect threshold than for the 5-minute threshold, resulting in a higher weighted severity.

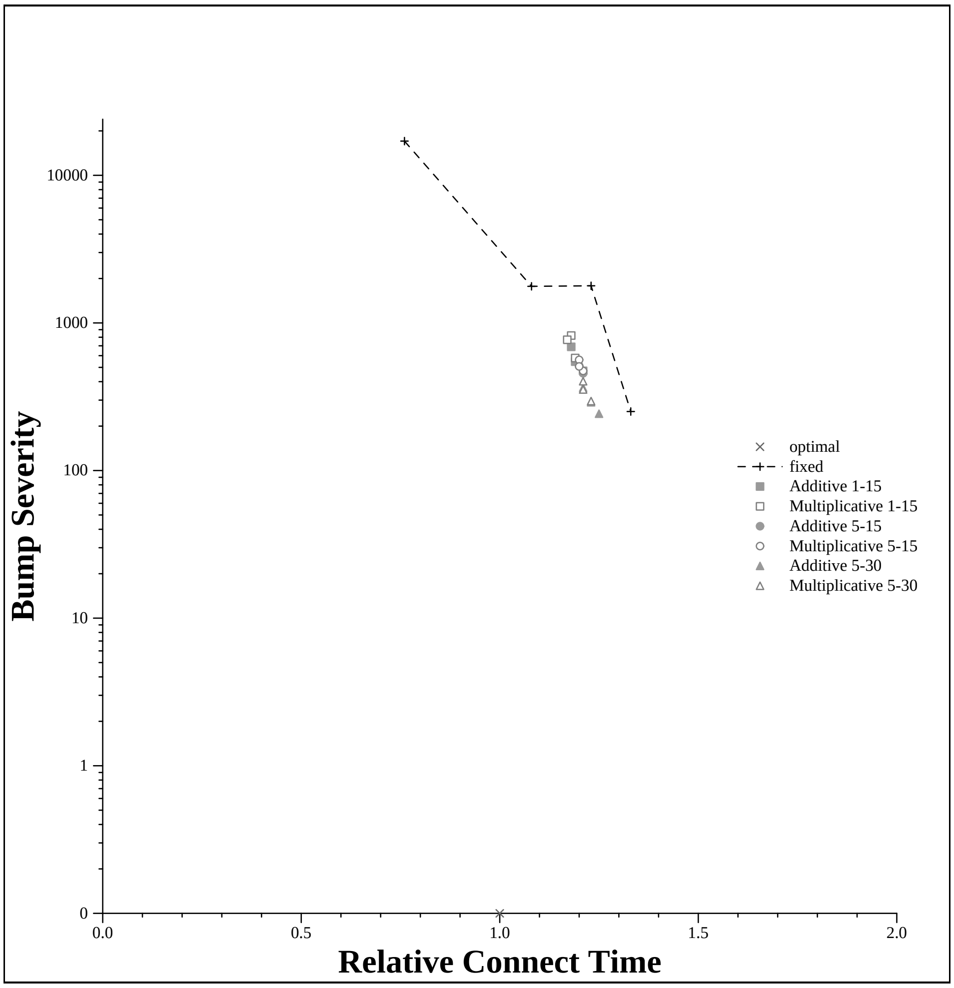

Figure 7: Bump severity, 10-minute bump threshold, excluding 19 workaholics.

6.3 Prediction Errors

When our system predicts a timeout based on repeated intervals of similar activity, it disconnects the modem quickly by comparison to the normal disconnection threshold, which varies using a "random walk" approach. As a result, if the prediction is in error, the user is likely to be much more inconvenienced than with the adaptive threshold. Currently, the system counts the number of such errors for each modem and stops making predictions for a modem that has already encountered two such errors. We investigated the sensitivity to this threshold by increasing it from 2 to 5. As one might expect, the effect was to increase the bump count while decreasing total connect time. All such changes were moderate, and not readily discernible in a graph.

6.4 Workaholics

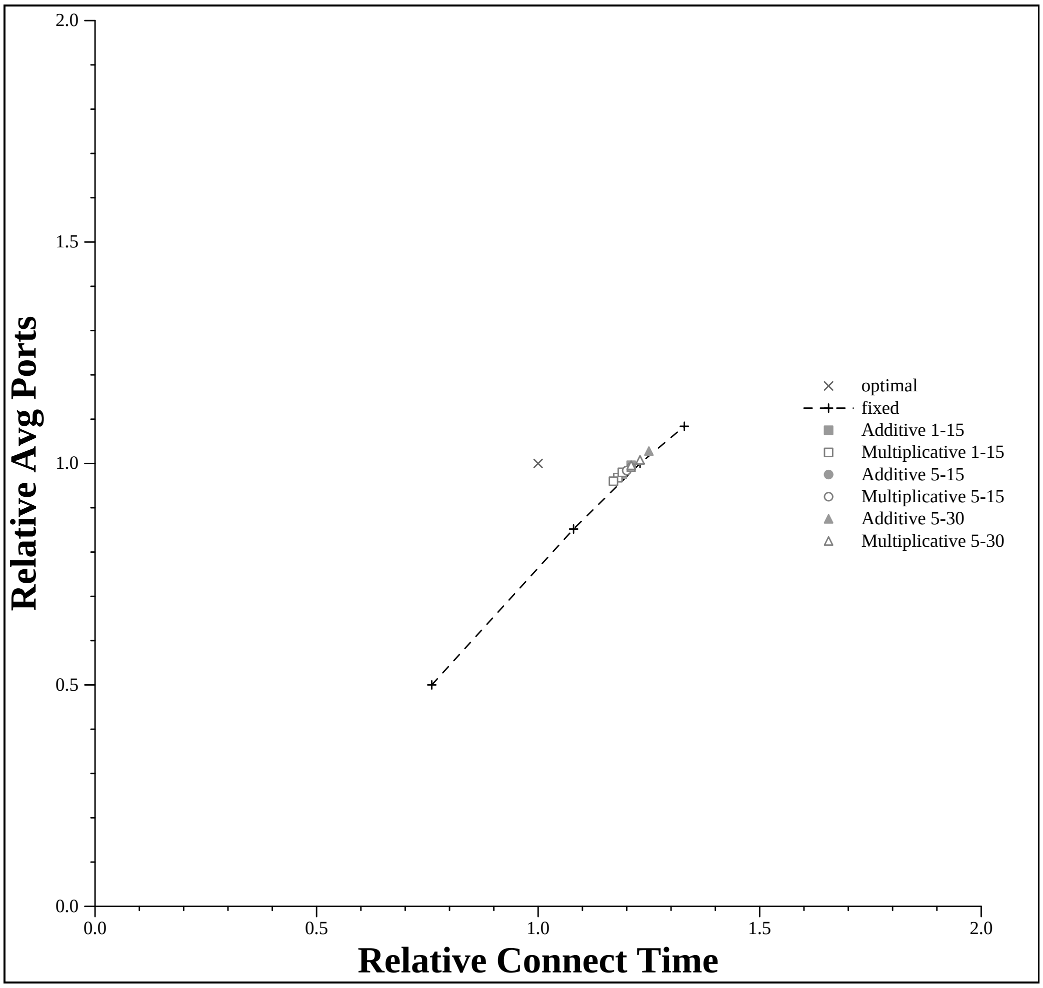

Figure 8: Mean modem usage, 10-minute bump threshold, excluding 19 workaholics.

The effect of including workaholics in the simulation results is to compress the "relative connect time" and "relative ... ports" numbers. This is because in each case, all numbers get shifted by a constant amount (such as about 7 days of connect time, multiplied by the number of workaholics), and then divided against the higher value for the off-line optimal. Figure 9 gives an example of the total bump count, including workaholics, corresponding to Figure 2. Note that the total number of bumps will remain constant in almost all cases, since workaholics will not be disconnected except with a very short fixed threshold.

6.5 Threshold Variation

Figures 10 and 11 plot the variation in threshold for the different algorithms for two clients. The modem in Figure 10 belonged to one of the authors and showed a marked fluctuation over

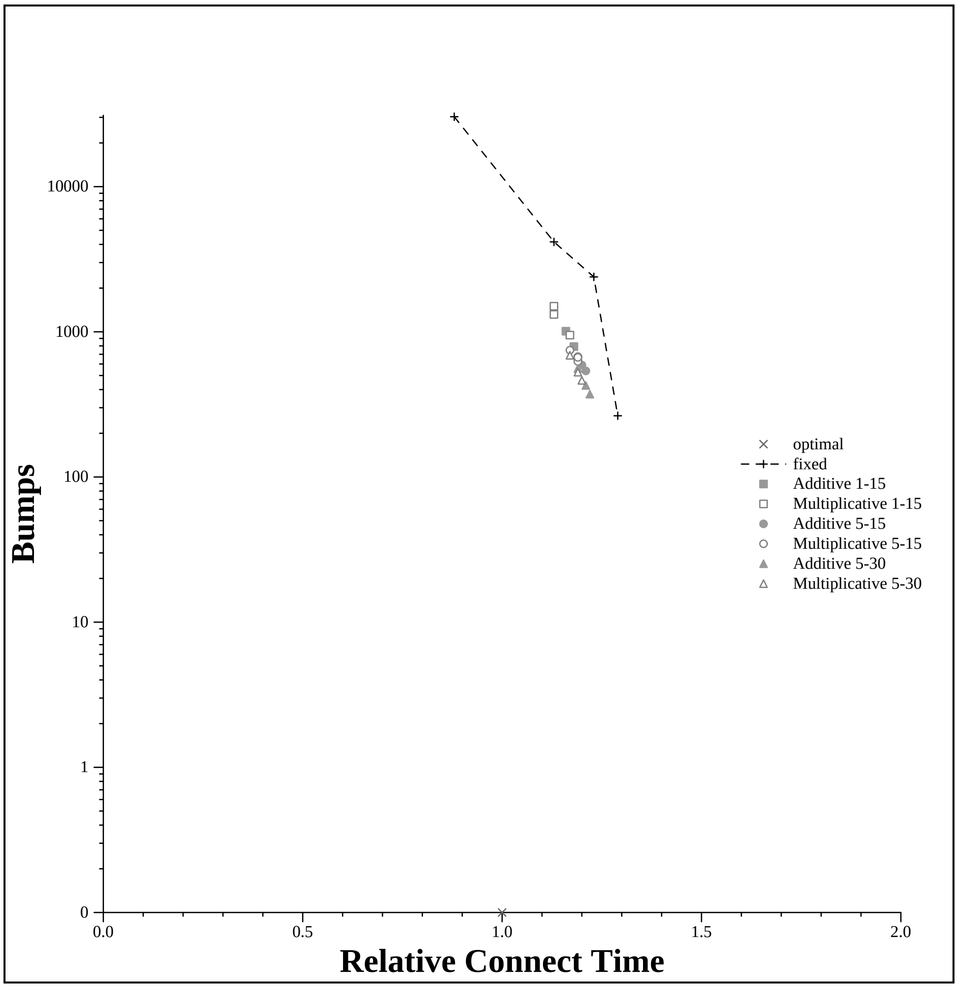

Figure 9: Total bump count, 5-minute bump threshold, including the 15 workaholics.

time. The modem in Figure 11 was one of the workaholics, accumulating 6.8 days' worth of connect time over the 7-day trace interval using a 15-minute threshold. As a consequence, the adaptive thresholds vary initially and eventually settle into a consistent state as a function of the rate at which they adjust and the range with which they can vary. The consistent state indicates that no further disconnects occur (otherwise the thresholds would continue to adjust).

7 Summary and Future Work

Adaptive techniques that use past history to predict a good timeout interval for modem disconnection can reduce overall resource consumption with little or no additional inconvenience to the user. These techniques require minimal state on the part of the ISP and are easy to implement.

The exact choice of which modifiers to use, and limitations to place on the range among which

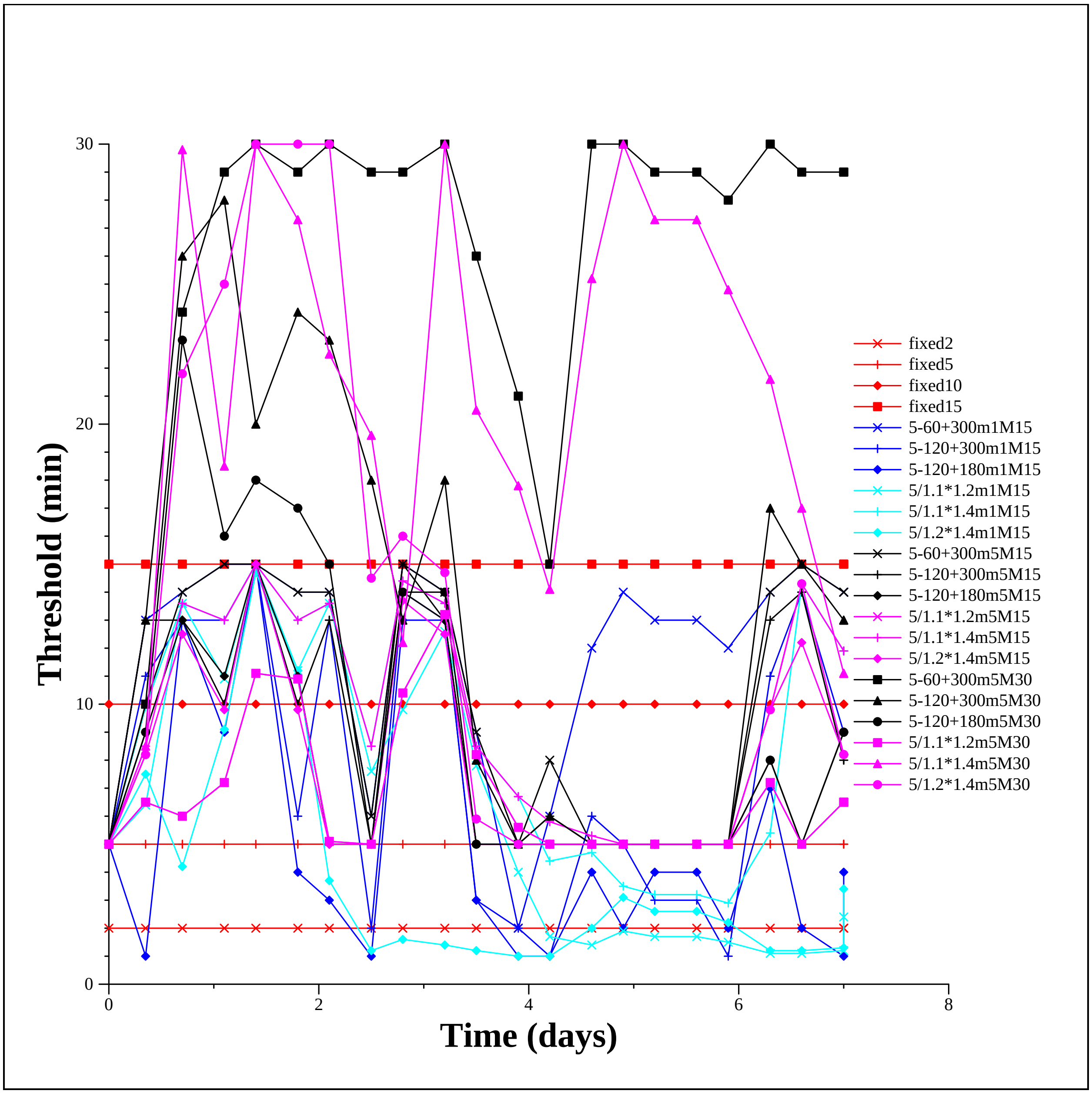

Figure 10: Variation in thresholds for a user with irregular accesses.

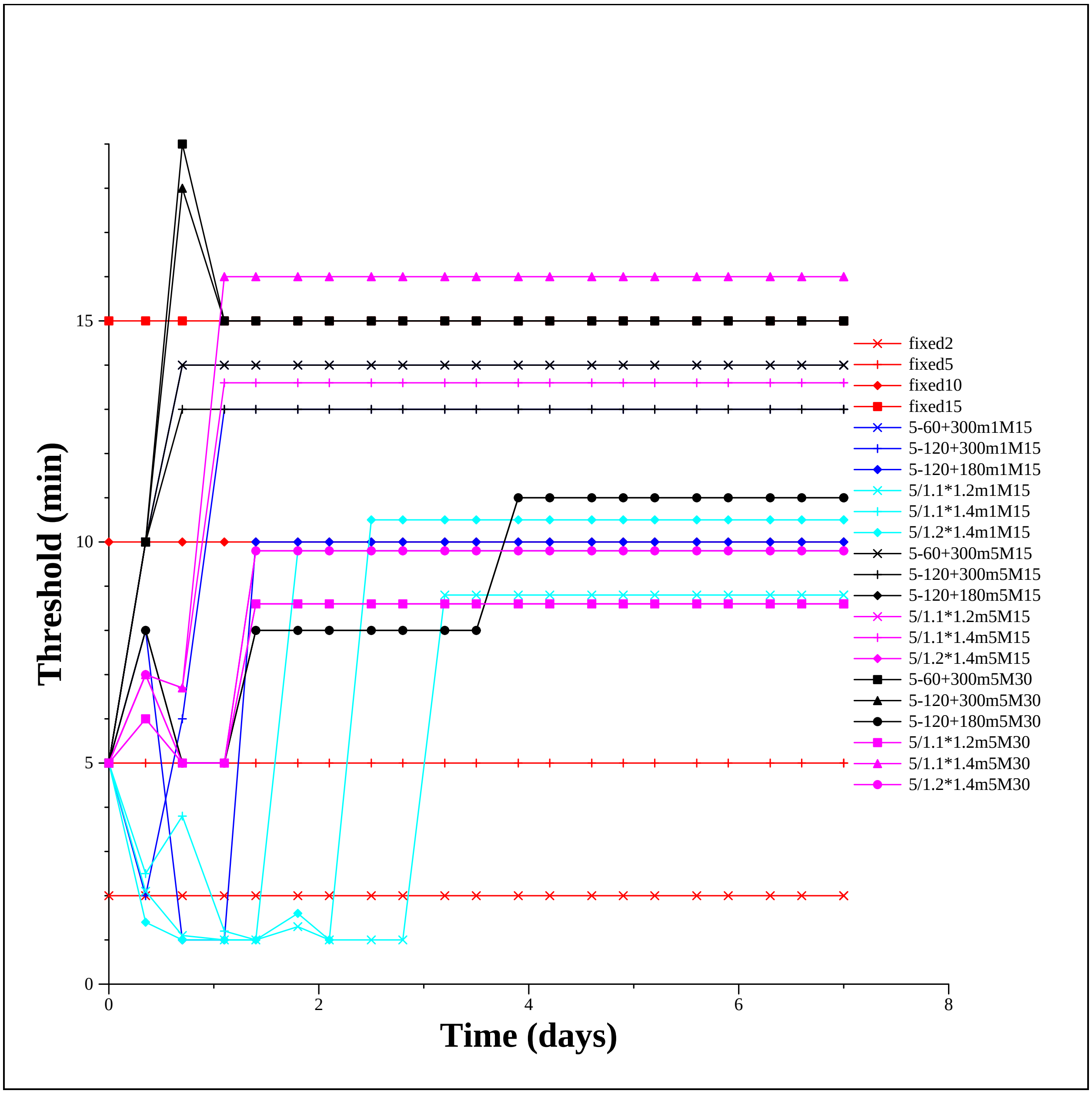

Figure 11: Variation in thresholds for a user with regular and frequent accesses (a "workaholic."

the timeout can vary, is thus far an imprecise art. Additional experimentation with "live users" will be helpful in further evaluating the technique and providing guidelines for these parameters.

References

[1] Daniel Barbará and Tomasz Imieliński. Sleepers and workaholics: Caching strategies in mobile environments. In Richard T. Snodgrass and Marianne Winslett, editors, *Proceedings of the International Conference on Management of Data*, pages 1–12, New York, NY, USA, May 1994. ACM Press.

[2] Fred Douglis, P. Krishnan, and Brian Bershad. Adaptive disk spin-down policies for mobile computers. *Computing Systems*, 8(4):381–413, Fall 1995. An earlier version appeared in *Proceedings of the Second Symposium on Mobile and Location-independent Computing*, pp. 121–137, April 1995.

[3] Fred Douglis, P. Krishnan, and Brian Marsh. Thwarting the Power Hungry Disk. In *Proceedings of 1994 Winter USENIX Conference*, pages 293–306, San Francisco, CA, January 1994.

[4] Anna R. Karlin, Kai Li, Mark S. Manasse, and Susan Owicki. Empirical studies of competitive spinning for a shared-memory multiprocessor. In *Proceedings of 13th ACM Symposium on Operating Systems Principles*, pages 41–55. Association for Computing Machinery SIGOPS, October 1991.

[5] S. Keshav, C. Lund, S. J. Phillips, N. Reingold, and H. Suran. An empirical evaluation of virtual circuit holding time policies in IP-over-ATM networks. *IEEE Journal on Selected Areas in Communications*, 13(8):1371–1382, October 1995.

[6] Joseph R. Kiniry and Christopher Metz. Cable modems: Cable TV delivers the internet. *IEEE Internet Computing*, 2(3):12–15, May–June 1998.

[7] Kester Li, Roger Kumpf, Paul Horton, and Thomas Anderson. A Quantitative Analysis of Disk Drive Power Management in Portable Computers. In *Proceedings of the 1994 Winter USENIX*, pages 279–291, San Francisco, CA, 1994.

[8] William Simpson et al. RFC 1661: The point-to-point protocol (PPP), July 1994.

[9] Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. Scheduling for reduced CPU energy. In *Proceedings of the First Symposium on Operating Systems Design and Implementation (OSDI)*, pages 13–23. USENIX Association, November 1994.