ARTICLE IN PRESS

Computer Networks xxx (2008) xxx–xxx

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Equilibrium analysis through separation of user and network behavior

Y.C. Tay [a,*], Dinh Nguyen Tran [b], Eric Yi Liu [c], Wei Tsang Ooi [d], Robert Morris [e]

[a] National University of Singapore, Mathematics Department, 2 Science Drive 2, Kent Ridge 117543, Singapore
[b] New York University, Department of Computer Science, Courant Institute of Mathematics, 715 Broadway Room, New York, NY 10003, USA
[c] University of North Carolina, Department of Computer Science, Sitterson Hall, Chapel Hill, N.C. 27599-3175, USA
[d] National University of Singapore, Department of Computer Science, Law Link, Singapore 117590, Singapore
[e] Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory, 32 Vassar Street, Cambridge, MA 02139, USA

ARTICLE INFO

Article history:
Received 23 March 2008
Received in revised form 5 August 2008
Accepted 9 September 2008
Available online xxxx

Responsible Editor: A. Abouzeid

Keywords:
User–TCP interaction
User backoff
Aborted downloads
Internet robustness
Bandwidth provisioning

ABSTRACT

Internet complexity makes reasoning about traffic equilibrium difficult, partly because users react to congestion. This difficulty calls for an analytic technique that is simple, yet have enough details to capture user behavior and flexibly address a broad range of issues.

This paper presents such a technique. It treats traffic equilibrium as a balance between an inflow controlled by users, and an outflow controlled by the network (link capacity, congestion avoidance, etc.). This decomposition is demonstrated with a surfing session model, and validated with a traffic trace and NS2 simulations.

The technique's accessibility and breadth are illustrated through an analysis of several issues concerning the location, stability, robustness and dynamics of traffic equilibrium.

1. Introduction

Given an Internet usage pattern (web surfing, video streaming, etc.), the network protocols would adjust to that pattern (packet routing, congestion control, etc.) and eventually settle into an equilibrium state of traffic flow. When there is a change in the usage (e.g. flash crowd) or network (e.g. router misconfiguration), this equilibrium would shift. As the Internet grows larger and more complex, researchers and engineers need some technique to help them cut through the complexity, so they can reason intuitively about shifts in traffic equilibrium.

This need is particularly acute if the usage pattern interacts with network performance. For example, although TCP exercises congestion control, traffic volume is not determined by TCP alone—while TCP controls the flow in a connection, the number of connections is controlled by users. The following loop illustrates the interaction between users and TCP:

User–TCP feedback cycle

(i) Congestion causes TCP to reduce bandwidth per connection.
(ii) The reduced bandwidth and increased delay causes users to launch fewer connections, or abandon them.
(iii) The decrease in number of active connections allows TCP to increase flow per connection.
(iv) The flow increase encourages users to launch more connections, causing congestion to increase and returning to (i).

It is not clear where (i.e. what equilibrium state) this cycle converges to.

Besides making it hard to reason about traffic equilibrium, Internet complexity also broadens the range of issues. We illustrate this breadth by posing some questions:

* Corresponding author. Tel.: +65 68742949.
E-mail address: dcstayyc@nus.edu.sg (Y.C. Tay).

1389-1286/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2008.09.008

The advent of file-sharing systems has resulted in much bandwidth being consumed by transfers of large (e.g. MP3) files [37].

(**Q1**) Can elephantine transfers cause a collapse of Internet performance?

Another cause for worry is the growing use of UDP, which does not exercise congestion control. For example, Internet telephony is a UDP application that has become hugely popular. Whereas a user who initiates a large file transfer is likely to ignore the transfer till after it completes, a voice-over-IP (VoIP) user is actively engaged in a telephone connection. The latter implies that one must consider how users behave dynamically when analyzing traffic equilibrium [7]:

(**Q2**) Does user behavior help or hurt Internet performance?

This issue remains even if all UDP traffic is TCP-friendly [16], since we still need to resolve the user–TCP feedback cycle. The cycle also makes the answer to the following not obvious:

(**Q3**) Would an increase in bottleneck bandwidth relieve congestion?

For example, Yang and de Veciana reckoned that overprovisioned bandwidth will be quickly consumed, possibly by unwanted traffic [42].In any case, overprovisioning cannot prevent overloading (even in the Internet backbone [22]), since failure in one link can strain another link with redirected traffic.

(**Q4**) How does traffic reach a new equilibrium when load changes?

Of course, the answer partly depends on TCP's congestion avoidance mechanism, but flow volume is not completely controlled by TCP, so:

(**Q5**) What role does TCP play in determining traffic equilibrium?

1.1. Congestion-induced user behavior in equilibrium analysis

Since traffic volume is partly controlled by users, one must consider their reaction to congestion when addressing questions like Q1–Q5. As Prasad and Dovrolis observed: TCP's congestion control and capacity overprovisioning are not the only reasons for the rare occurrence of significant congestion events in the Internet today [35].

Most Internet traffic is carried by TCP, so we focus attention on the dominant TCP application—namely, the web. A web surfer reacts to congestion in two ways:

$U^{\text{abort}}_{\text{backoff}}$: she may abort a slow download by clicking "Stop", "Reload" or another hyperlink, and
$U^{\text{session}}_{\text{backoff}}$: she may cut short her surfing session.

Bonald and Roberts point out such user reaction [6], and we found evidence for $U^{\text{abort}}_{\text{backoff}}$ and $U^{\text{session}}_{\text{backoff}}$ in traffic traces: Fig. 1 shows that, as bandwidth per download decreases, the probability that the download is aborted increases; Fig. 2 shows that, as bandwidth in a session decreases, the number of completed downloads in a session decreases. One can view such behavior as a form of *congestion-induced user backoff*.

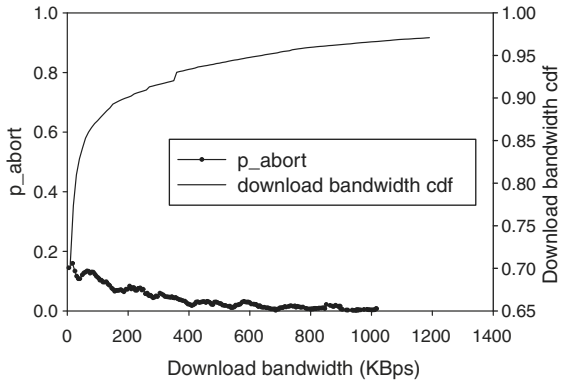


Fig. 1. Evidence for $U^{\text{abort}}_{\text{backoff}}$: as download bandwidth decreases, the abort probability increases. The cumulative distribution function (cdf) is represented by the smoother curve.

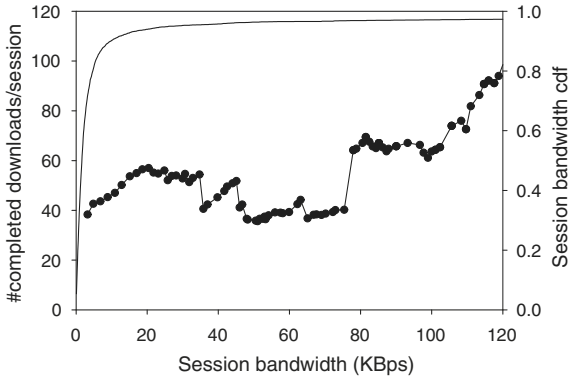


Fig. 2. Evidence for $U^{\text{session}}_{\text{backoff}}$: as session bandwidth decreases, number of completed downloads per session decreases; here, we focus on session bandwidths of 20 KBytes/s or less— the cdf indicates that they make up 95% of the data.

Hence, the objective of this paper is *to present an intuitive and flexible technique for analyzing traffic equilibrium—as an interaction between user and network behavior—that can be applied to a broad range of issues.*

Specifically, we relate the user–network decomposition to the TCP instability region observed by Garetto et al. [18], the asymmetry in congestion buildup and decay observed by Iyer et al. [22], and the primal–dual formulation by Kelly et al. [24,28]. We also use it to identify the different contributions made by users and TCP to robustness of the equilibrium, examine how user reaction affects bandwidth provisioning, formulate a rule for anticipating overload, and formalize the deleterious effect of elephantine downloads on performance.

1.2. Current models and techniques

Our starting point is a model for web surfing. Although surfer reaction to delays was mentioned by several authors [4,14,3], it is not modeled by most studies of web traffic [9,12,20,29]. For example, Key et al. study the benefit to "far-sighted" users if they defer their bandwidth demand

when the network is in a congested phase, but this postponed demand is not reduced, and there are no aborts [25].

There are studies of user-initiated TCP connection abortion [17,6,36], but a click may generate more than one TCP connection. Also, there are several surfer impatience models for abort ($U^{\text{abort}}_{\text{backoff}}$) [10,41], but none for sessions ($U^{\text{session}}_{\text{backoff}}$). Choi and Limb's behavioral model [12] and Barford and Crovella's user equivalent [5] also do not include sessions, whereas the layered model by Hlavacs and Kotsis does not provide for user reaction to delays [20].

Similarly, the various measurement studies to characterize web traffic [9,27,29] implicitly assume user behavior is invariant under congestion. Vicari and Köhler measure how modem users' behavior depends on access speed [40], but this effect is not congestion-induced. Studies on how users react to server delays [14] and models of shopping behavior at e-commerce sites [31] are only marginally relevant since, in our context, we are considering multiple surfers—they may visit different websites, and each user may visit multiple websites.

The surfing session model we present here (Figs. 3 and 19) is therefore the first to study both $U^{\text{abort}}_{\text{backoff}}$ and $U^{\text{session}}_{\text{backoff}}$.

Recall that we focus on surfing only as a setting for studying traffic equilibrium. There is a huge literature on equilibrium analysis for classical telephone networks, including papers on repeated call attempts under congestion [11]. However, there are important differences: circuit-switched calls have dedicated bandwidth, a "session" rarely has more than one call, and an aborted call does not consume bandwidth. (We will see in Section 6 how bandwidth wastage from abortion can cause a performance collapse.)

The literature on Internet research has various techniques for equilibrium analysis. Two recently popular ones use game theory [2] and mathematical programming [24], but these are techniques unsuitable for intuitive, back-of-an-envelope arguments. In performance modeling, stochastic analysis is a well-established technique but, for tractability, one has to choose the right level of detail; e.g. Gromoll et al. point out that the literature has few results on processor-sharing queues (modeling bandwidth sharing) with impatient customers [19].

Since our objective is to offer an abstraction for researchers and engineers to reason intuitively about traffic equilibrium, the model must not be intricate, yet have enough details to address questions like Q1–Q5. For this, we adapt an old fluid approximation by Kleinrock and Lam [26] that view an equilibrium as a balance between input and output flows. The technique also provides closed-form expressions that facilitate the analysis.

We will describe other related work as the need arises.

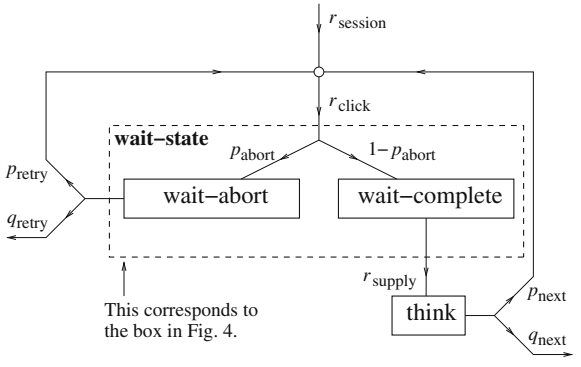


Fig. 3. Surfing session model: p_{retry} is the proportion of aborted downloads that are followed by another click in the session, and p_{next} is the proportion of completed downloads that are followed by another click in the session.

1.3. Our contribution

This paper's contributions are:

(i) *The first model to study both* $U^{\text{abort}}_{\text{backoff}}$ *and* $U^{\text{session}}_{\text{backoff}}$ *in user backoff*. This *surfing session model* can be used to generate realistic surfing traffic in network simulation, compare web traffic traces (Section 4), and help Internet service providers anticipate overload (Section 6.9).

(ii) *An intuitive technique for analyzing traffic equilibrium*. This technique is based on a *user–network* model that separates user and network behavior into two curves, and the traffic equilibrium is where these two curves intersect (Fig. 13). When there is a change in network condition, its impact on the equilibrium can be analyzed by considering how that change affects each curve *individually*. This offers a mechanism for breaking the user–TCP feedback cycle when reasoning about the equilibrium, and can be used to address questions like those above (Q1: Section 7.1; Q2: Section 6.5; Q3: Section 6.8; Q4: Section 6.4; Q5: Section 6.6).

1.4. Overview of paper

We begin by introducing the surfing session model in Section 2. Section 3 then presents the user–network model, consisting of two equations that encapsulate user and network behavior. In the literature, "congestion" may refer to phenomena that occur at different timescales—from micro-granularity (e.g. sub-second packet bursts) to medium-granularity (e.g. multi-chunk peer-to-peer downloads) to macro-granularity (e.g. daily peak). We will see how this range is reflected in the equations. We also derive an equation that characterizes the fixed point of the user–TCP feedback cycle (Section 1).

In Section 4, we validate the models by extracting from a `tcpdump` the surfing session model's parameters, as well as the user and network curves.

Trace analysis is restricted by the corresponding network's hardware and software configuration. We work around this restriction by doing a simulation study. Section 5 presents user and network curves obtained with `NS2` simulations [30] and fits the network curve with Padhye et al.'s TCP equation [33] (referred to as *PFTK equation* below). We also examine how congestion control affects the network curve.

Please cite this article in press as: Y.C. Tay et al., Equilibrium analysis through separation of user and network behavior, Comput.Netw. (2008), doi:10.1016/j.comnet.2008.09.008

Section 6 demonstrates how the user–network model can be used to analyze intuitively a broad range of issues: the location and dynamics of the equilibrium, the contribution of users and TCP to Internet robustness, how user backoff affects bandwidth provisioning, and how overload can be anticipated by monitoring some inequalities.

Section 7 further illustrates our models' flexibility by presenting a two-class model to study the effect of non-reactive elephantine file transfers, and a closed model for Internet service providers. Section 8 then concludes with a summary.

2. Surfing session model

For now, we assume the user is a web surfer who reacts to congestion, and focus on HTTP flows and user backoff. (Section 7.1 considers UDP flows and non-reactive users.) The model groups HTTP requests into *sessions*. The requests in a session may span multiple sites.

We first adopt an open model, i.e. the *session arrival rate* r_{session} is constant, regardless of congestion. Users are unaware of the congestion level until they arrive, so this assumption is reasonable. Nonetheless, we will remove this assumption in Section 7.2.

It is well-known that traffic has 24-h and weekly cycles, so it is not constant. Our assumption of constant r_{session} therefore restricts the open model to a time span of, say, minutes. This is consistent with our focus here on user behavior, which is manifested over a medium-granularity timescale. Variation in r_{session} is, in our model, over a macro-granularity timescale. We revisit this point later (Sections 4.6 and 6.4).

In each session, a user sends HTTP requests with *clicks* on bookmarks, hyperlinks, submit buttons, etc. For convenience, we consider typing in an URL as a click too. Let r_{click} be the *click rate*.

In general, each click generates multiple (serial or parallel) HTTP request-responses; the traffic they send to the user is called a *download* (equivalently, *Web object* [5] or *Web-request* [12]). Note that one user may launch parallel downloads with multiple browsers.

After a click, a surfer enters a *wait-state*. If the wait is too long, she may abort the download; e.g. a user who is presented with several links by a search engine may click one link, find the download too slow, and abort it by clicking another link. Let p_{abort} be the proportion of downloads that are aborted.

We can model this behavior by splitting the *wait-state* into a *wait-abort* state for aborted downloads and a *wait-complete* state for completed downloads. We follow up the latter with a *think* state for when the surfer is viewing a completed download. Since she may click again after viewing, or after aborting a download, we get feedback flows. Fig. 3 shows the resulting session model. (Schroeder et al. call such a model "partly-open" [38].)

Let p_{retry} be the proportion of aborted downloads that are followed by another click in the session, and p_{next} the proportion of completed downloads that are followed by another click in the session. One can think of $1-p_{\text{retry}}$ and $1-p_{\text{next}}$ as the probability of quitting a session after an aborted or completed download. The importance of having p_{next} and p_{retry} is reflected in Prasad and Dovrolis' measurements, which showed that they generated 60–80% of clicks [35].

This surfing session model is bare-bone simple, but we will see that its analysis (Section 6) provides rich details. We next present the equations that relate these variables.

3. User–network model

The key to analyzing traffic equilibrium lies in separating the two forces (user and network) that determine it. We first decide the metric for describing network state, then show how the session model can be decomposed into a *user–network* model consisting of a user curve and a network curve.

3.1. Defining network state: k concurrent downloads

We first need to define network "state". Most analytic models are based on a single bottleneck link [2,6,10]. We consider, however, the general case where congestion may be the result of interaction among multiple links in a bottleneck subnet, without assuming any particular topology (e.g. line/star/circle [8,17]) for this subnet. In the following, *bottleneck bandwidth* refers to the traffic capacity of this subnet.

A download takes some time to complete, so an obvious candidate for describing the subnet's state is the average number of ongoing concurrent downloads, denoted by k; this corresponds to the number of jobs in a processor-sharing model [19].

We do not assume that there is a fixed number of downloads; in fact, we want to use the model to analyze how the equilibrium behaves dynamically (Figs. 13–18). Rather, we aim to facilitate the analysis by deriving closed-form expressions that relate various average values, including k.

There are other candidate metrics for network state. Since our focus is on congestion-induced user behavior, one obvious choice is download time, but that includes round-trip time and server delays that are unrelated to congestion. Another possibility is download speed, but that would mean squeezing the interesting parts of the graphs into the low-bandwidth region (like in Figs. 1 and 2) instead of spreading them out for clarity.

3.2. Deriving a user curve from the surfing model

The *wait-state* in the surfing session model (Fig. 3) thus has k downloads in progress, with clicks arriving at rate r_{click} and aborted at rate $p_{\text{abort}}r_{\text{click}}$. The unaborted click rate is $(1-p_{\text{abort}})r_{\text{click}}$, as illustrated in Fig. 4.

At equilibrium, the unaborted click rate is the rate of completed downloads; this we call the *goodput*, and it is the central performance measure in this paper.

To get the user and network curves, we split the *wait-state*, like in Fig. 5. The *user curve* is then the relationship between the unaborted click rate $(1-p_{\text{abort}})r_{\text{click}}$ and k, and the *network curve* is the relationship between the

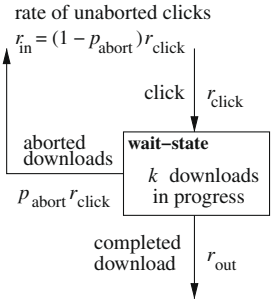


Fig. 4. Flow of downloads (one click generates one download). r_{click} is the click rate and p_{abort} is the proportion of clicks that are aborted, so the input rate of unaborted clicks is $r_{in} = (1 - p_{abort}) r_{click}$.

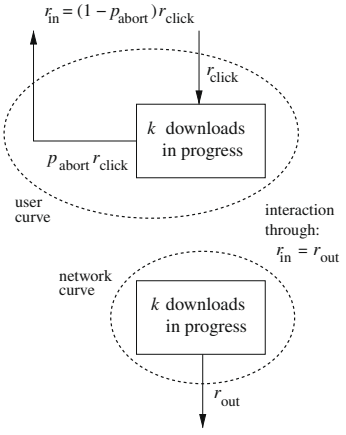


Fig. 5. The model in Fig. 4 is split into two submodels: a user curve that captures the relationship between k and the input rate of unaborted clicks, and a network curve that captures the relationship between k and the output rate of completed downloads. They interact to determine system state through the equilibrium $r_{in} = r_{out}$, giving the goodput.

rate of completed downloads and k. Flow balance in Fig. 3 gives

$$r_{click} = p_{retry} p_{abort} r_{click} + r_{session} + p_{next}(1 - p_{abort}) r_{click},$$

so the user curve is

$$r_{in} = (1 - p_{abort}) r_{click} = \frac{(1 - p_{abort}) r_{session}}{1 - p_{abort} p_{retry} - (1 - p_{abort}) p_{next}}. \quad (1)$$

If we consider p_{abort}, p_{retry} and p_{next} as functions of k—these functions depend on user population, applications, etc.—then Eq. (1) specifies how the input flow r_{in} depends on k. One can also view r_{in} as user *demand* for bandwidth, measured by goodput.

Here, we see explicitly how variation in congestion caused by a change in session arrivals over a macro-granularity timescale is factored into the model through $r_{session}$ in Eq. (1).

3.3. Deriving a network curve with Little's Law

Let T_{abort} and $t_{completed}$ be the average time spent in the *wait-abort* and *wait-complete* states, respectively. By Little's Law [23]—which is robust and holds for an arbitrary bottleneck subnet—Fig. 3 gives

$$k = p_{abort} r_{click} T_{abort} + (1 - p_{abort}) r_{click} t_{completed},$$

so we get a different equation for r_{click}:

$$r_{click} = \frac{k}{p_{abort} T_{abort} + (1 - p_{abort}) t_{completed}}.$$

Let b_{TCP} be the average bandwidth provided by TCP for a (multi-connection) download; b_{TCP} is the variable on the horizontal axis of Fig. 1. Let $S_{completed}$ be the average size for a completed download, so $t_{completed} = S_{completed}/b_{TCP}$. (For our trace experiments in Section 4, these averages are computed with hour-long data.)

One can view the network curve as the locus of equilibrium when the user curve shifts, so

$$r_{out} = (1 - p_{abort}) r_{click} = \frac{k}{\frac{p_{abort}}{1 - p_{abort}} T_{abort} + \frac{S_{completed}}{b_{TCP}}}. \quad (2)$$

The subnet metric k determines the other variables, so Eq. (2) is the equation for the *network curve*. One can also view r_{out} as network *supply* of bandwidth, measured by goodput.

Eq. (2) shows explicitly that TCP controls r_{out} only through b_{TCP}, whereas the user controls p_{abort}, T_{abort} and thus the amount of wasted bandwidth.

Note that our use of b_{TCP} in the derivation does not assume TCP is fair. With different round-trip times per connection, different number of connections per download, etc., the downloads may have very different bandwidths, and b_{TCP} is simply their average. Also, we see explicitly how variation in congestion, as controlled by TCP and active queue management over a micro-granularity timescale, is factored into the model through b_{TCP} in Eq. (2).

The equilibrium $r_{in} = r_{out}$ gives

$$k = \frac{(1 - p_{abort}) r_{session}}{1 - p_{abort} p_{retry} - (1 - p_{abort}) p_{next}} \times \left(\frac{p_{abort}}{1 - p_{abort}} T_{abort} + \frac{S_{completed}}{b_{TCP}} \right),$$

which is the fixed point [8,18] to the user–TCP feedback cycle (Section 1).

4. Model validation with traffic trace

This section has three tasks: first, we present evidence extracted from a `tcpdump` for user backoff ($U^{abort}_{backoff}$ and $U^{session}_{backoff}$); this evidence is independent of our surfing session model (Fig. 3). Second, we present measurements of the three probabilities that parameterize our surfing session model. Third, we show that user and network curves can indeed be extracted from the `tcpdump`, and verify that their intersection agrees with the measured traffic equilibrium.

4.1. SAX: surfer action extraction tool

The difficulties of extracting HTTP information from packet-level data are well-documented [1,15]. Going further up the stack to deduce user actions brings new

Please cite this article in press as: Y.C. Tay et al., Equilibrium analysis through separation of user and network behavior, Comput.Netw. (2008), doi:10.1016/j.comnet.2008.09.008

complications: (i) before a web page finishes downloading, the user may click on one of its links, thus initiating another download. These two downloads are hard to separate by using some time gap threshold, so we need to relate the URL for the second download to the contents of the first download. (ii) How to distinguish between aborted and completed downloads? (iii) Downloads that are generated by software (e.g. auto-requests for updating a web page) need to be filtered out.

To address these additional issues, we implemented a software tool, called SAX, to extract surfer actions from a trace. (SAX is available at www.comp.nus.edu.sg/~tayyc/user-network/ and described in a companion paper [39].) In the following, we present results from using SAX to analyze a trace taken on a link in an academic network; the trace lasted two days, and had about 50 GB of data, 5600 IP addresses and 818000 TCP connections.

4.2. Measuring subnet state with k

Recall (Section 3.1) that we focus on a bottleneck subnet and measure its congestion by k, the average number of concurrent downloads. For the trace, the subnet is simply a bottleneck access link. We first verify that k is an appropriate measure.

One expects the link congestion (however defined) to rise and fall with the session arrival rate. Fig. 6 shows that k does in fact track the 24-h cycle in r_{session}, thus supporting our choice of k as the link metric.

4.3. Verifying user behavior: p_{abort}, p_{next}, p_{retry}

Fig. 7 shows how the SAX-measured values of the probabilities in the surfing session model vary with subnet state k. As expected, p_{abort} increases with k, while p_{next} decreases.

As for p_{retry}, it is mostly constant except for the first 10% of k values (where p_{retry} increases) and the last 10% of k values (where p_{retry} decreases). Note that, as expected, p_{retry} is less than p_{next} at every congestion level k, indicating that a user is less likely to continue a session after an abort than after a completed download.

When p_{abort} is plotted against the download bandwidth b_{TCP}, we get Fig. 1—showing increasing p_{abort} as bandwidth decreases—thus verifying $U^{\text{abort}}_{\text{backoff}}$. Let $n_{\text{comp/session}}$ be the average number of completed downloads per session. Fig. 2 shows that $n_{\text{comp/session}}$ decreases as bandwidth in a session decreases, thus verifying $U^{\text{session}}_{\text{backoff}}$.

With $U^{\text{abort}}_{\text{backoff}}$, it must be that the (completed and aborted) download size distribution has a thinner tail when congestion is heavier. Fig. 8 confirms this: the tail when k is large (36–42 h in Fig. 6) is lower than when k is small (32–36 h and 42–47 h).

It is well known that heavy-tailed downloads are one reason for self-similar traffic [13]. User backoff may therefore help smoothen self-similar traffic as congestion increases.

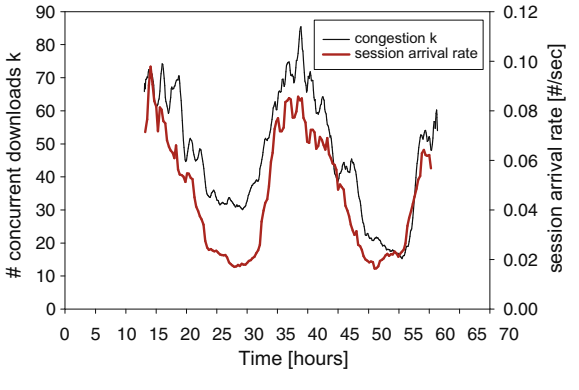


Fig. 6. The link metric k rises and falls with the session arrival rate r_{session} during the trace.

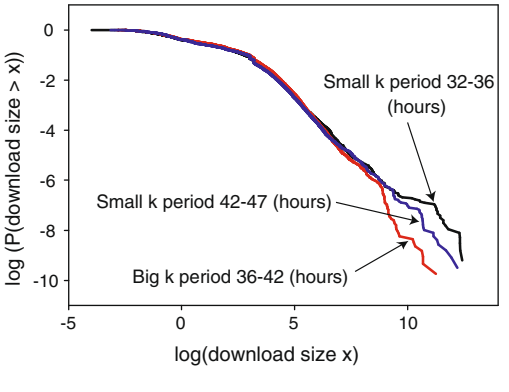


Fig. 8. The distribution for (completed and aborted) download size is lower (i.e. thinner) for large k than for small k; the hours refer to periods in Fig. 6.

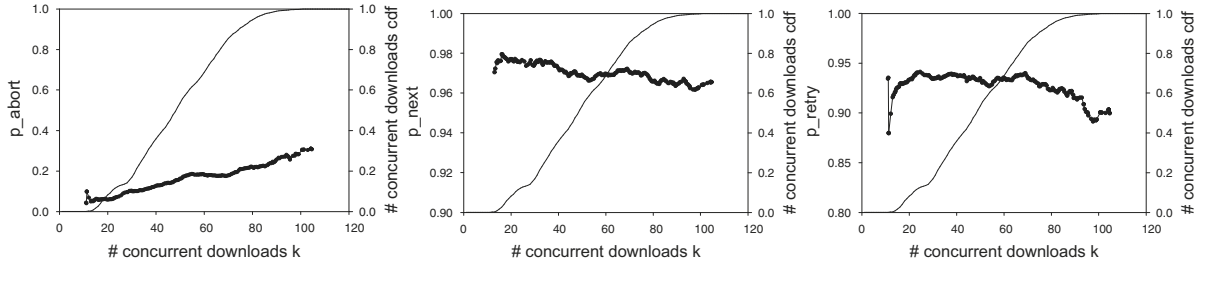
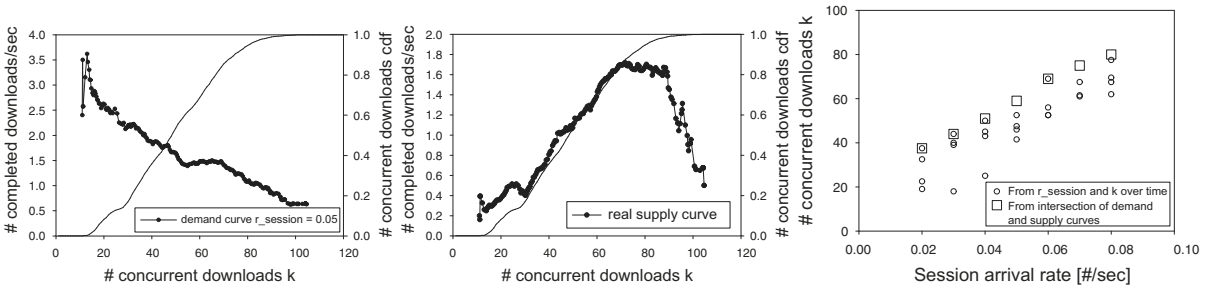


Fig. 7. The effect of congestion on p_{abort}, p_{next} and p_{retry}. The smooth curves are cumulative distribution functions for k.



(a) user curve extracted from trace (b) network curve extracted from trace (c) comparison of equilibrium points

Fig. 9. (a) Shows the user curve for a selected session arrival rate; (b) shows the network curve; (c) compares, for each selected session arrival rate, the intersection point of these two curves to the multiple equilibrium points in the trace.

4.4. Extracting the user curve from the `tcpdump`

Now, $n_{\text{comp/session}} r_{\text{session}}$ gives the rate of unaborted clicks, which is r_{in}, i.e.

$$r_{\text{in}} = r_{\text{session}} n_{\text{comp/session}}. \tag{3}$$

Fig. 9a plots Eq. (3) for $r_{\text{session}} = 0.05$ per sec. (Recall: r_{session} is constant for an open model.) It shows that the user curve mostly decreases as congestion level k increases.

With r_{session} fixed in (3), the user curve is basically $n_{\text{comp/session}}$ plotted against k, so Fig. 9a and 2 are essentially the same curve but with the horizontal axis reversed (since k and session bandwidth have, loosely speaking, a reciprocal relationship). From Eqs. (1) and (3), we get

$$n_{\text{comp/session}} = \frac{1}{\frac{1-p_{\text{retry}}}{\frac{1}{p_{\text{abort}}}-1} + (1 - p_{\text{next}})}.$$

It follows that, even if p_{retry} is roughly constant like in Fig. 7, the p_{abort} increase and p_{next} decrease suffice to decrease $n_{\text{comp/session}}$ as a function of k. This yields the shapes in Fig. 2 and 9a.

4.5. Extracting the network curve from the `tcpdump`

To plot the network curve, we directly measure the rate of completed downloads at each k value. The result is Fig. 9b. This shape is confirmed by simulations (Section 5) and we will discuss it in Section 6.

4.6. Intersection of curves vs measured equilibrium

Our user–network model takes a traffic equilibrium and views it abstractly as a balance between two forces (Fig. 5). A crucial test is whether this abstraction matches the reality.

For this test, we pick $r_{\text{session}} = 0.02, 0.03, \ldots, 0.08$ for our trace data. For each of these values λ, we use Fig. 6 to determine the times t when $r_{\text{session}} = \lambda$, and the k value at t. We thus get three or four (λ, k) measurements for each λ.

We next plot the user curve (Fig. 9a) for this λ and determine k at its intersection with the network curve (Fig. 9b); we thus get a (λ, k) pair from our user–network model. Fig. 9c compares each such (λ, k) pair to the measured values; it shows good agreement, considering the many complications and approximations in trace analysis.

As r_{session} varies over time, the user curve in Fig. 9a moves up and down while the network curve in Fig. 9b remains stationary; their intersection thus yields a k value that varies with time, like in Fig. 6.

4.7. Variation from, and changes in, the averages

Our equations are derived with average values. In truth, two users may behave differently, and two web servers may run different TCP versions, so p_{abort}, T_{abort}, b_{TCP}, etc. are aggregates. While one may possibly refine our model to account for individual variation, validating the refinement will be difficult.

The average values themselves can also change. Office workers in the day and home users at night may behave differently, causing the user curve to change its shape. Similarly, a hardware malfunction or software update may reshape the network curve. Such changes in the curves do not affect the qualitative analysis that is the focus of this paper.

However, for the user–network model to be used quantitatively in bandwidth provisioning or overload monitoring (Sections 6.8 and 6.9), one will need to calibrate the model parameters dynamically. Doing this at line rate is probably an overkill and unnecessary; one could simply collect a `tcpdump` periodically and run SAX on the traces offline to get updates for the parameters.

5. Simulation study using NS2

The user–network model abstractly decomposes the equilibrium into two curves. The previous section shows that we can in fact extract these two curves from a `tcpdump`. However, trace measurements offer very little scope for studying the user and network curves. For example, the user population, TCP versions and link bandwidth are fixed. To get greater flexibility for our study, we resort to `NS2` simulation [30]. In particular, we want to check if the network curve captures the effects of congestion control, queue management, etc.

This section describes the simulation, how we obtain the network and user curves, fitting the network curve

with the PFTK equation [33], and the effects of congestion control and queue management.

5.1. FTO: frustration timeout T_{abort}

The network curve (2) depends crucially on p_{abort} and the average time T_{abort} before an abort. For the simulation, we model $U_{\text{backoff}}^{\text{abort}}$ with a *frustration timeout* (*FTO*) [32]: a download is aborted if and only if its duration exceeds the FTO, specified by a constant T_{abort}. Thus,

$$p_{\text{abort}} = \text{Prob}(\text{download time} > T_{\text{abort}}). \quad (4)$$

This model is imperfect: it does not consider file size (a user may be more patient when downloading a video clip), nor distinguish between server and network delays. We consider the FTO as a first-order approximation of abort behavior. There are several more sophisticated models in the literature that one can adopt for T_{abort} [6,10,41].

Note that we use FTO as a model for abort behavior *only for the simulation*, for which T_{abort} is an input parameter to the simulator. For the trace analysis (Section 4), T_{abort} can be directly measured; for the equilibrium analysis (Section 6), T_{abort} may have an arbitrary distribution.

5.2. Simulation scenario and parameters

We focus the simulations on how TCP affects the network curve and use the simplest download model:

$$1 \text{ download} = 1 \text{ connection}. \quad (5)$$

Like FTO, this simplification applies only to the simulation.

We use the standard dumbbell configuration with k sources sending data to k destinations over a bottleneck link. To maintain k concurrent connections, all sources are in busy states all the time; in particular, whenever a download completes or aborts, another starts immediately for the same source/destination pair.

In reality, k varies with arrival rate (Fig. 6), and our model does reflect this—see end of Sections 4.6 and 6. This modeling technique (of fixing k in the simulation) is similar to the use of flow-equivalent servers in queuing network analysis [23].

We restrict the simulations to small values of k, as NS2 becomes very slow when the number of sources is large. It follows that we have to set the bottleneck bandwidth to unrealistically small values, but this is not an issue since the bandwidth per connection b_{TCP} is more important.

Unless otherwise stated, the bottleneck link has 50 ms delay, access link speed is 100 Mbps, each download transfers thirty 536-byte packets, and FTO is 20 s.

5.3. Simulated network curve

As k is a simulation parameter, the network curve is a straightforward plot of completed download rate against k. Fig. 10a shows the simulated network curves for two bottleneck bandwidths.

The simulated network curves are similar in shape to Fig. 9b, except that the trace version does not show clearly what the tail on the right looks like. This is because the tail is where congestion is heavy (large k), so it is hard to observe in a real network that is well-provisioned. The simulation thus shows us a part of the network curve that is crucial but hard to observe with a trace.

Note the shift in network curve when bottleneck bandwidth is changed from 1 Mbps to 2 Mbps. Again, observing such a shift through a trace is difficult. (Section 6 discusses this shift.)

5.4. Plotting user curve without simulating sessions

We could implement the session model and user backoff in NS2, and generate the user curve with simulations—this is our current work. For this paper, our simulation is focused on the network curve.

Although the experiments do not simulate users, one can plot the user curve, as follows: From the trace measurements, do a linear regression fit for p_{next} and set $p_{\text{retry}} = 0.93$ (see Fig. 7), and use p_{abort} measured from the network curve simulation; substituting these values into Eq. (1), we get the user curves in Fig. 10b for two session arrival rates. Note the similarity in shape to the trace version in Fig. 9a, and the shift when r_{session} is changed.

5.5. Traffic equilibrium where the curves intersect

Fig. 10c shows how a pair of user and network curves intersect to determine the traffic equilibrium. We will analyze this equilibrium in Section 6.

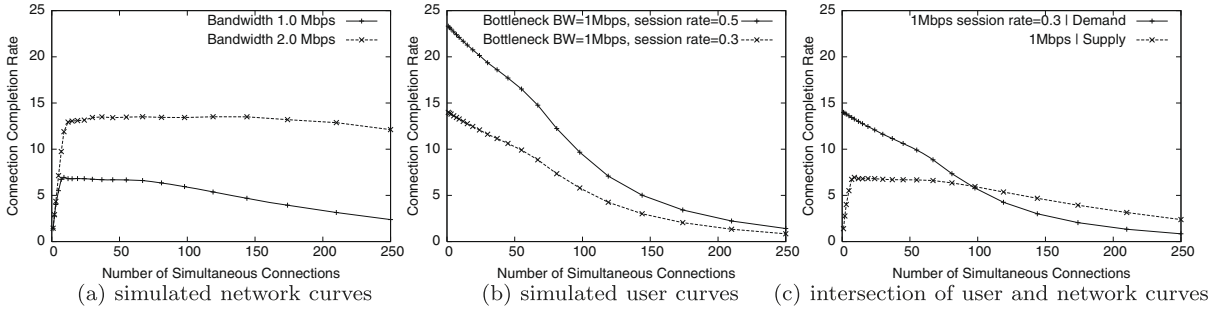


Fig. 10. Default simulation parameters: 100 Mbps access links, a 50 ms bottleneck link delay, thirty 536-byte packets per transfer and a 20-s FTO (frustration timeout). For the simulations, 1 download = 1 connection.

5.6. Using the PFTK equation for b_{TCP}

Recall that b_{TCP} in Eq. (2) is the download bandwidth provided by TCP. With the download simplification (5) for the simulation, there are several equations in the literature describing TCP throughput that are candidates for b_{TCP}. One of the earliest was by Padhye et al. [33] for TCP Reno.

The PFTK equation expresses TCP throughput $T(p)$ as a function of p, the probability of a loss indication. The equation was derived without including slow start, making it inaccurate when congestion is low, window size grows large and slow start dominates. To get around this issue, we simulated a lossy bottleneck link that dropped 10% of the packets; this reduces the window size and the effect of slow start.

We then compare the simulated r_{out} and the calculated version (2) using $b_{TCP} = T(p)$, with p and p_{abort} measured from the simulation. Fig. 11 shows excellent agreement.

The PFTK equation has other shortcomings, but our point here is just to show that one can refine Eq. (2) further by substituting an appropriate formula for b_{TCP}.

5.7. Congestion control, queue management and r_{out}

With our surfing session model, user reacts to congestion through p_{abort}, p_{next} and p_{retry}. The network, however, reacts through TCP's congestion control. To study how congestion control affects the network curve, we simulated three versions of TCP: Reno, Tahoe and the original TCP—RFC793 [34]. Fig. 12 compares their network curves.

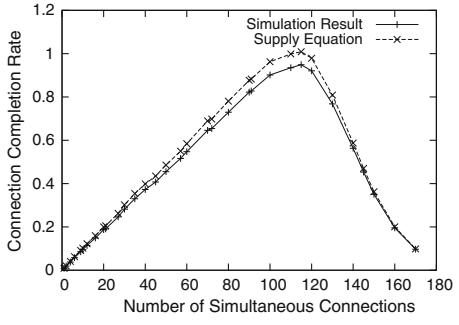


Fig. 11. Substituting the PFTK equation for b_{TCP} in Eq. (2) gives a good fit for the measured network curve.

Without congestion control, RFC793 is more aggressive, so it is no surprise that its linear segment is higher for small k. The three versions have similar tails in Fig. 12a, where the average download size is 200 packets. When this is increased to 400 packets, Fig. 12(b) shows that RFC793 has a clearly higher tail. This is counter-intuitive—one expects RFC793 to suffer when congestion is high (i.e. large k), rather than provide a higher goodput.

This happens because RFC793 is unfair: With download size increased to 400 packets, fewer connections complete before the FTO (=30 s), so the tails drop in Fig. 12b. As Tahoe and Reno are fairer, their connections have similar download times, so when pushed beyond the FTO by an increased k, they suffer a sudden increase in p_{abort} and a sharp drop in r_{out}.

In contrast, RFC793 allows some connections to hog bandwidth while others stall. This bigger spread of bandwidth allocation means there are more fast connections that can complete before the FTO, so RFC793's tail drops gradually. One expects this "advantage" would be lost if the router enforces fairness.

Indeed, Fig. 12c shows that, when the default management policy (*droptail*) is changed to *RED* (random early detection), which has fairer bandwidth allocation, RFC793's tail drops below those of Tahoe and Reno.

In general, the shape of r_{out} is affected by any factor that influences TCP throughput. For example, our experiments (omitted here for lack of space) show that a large maximum window size separates the tails for Tahoe and Reno, and a heavy-tailed download size distribution adds jitter to the curve.

6. Equilibrium analysis via user–network model

Having validated the models by measurement and studied the network curve by simulation (Sections 4 and 5), we now return to the models (Sections 2 and 3) for an abstract analysis. Recall that our objective is to offer a technique for analyzing traffic equilibrium that (i) takes into account the interaction between user and network behavior, (ii) is intuitive and (iii) can flexibly address a broad range of issues. We now illustrate how the two models can realize these goals by examining the location, stability, robustness and dynamics of the equilibrium. To do so, we need to first understand the shape of the user and network curves.

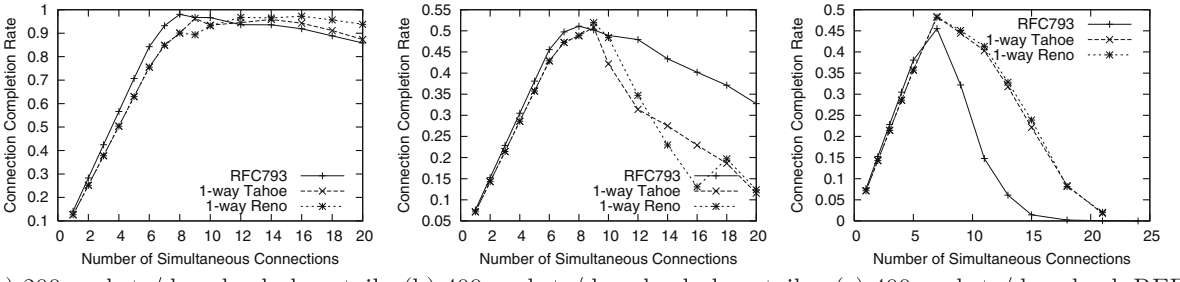


Fig. 12. The network curve with/without congestion control, for drop-tail and RED (FTO = 30 s).

Please cite this article in press as: Y.C. Tay et al., Equilibrium analysis through separation of user and network behavior, Comput.Netw. (2008), doi:10.1016/j.comnet.2008.09.008

6.1. The shape of user curve r_{in}

We can rewrite the user curve formula (1) as

$$r_{\text{in}} = \frac{r_{\text{session}}}{(1-p_{\text{next}}) + \frac{1-p_{\text{retry}}}{\frac{1}{p_{\text{abort}}}-1}}.$$

When congestion is low, $p_{\text{abort}} \approx 0$ and

$$r_{\text{in}} \approx \frac{r_{\text{session}}}{1-p_{\text{next}}} \approx c_1 r_{\text{session}} \quad \text{for small } k, \tag{6}$$

where c_1 is the number of completed downloads per session for $k = 1$. When k increases, the decrease in bandwidth per user and increase in delay cause users to backoff: p_{abort} increases, while p_{next} and p_{retry} decrease, causing r_{in} to decrease. (Note that goodput only measures completed downloads; the total download rate—including aborts—may increase with k.)

Thus, the user curve starts on the left at height $c_{1r_{\text{session}}}$, then drops as k increases. This explains the shapes in Figs. 9a and 10b. When session arrival rate r_{session} fluctuates over a macro timescale, its scaling effect on r_{in} shifts the user curve, like in Fig. 10b.

6.2. The shape of network curve r_{out}

Now consider the network curve

$$r_{\text{out}} = \frac{k}{\frac{p_{\text{abort}}}{1-p_{\text{abort}}} T_{\text{abort}} + \frac{S_{\text{completed}}}{b_{\text{TCP}}}}.$$

When k is small and congestion is negligible, the download bandwidth is mostly constrained by receiver window size and access link bandwidth, so b_{TCP} does not change with k. Moreover, $p_{\text{abort}} \approx 0$ if users seldom abort when congestion is low. Hence,

$$r_{\text{out}} \approx \frac{k b_{\text{TCP}}}{S_{\text{completed}}} \text{ grows linearly with } k, \text{ for small } k.$$

As k increases, the total throughput approaches the bottleneck capacity, so r_{out} flattens out. Any further increase in k will translate into longer queues, slower response and dropped packets. If users notice and backoff, $U^{\text{abort}}_{\text{backoff}}$ increases p_{abort}. This and the decrease in b_{TCP} result in a drop in r_{out}. We thus get the shapes in Figs. 9b and 10a.

For large k, some bandwidth is lost to retransmissions, but the drop in r_{out} is mostly caused by aborted downloads.

An increase in bottleneck bandwidth will shift most of r_{out}, like in Fig. 10a, except for the initial linear segment, where the bandwidth increase has a small effect on b_{TCP}.

Note that link capacity, queue management, router configuration, etc. affect the shape of r_{out} through b_{TCP}—this includes the micro timescale congestion effects (packet drops, window size adjustment, etc.). One can analyze their impact mathematically by substituting the appropriate formula for b_{TCP} into Eq. (2), like we did for Fig. 11.

6.3. Stability of the equilibrium

The user and network curves may intersect at multiple points, but the equilibrium at these intersections may differ in stability. To see this, we adapt Kleinrock and Lam's fluid argument [26].

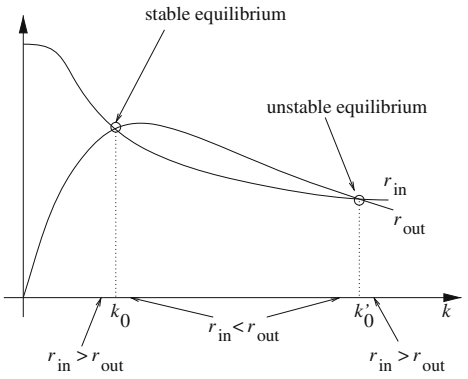


Fig. 13. Equilibrium at k_0 is stable; the one at k'_0 is unstable.

Fig. 13 shows an $r_{\text{in}} - r_{\text{out}}$ pair that intersect at $k = k_0$ and $k = k'_0$. Now consider a small $\varepsilon > 0$. At $k = k_0 - \varepsilon$, $r_{\text{in}} > r_{\text{out}}$; by Fig. 4, k increases and moves towards k_0.

Similarly, at $k = k_0 + \varepsilon$, we have $r_{\text{in}} < r_{\text{out}}$, so k decreases and again moves towards k_0. Thus, k hovers around a stable equilibrium as r_{in} and r_{out} shift over a macro timescale (as session arrivals, protocol mix, etc. change).

The same argument shows that k is decreasing at $k = k'_0 - \varepsilon$, and increasing at $k = k'_0 + \varepsilon$, in Fig. 13. Therefore, any perturbation to k will cause it to move away from that equilibrium; i.e. the equilibrium at k'_0 is unstable.

There may be stable equilibrium points for larger $k(> k'_0)$, but the goodput may be so low that performance has collapsed.

This stability analysis agrees with Garetto et al.'s results [18]. They have identified an unstable region for TCP; if the traffic equilibrium is close to this region, a random traffic fluctuation can push the system into that region, causing a big and sudden increase in the number of active connections.

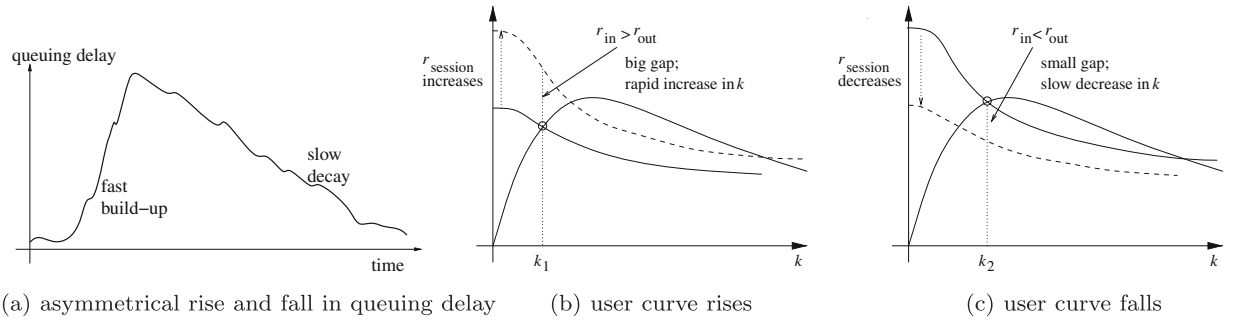
Their observations can be interpreted with Fig. 13: k'_0 marks the start of an unstable region; if the traffic equilibrium is at k_0, and k_0 is close to k'_0, then a fluctuation can push k beyond k'_0, causing k to slip further out to the next stable equilibrium at some k''_0.

Users may react to the degraded equilibrium by backing off, but new sessions continue to arrive. This stable equilibrium at k''_0 must wait for a macro timescale change in traffic, e.g. a drop in r_{session} after midnight [6,18]. The user curve then shifts left, causing the stable equilibrium at k''_0 to drift towards the unstable equilibrium at k'_0; once there ($k''_0 = k'_0$), the momentum then drives k down further, and the link returns to a stable equilibrium at k_0 with a high goodput.

We see here how, by treating the equilibrium as a balance between user and network behavior, we can break the user–TCP feedback cycle in reasoning about the equilibrium, and get a deeper understanding of traffic dynamics.

6.4. Congestion buildup and decay

Hohn et al. have observed in their router measurements [21] that a busy period is asymmetrical: the buildup in



(a) asymmetrical rise and fall in queuing delay (b) user curve rises (c) user curve falls

Fig. 14. The decreasing gap between user curves results in asymmetrical forces that underly the buildup and decay in (a).

queuing delay is faster than its decay. This is illustrated in Fig. 14a.

The asymmetry observed by Hohn et al. happens over a micro timescale (milliseconds), but it appears over a macro timescale as well: Iyer et al.'s measurements show that, even in the backbone, some links suffer sudden overloads (caused by, say, traffic redirected by link failures) that can persist for hours [22].

Such asymmetry is common in queuing systems, but amplified by user behavior in the following way: Suppose the equilibrium is at $k = k_1$ in Fig. 14b, and there is an increase in $r_{session}$ from λ_1 to λ_2, causing the user curve to shift up. Now, r_{in} exceeds r_{out} and k increases.

After a new equilibrium is reached at $k = k_2$ in Fig. 14c, suppose $r_{session}$ returns to λ_1, and the user curve drops, so r_{out} exceeds r_{in} and k decreases.

From Eq. (3), the gap between the user curves is $n_{comp/session}(\lambda_2 - \lambda_1)$. Since $n_{comp/session}$ is a decreasing function of k, the gap at k_1 is larger than the one at k_2, so the imbalance that drives k up in Fig. 14b is greater than the imbalance that drives k down in Fig. 14c. The rise and fall in congestion are thus driven by asymmetrical forces.

Again, the user–network decomposition offers us insight into traffic dynamics and answers, in part, question Q4 about the impact of a load change. The next two subsections will consider the stability aspect to Q4.

6.5. Internet robustness: the role of user behavior

The persistent overload measured by Iyer et al. is also observed in simulations by Prasad and Dovrolis, who point out that TCP congestion control cannot curtail this persistence in an open system [35]. Worse, TCP cannot prevent a loss of equilibrium by itself, as we now show with our model.

To see the complementary role played by user behavior, consider first the case where users do not backoff, so p_{abort}, p_{next} and p_{retry} in the surfing session model are constants, and r_{in} remains the same whatever the congestion level k may be. (Note that congestion slows down web retrievals and reduces the click rate per user, but users stay longer in the system, so the total click rate remains constant.)

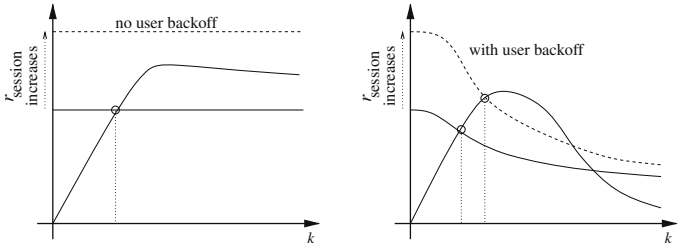
Hence, if an increase in $r_{session}$ is sufficiently large, the user curve will rise above the network curve, like in Fig. 15a. With the two curves disengaged, no equilibrium is possible and performance plummets: link buffers overflow, TCP timers go off, retransmissions dominate, etc., and recovery must wait for existing users to quit and for $r_{session}$ to drop.

On the other hand, if users do backoff, then the user curve is decreasing. Although the network curve is also depressed by p_{abort}, equilibrium is possible for a higher range of $r_{session}$, as illustrated in Fig. 15b.

User backoff thus acts as a form of congestion control. As Bonald and Roberts noted: In an overload, network stability is in fact maintained by user impatience [6].

User backoff plays this stabilizing role for UDP applications too: Bu et al. did a detailed study of aborted VoIP calls when perceived quality is degraded by congestion, and concluded that user backoff will prevent such UDP applications from bringing down the Internet [7].

So, one answer to Q2 is that user backoff helps performance. However, it also hurts because abortion forces down the network curve's tail. We next see how this can lead to a performance collapse.



(a) loss of equilibrium (b) no loss of equilibrium

Fig. 15. How the shape of the user curve affects robustness: (a) Without user backoff, a sufficiently large increase in $r_{session}$ will result in a loss of equilibrium. (b) With user backoff, equilibrium can be maintained for a larger range of $r_{session}$. Recall (6) that user curves start on the left at height $c_1 r_{session}$.

Please cite this article in press as: Y.C. Tay et al., Equilibrium analysis through separation of user and network behavior, Comput.Netw. (2008), doi:10.1016/j.comnet.2008.09.008

6.6. Internet robustness: the role of TCP

Notice from Fig. 12 that different TCP versions make only marginal differences to the linear segment and height of the network curves—the big differences are in the tail. This is intuitive, since these TCP versions differ in their congestion control, so their differences show up only for large k.

On the other hand, the tail of a performance curve, being lower than its height, is usually considered a sub-optimal operating range that is to be avoided. Much engineering wisdom and technical analysis have gone into designing TCP variations, yet it appears now from the network curve that this effort is focused on a sub-optimal region. This brings us back to Q5: What is TCP's role in determining the equilibrium?

The motivation in the TCP variations is varied (fairness, latency, etc.). We now interpret the variations in terms of robustness.

As Fig. 12c shows, modifications to RFC793 can lift the tail of the network curve. If the tail drops rapidly, then a sudden upward shift of the user curve (e.g. flash crowd or traffic failover) can drive the equilibrium point deep down the tail, as illustrated in Fig. 16a, resulting in a performance collapse.

To be robust, a TCP version must hold the tail as high as possible, and for k as large as possible, so that—even if the user curve suddenly shifts up — the equilibrium will stay high, like in Fig. 16b, and there is no performance collapse.

Since the drop in the tail is largely caused by abortion, lifting the tail would require changes to TCP that take user behavior into account, as demonstrated by Yang and de Veciana's TCP SAReno [42]. Instead of fair sharing, SAReno uses residual transfer size to modulate the congestion window (using, essentially, the queuing discipline called Shortest-Remaining-Processing-Time-First [38]). The effect is to reduce user-perceived delays and decrease abortion. Indeed, their simulation shows that SAReno lifts up the goodput tail for an overloaded link.

We see here how micro timescale congestion control shapes the network curve, and thus affect the medium timescale equilibrium.

6.7. A primal–dual formulation of user–network interaction

Besides the above user–network view of SAReno's impact on equilibrium, there is another perspective. Kelly and Low et al. view traffic equilibrium as the result of a primal–dual optimization [24,28]. For example, the primal variable is sending rate controlled by TCP, as "user", while the dual variable is a congestion signal (e.g. packet drop probability or delay) controlled by active queue management (AQM), and the utility to be maximized depends on the TCP/AQM pair. In our user–network model, TCP is not the "user", but part of the network.

Low and Lapsley have used Lagrange duality to show that TCP/AQM maximizes transport utility. One can similarly use a primal–dual formulation of user–network decomposition to interpret user backoff as utility maximization, with $U_{\text{backoff}}^{\text{abort}}$ and $U_{\text{backoff}}^{\text{session}}$ as the "primal algorithm" and the TCP/AQM pair as the dual algorithm.

To illustrate, assume $p_{\text{abort}} = 0$ and consider just $U_{\text{backoff}}^{\text{session}}$. Let r_{click} be the primal variable, $t_{\text{completed}}$ the dual variable that is the download time (Section 3.3) serving as a congestion signal, U the utility function and C the bottleneck capacity. Then the optimization problem is

$$\max U(r_{\text{click}}) \quad \text{such that} \quad S_{\text{completed}} r_{\text{click}} \leqslant C.$$

By $U_{\text{backoff}}^{\text{session}}$, suppose $p_{\text{next}} = g(t_{\text{completed}})$ for some decreasing function g. One can determine g by, say, assuming p_{next} is a linear function of k (see Fig. 7) and $k = Ct_{\text{completed}}/S_{\text{completed}}$. By the Karush–Kuhn–Tucker Theorem, the optimal solution has

$$U'(r_{\text{click}}) = S_{\text{completed}} t_{\text{completed}} = S_{\text{completed}} g^{-1}(p_{\text{next}}).$$

With $p_{\text{abort}} = 0$, Eq. (1) gives

$$U(r_{\text{click}}) = \int S_{\text{completed}} g^{-1}\left(1 - \frac{r_{\text{session}}}{r_{\text{click}}}\right) \mathrm{d}r_{\text{click}}.$$

In this fashion, one can discover the utility function underlying user behavior in Fig. 7.

One could, like Low et al., go further and use the primal-dual formulation to design TCP/AQM algorithms that take user backoff into account, and compare the result to SAReno.

6.8. Bandwidth provisioning

We have illustrated how the simulated user and network curves can move individually. In reality, both may move together, and this affects bandwidth provisioning.

As Fig. 10a shows, an increase in bottleneck bandwidth raises the network curve. Surfers will react to the change in bandwidth by changing (for the same k) $p_{\text{abort}}, p_{\text{next}}$ and p_{retry}, thus raising the user curve as well. It follows that

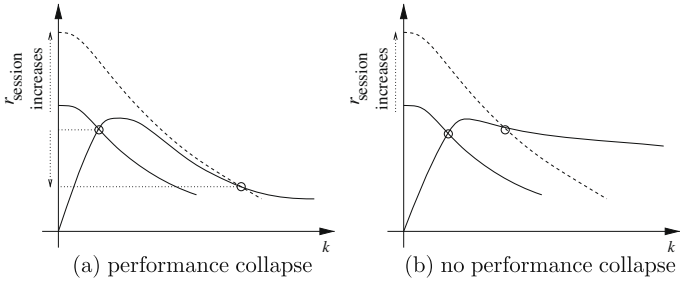


Fig. 16. How the tail of the network curve affects robustness: (a) If the tail drops steeply, a sufficiently large upward shift in user curve can result in a performance collapse. (b) If the tail drops gradually, high performance can be maintained for a larger range of r_{session}.

Please cite this article in press as: Y.C. Tay et al., Equilibrium analysis through separation of user and network behavior, Comput.Netw. (2008), doi:10.1016/j.comnet.2008.09.008

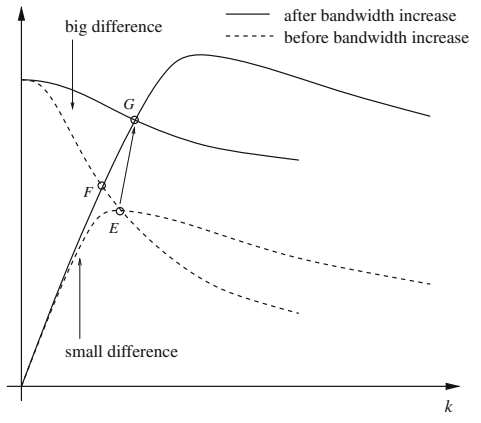


Fig. 17. An increase in bottleneck bandwidth raises both the user and network curves. Consequently, the equilibrium point E does not slide along the old user curve to another point F, but jumps to point G on another user curve.

the equilibrium point does not just slide along the old user curve, but jumps onto another curve altogether, as illustrated in Fig. 17.

We can also see from this figure a problem with engineering arguments like the one in the user–TCP feedback cycle: they do not distinguish between a movement along a user curve (e.g. from E to F) and a movement of the curve itself.

Bandwidth provisioning thus requires not just an understanding of the shape of the user curve, but also how the entire curve moves with network changes. Whether increased bandwidth relieves congestion (Q3) thus depends on how the user and network curves behave.

Alternatively, one can reformulate the model so that the user curve does not change when bottleneck bandwidth is changed. This is possible if users perceive congestion through its effect on b_{TCP}. One can then redo Fig. 7 and plot $p_{\text{abort}}, p_{\text{next}}$ and p_{retry} as functions of b_{TCP} (instead of k). The user curve can then be plotted against b_{TCP}, and this curve does not change with bottleneck bandwidth. For a proposed bottleneck bandwidth, one can estimate k as a function of b_{TCP}, and use Eq. (2) to plot the corresponding network curve and determine the new equilibrium. This goodput-vs-b_{TCP} formulation of the user–network model may be more suitable for bandwidth provisioning.

6.9. Watching for an overload

Fredj et al. have cautioned that, with bandwidth provisioning, precise guarantees are in vain because random fluctuation can overwhelm deterministic design [17]. In our context, consider Fig. 13. For such an r_{in}-r_{out} pair, Kleinrock and Lam point out that the equilibrium cannot stay around k_0 indefinitely—with probability 1, a random fluctuation will cause the number of concurrent downloads to cross the critical point k_0' in finite time T_0', say, resulting in a stable, collapsed equilibrium further out. If T_0' is large, r_{session} may drop first, making the collapse unobservable. In any case, one can pre-empt a collapse by keeping the equilibrium where $\frac{\mathrm{d}r_{\text{out}}}{\mathrm{d}k} > 0$; the following is a guide:

Proposition 1. *Suppose the averages T_{abort} and $S_{\text{completed}}$ are constant.*

(a) *If* $\frac{k}{p_{\text{abort}}}\frac{\mathrm{d}p_{\text{abort}}}{\mathrm{d}k} \leqslant 1 - p_{\text{abort}}$ *and* $\frac{k}{b_{\text{TCP}}}\frac{\mathrm{d}b_{\text{TCP}}}{\mathrm{d}k} \geqslant -1$, *then* $\frac{\mathrm{d}r_{\text{out}}}{\mathrm{d}k} > 0$.

(b) *If* $\frac{k}{p_{\text{abort}}}\frac{\mathrm{d}p_{\text{abort}}}{\mathrm{d}k} > 1 - p_{\text{abort}}$ *and* $\frac{k}{b_{\text{TCP}}}\frac{\mathrm{d}b_{\text{TCP}}}{\mathrm{d}k} < -1$, *then* $\frac{\mathrm{d}r_{\text{out}}}{\mathrm{d}k} < 0$.

An Internet service provider can use a tool like SAX (Section 4) to monitor $\frac{k}{p_{\text{abort}}}\frac{\mathrm{d}p_{\text{abort}}}{\mathrm{d}k}$ and $\frac{k}{b_{\text{TCP}}}\frac{\mathrm{d}b_{\text{TCP}}}{\mathrm{d}k}$. If they violate one of the inequalities in Proposition 1(a), then the equilibrium is approaching overload.

If they satisfy both inequalities in Proposition 1(b), then not only is the equilibrium in danger of collapse, but there is already bandwidth wastage from abortion. Clearly, the equilibrium is in an undesirable operating zone, and the provider may need to take action (e.g. admission control or traffic diversion [22]) to lower congestion to where the inequalities in Proposition 1(a) are true again.

This paper focuses on providing a technique for intuitive analysis of the equilibrium. Nonetheless, Proposition 1 shows how the underlying closed-form expressions can facilitate a formal, technical analysis.

7. Model variations

The models in Sections 2 and 3 make two important assumptions: (i) users react to congestion and (ii) r_{session} is independent of congestion level. This section relaxes these assumptions, thus further demonstrating our models' flexibility.

7.1. Non-reactive elephantine flows

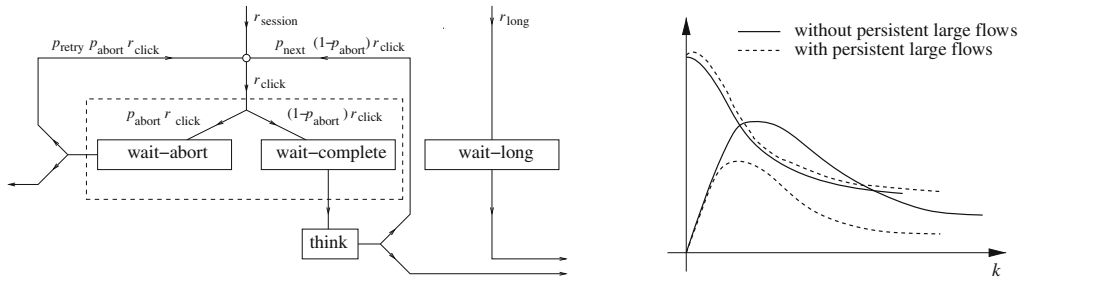
We now address the question Q1 on how elephantine transfers affect the equilibrium.

Users react to congestion only if they are monitoring the progress of a download, which may not be the case—a user may start downloading a large file over a modem link, then go for dinner or to bed. Only a small fraction of Internet connections carry such large flows, but they dominate in terms of byte count [37]. These large downloads can last for a long time, so with only TCP congestion control and no user backoff to regulate them, intuition says that they can cause a performance collapse for reactive users. How can our model capture this intuition?

To do so, we add a class of downloads that arrive at rate r_{long}. (A peer-to-peer application may spawn multiple connections to download fragments of a large file, but our models already allow a download to include multiple TCP connections.) Assume each download has average size S_{long}, and does not abort—regardless of congestion—as illustrated in Fig. 18a. The user and network curves become

$$r_{\text{in}} = \frac{r_{\text{session}}}{1 - p_{\text{next}} + \frac{1-p_{\text{retry}}}{\frac{1}{p_{\text{abort}}}-1}} + r_{\text{long}}, \tag{7}$$

$$r_{\text{out}} = \frac{k - r_{\text{long}}\frac{S_{\text{long}}}{b_{\text{TCP}}}}{\frac{p_{\text{abort}}}{1-p_{\text{abort}}}T_{\text{abort}} + \frac{S_{\text{completed}}}{b_{\text{TCP}}}} + r_{\text{long}}. \tag{8}$$



(a) A two-class model for reactive/non-reactive flows. (b) The effect of non-reactive flows on r_{in} and r_{out}.

Fig. 18. (a) Non-reactive large flows are modeled by adding a class to Fig. 3. Requests in this class arrive at a rate r_{long}, the average size of each download is S_{long}, and they do not abort. (b) Although non-reactive large flows may form a small percentage of TCP connections (so their effect on the user curve is small), they can significantly lower the network curve, possibly causing a performance collapse (despite TCP congestion control).

As large flows form a small fraction of Internet connections, we have $r_{long} \ll r_{session}$. Moreover, without $1-p_{retry}, 1-p_{next}$ and p_{abort} to magnify its contribution, r_{long} makes a negligible difference to r_{in}, as illustrated in Fig. 18b.

The impact on r_{out}, however, is significant. Comparing Eqs. (8) and (2), we see that (for the same value of k) r_{out} has an additional term r_{long} and a reduced term for the user-responsive flows. In effect, the persistent large flows reduce the bandwidth available to the others by occupying $r_{long}S_{long}/b_{TCP}$ of the k downloads. (Indeed, Saroiu et al.'s measurements show that peer-to-peer downloads take up some two-thirds of concurrent HTTP flows [37].)

The elephantine flows thus push down the network curve, like in Fig. 18b. This push can separate the user and network curves, cause a loss of equilibrium and induce a performance collapse. This can happen even if the large flows are using TCP congestion control, so the conclusion also applies even if the applications use TCP-friendly UDP flows.

7.2. Surfing session model: closed system

So far, we have focused on an open model, in which sessions arrive at a constant rate $r_{session}$ (regardless of congestion). However, it may sometimes be more relevant to consider a closed model, in which there is a constant number of users. For example, an Internet service provider may have a small customer population. Then, $r_{session}$ depends on how many of them are already active.

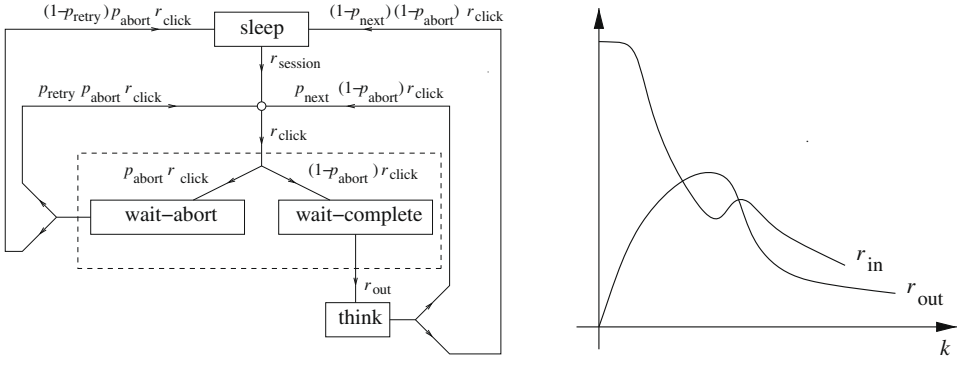
To model such a system, we close the loops in Fig. 3 by introducing a *sleep* state for inactive users who are not in download-cum-think sessions, as shown in Fig. 19a.

Let T_{sleep} and T_{think} be the average time a user spends in the *sleep* and *think* states, respectively, and suppose the user population has size N_{user}. Then, the user curve becomes

$$r_{in} = \frac{N_{user} - k - \frac{kT_{think}}{\frac{p_{abort}}{1-p_{abort}}T_{abort} + \frac{S_{completed}}{b_{TCP}}}}{T_{sleep}\left(1 - p_{next} + \frac{1-p_{retry}}{\frac{1}{p_{abort}}-1}\right)} \tag{9}$$

The network curve is unaffected.

Fig. 19b illustrates the new r_{in} equation. It shows that, while congestion is light, r_{in} behaves like in the open model, decreasing as k increases. However, when congestion becomes serious and p_{abort} approaches 1, the number of users in *think* state (k_{think}) drops. By Eq. (9), this can cause a local increase in r_{in}. However, this non-monotonicity does not affect the equilibrium and stability analysis in Section 6.



(a) Closed surfing session model (b) The shape of r_{in} and r_{out} for a closed model.

Fig. 19. (a) Changing Fig. 3 into a closed model by adding a sleep state for users who are not in a session. (b) Heavy congestion can cause a drop in k_{think} (the number of users who are viewing their downloads), resulting in a local increase for r_{in} in Eq. (9).

Please cite this article in press as: Y.C. Tay et al., Equilibrium analysis through separation of user and network behavior, Comput.Netw. (2008), doi:10.1016/j.comnet.2008.09.008

Prasad and Dovrolis point out that closed-loop traffic is always stable and cannot cause an overload. This is true if users do not abort, so the r_{out} tail is high like in Fig. 16b. Otherwise, abortions will force down the tail, and a performance collapse in a closed system becomes possible, like in Fig. 16a for an open system.

8. Conclusion

The Internet's complexity offers tremendous scope for the application of sophisticated techniques (game theory, mathematical programming, stochastic processes, etc.) to equilibrium analysis, but the resulting models are not easy to understand or use. In contrast, our objective here is to offer a simple, accessible model that engineers and researchers can use *routinely* to cut through the complexity and formulate back-of-an-envelope arguments about traffic equilibrium.

(Indeed, our user–network model is inspired by a similar separation in economics—despite market complexities, the demand-supply decomposition of price equilibrium is a simple, yet powerful and enduring paradigm in economic discourse.)

We have made equilibrium analysis easy by

(i) separating user and network behavior and
(ii) describing them with closed-form expressions.

We capitalize on the robustness of Little's Law to overcome the difficulty in modeling multi-link bottlenecks (Section 3.3).

We also incorporate—in one model—factors from the entire timescale: While focusing on user–network interaction at medium granularity, the equations factor in micro-granularity TCP/AQM characteristics through b_{TCP} (Figs. 12 and 16) in the network curve (2), as well as macro-granularity population characteristics through $r_{session}$ (Figs. 14, 15, 18, 19) in the user curve (1). We refined the model in Section 7.2 to study how a macro-granularity effect on $r_{session}$ affects the equilibrium, and one can similarly examine how micro-granularity TCP/AQM issues affect b_{TCP} and, consequently, the equilibrium.

The user–network separation is natural, once we observe the equilibrium as a balance between input flow r_{in} and output flow r_{out}. For example, we see now that a flash crowd affects the equilibrium by moving the user curve, while a router misconfiguration's effect is in warping the network curve. The separation is also necessary, if we are not to confuse behavioral change that is a slide along one curve (arising from a change in k) with a change brought by a shift in the curve.

The fluid approximation is also essential to deriving the formulas, since it glosses over details like TCP window adjustment and abort time distribution. Instead, we only work with averages (b_{TCP}, T_{abort}, etc.), as is often done in performance analysis [28,33]. These averages hide the Internet complexity, and provide simple closed-form expressions that make analysis tractable.

Prop. 1 (Section 6.9) illustrates how we can, starting from the equations here, mathematically elaborate on the informal analysis above. However, one should not lose sight of the accessibility objective.

We do not claim to have completely addressed the large issues mentioned in this paper (Q1–Q5, TCP modification, traffic monitoring, etc.), but we have demonstrated the technique's range and flexibility (Sections 6 and 7) by touching on these disparate issues, and by relating it to many results in the literature.

Acknowledgements

Jaeyeon Jung and Alex Yip did preliminary measurements that suggested evidence of user backoff; Yuan Li initiated the measurements and simulations reported here; Vidit Maheshwari illustrated the primal–dual formulation in Section 6.7, Arthur Berger, A. Kevin Tang, Chee-Wei Tan and the reviewers of the various incarnations of this paper gave many helpful comments—we thank them all. This work was supported in part by National University of Singapore under ARF Grant R-146-000-051-112.

References

[1] H. Abrahamsson, B. Ahlgren, Using empirical distributions to characterize web client traffic and to generate synthetic traffic, in: GLOBECOM, vol. 1, November 2000, pp. 428–433.
[2] A. Akella, S. Seshan, R. Karp, S. Shenker, C. Papadimitriou, Selfish behavior and stability of the internet: a game-theoretic analysis of TCP, in: SIGCOMM, 2002, pp. 117–130.
[3] E.J. Anderson, T.E. Anderson, On the stability of adaptive routing in the presence of congestion control, in: INFOCOM, 2003, pp. 948–958.
[4] M.F. Arlitt, C.L. Williamson, Internet web servers: workload characterization and performance implications, IEEE/ACM Transactions on Networking 5 (5) (1997) 631–645.
[5] P. Barford, M. Crovella, Generating representative web workloads for network and server performance evaluation, in: SIGMETRICS, 1998, pp. 151–160.
[6] T. Bonald, J.W. Roberts, Congestion at flow level and the impact of user behavior, Computer Networks 42 (2003) 521–536.
[7] T. Bu, Y. Liu, D.F. Towsley, On the TCP-friendliness of VoIP traffic, in: INFOCOM, 2006.
[8] T. Bu, D. Towsley, Fixed point approximations for TCP behavior in an AQM network, in: SIGMETRICS, 2001, pp. 216–225.
[9] E. Casilari, A. Reyes-Lecuona, F.J. González, A. Diaz-Estrella, F. Sandoval, Characterization of web traffic, in: GLOBECOM, 2001, pp. 1862–1866.
[10] C.S. Chang, Z. Liu, A bandwidth sharing theory for a large number of HTTP-like connections, IEEE/ACM Transactions on Networking 12 (5) (2004) 952–962.
[11] K.K. Cheng, K.T. Ko, S.W.C. Suen, Optimization of telephone networks in developing nations with example, in: TENCON, 1990, pp. 371–375.
[12] H.K. Choi, J.O. Limb, A behavioral model of Web traffic, in: ICNP, 1999, pp. 327–334.
[13] M.E. Crovella, A. Bestavros, Self-similarity in World Wide Web: evidence and possible causes, IEEE/ACM Transactions on Networking 5 (6) (1997) 835–846.
[14] A.C. Dalal, S. Jordan, Improving user-perceived performance at a World Wide Web server, in: GLOBECOM, 2001, pp. 2465–2469.
[15] A. Feldmann, BLT: bi-layer tracing of HTTP and TCP/IP, Computer Networks 33 (1–6) (2000) 321–335.
[16] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the Internet, IEEE/ACM Transactions on Networking 7 (4) (1999) 458–472.
[17] S.B. Fredj, T. Bonald, A. Proutiere, G. Régnié, J.W. Roberts, Statistical bandwidth sharing: a study of congestion at flow level, in: SIGCOMM, 2001, pp. 111–122.
[18] M. Garetto, M. Ajmone Marsan, R. Lo Cigno, M. Meo, On the use of fixed point approximations to study reliable protocols over congested links, in: GLOBECOM, 2003, pp. 3133–3137.

Please cite this article in press as: Y.C. Tay et al., Equilibrium analysis through separation of user and network behavior, Comput.Netw. (2008), doi:10.1016/j.comnet.2008.09.008

[19] H.C. Gromoll, P. Robert, B. Zwart, R. Bakker, The impact of reneging in processor sharing queues, in: SIGMETRICS/Performance, 2006, pp. 87–96.
[20] H. Hlavacs, G. Kotsis, Modeling user behavior: a layered approach, in: MASCOTS, 1999, pp. 218–225.
[21] N. Hohn, D. Veitch, K. Papagiannaki, C. Diot, Bridging router performance and queuing theory, in: SIGMETRICS, 2004, pp. 355–366.
[22] S. Iyer, S. Bhattacharyya, N. Taft, C. Diot, An approach to alleviate link overload as observed on an IP backbone, in: INFOCOM, 2003.
[23] R. Jain, The Art of Computer Systems Performance Analysis, John Wiley, New York, NY, 1991.
[24] F. Kelly, Mathematical modelling of the Internet, in: B. Engquist, W. Schmid (Eds.), Mathematics Unlimited – 2001 and Beyond, Springer-Verlag, 2001, pp. 685–702.
[25] P.B. Key, L. Massoulié, M. Vojnovic, Farsighted users harness network time-diversity, in: INFOCOM , 2005, pp. 2383–2394.
[26] L. Kleinrock, S.S. Lam, On stability of packet switching in a random multi-access broadcast channel, in: Hawaii Interenation Conference on System Sciences, 1976, pp. 74–77.
[27] K.C. Lan, J. Heidemann, Rapid model parameterization from traffic measurements, ACM Transactions on Modeling and Computer Simulation 12 (3) (2002) 201–229.
[28] S.H. Low, D.E. Lapsley, Optimization flow control, I: basic algorithm and convergence, IEEE/ACM Transactions on Networking 7 (6) (1999) 861–874.
[29] B.A. Mah, An empirical model of HTTP network traffic, in: INFOCOM, vol. 2, 1997, pp. 592–600.
[30] S. McCanne, S. Floyd, NS Notes and Documentation, <http://www.isi.edu/vint/nsnam/>.
[31] D.A. Menascé, V.A.F. Almeida, R. Fonseca, M.A. Mendes, A methodology for workload characterization of e-commerce sites, in: E-COMMERCE, 1999, pp. 119–128.
[32] D.P. Olshefski, J. Nieh, D. Agrawal, Inferring client response time at the web server, in: SIGMETRICS, 2002, pp. 160–171.
[33] J. Padhye, V. Firoiu, D.F. Towsley, J.F. Kurose, Modeling TCP Reno performance: a simple model and its empirical validation, IEEE/ACM Transactions on Networking 8 (2) (2000) 133–145.
[34] J. Postel, Transmission Control Protocol, IETF, RFC 793, September 1981.
[35] R.S. Prasad, C. Dovrolis, Measuring the congestion responsiveness of Internet traffic, in: PAM, 2007.
[36] D. Rossi, C. Casetti, M. Mellia, User patience and the Web: a hands-on investigation, in: GLOBECOM vol. 22, 2003, pp. 4163–4168.
[37] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble, H. Levy, An analysis of Internet content delivery systems, in: Usenix/ACM Symposium on Operating Systems Design and Implementation, 2002, pp. 315–327.
[38] B. Schroeder, A. Wierman, M. Harchol-Balter, Closed versus open system models: a cautionary tale, in: NSDI, 2006, pp. 239–252.
[39] D.N. Tran, W.T. Ooi, Y.C. Tay, SAX: a tool for studying congestion-induced surfer behavior, in: PAM (<http://www.pam2006.org/program.html/>), 2006.
[40] N. Vicari, S. Köhler, Measuring internet user traffic behavior dependent on access speed, in: ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, 2000.
[41] S. Yang, G. de Veciana, Bandwidth sharing: the role of user impatience, in: GLOBECOM, 2001, pp. 2258–2262.
[42] S.J. Yang, G. de Veciana, Enhancing both network and user performance for networks supporting best effort traffic, IEEE/ACM Transactions on Networking 12 (2) (2004) 349–360.

Y.C. Tay received his B.Sc. degree from the University of Singapore, and Ph.D. degree from Harvard University. He has a joint appointment with the Departments of Mathematics and Computer Science at NUS (http://www.comp.nus.edu.sg/~tayyc). His main research interest is performance modeling (database transactions, wireless protocols, cache misses, etc.). Other interests include distributed protocols and their correctness proofs.

Dinh Nguyen Tran received his B.Comp.(Hons) from School of Computing, National University of Singapore in 2005. He is now a graduate student at New York University.

Eric Yi Liu obtained his B.Comp.(Hons) from School of Computing, National University of Singapore in 2005. He was a research assistant in the NUS Department of Biological Sciences before joining the University of North Carolina at Chapel Hill as a graduate student in the Department of Computer Science.

Wei Tsang Ooi received the B.Sc. degree from the National University of Singapore and Ph.D. degree from Cornell University, Ithaca, NY. He is currently an Assistant Professor with the NUS Department of Computer Science, where he does research in multimedia systems, distributed systems and computer networking.

Robert Morris received the Ph.D. degree from Harvard University, Cambridge, MA, for work on modeling and controlling networks with large numbers of competing connections.

He is currently an Associate Professor with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, and a Member of the Computer Science and Artificial Intelligence Laboratory. As a graduate student, he helped design and build an ARPA-funded ATM switch with per-circuit hop-by-hop flow control. He led a mobile communication project which won a Best Student Paper Award from USENIX. He cofounded Viaweb, an e-commerce hosting service. His current interests include modular software-based routers, analysis of the aggregate behavior of Internet traffic, and scalable ad-hoc routing.

Please cite this article in press as: Y.C. Tay et al., Equilibrium analysis through separation of user and network behavior, Comput.Netw. (2008), doi:10.1016/j.comnet.2008.09.008