2013 IEEE 21st International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems

Resource Estimation for Network Virtualization through Users and Network Interaction Analysis

Bo-Chun Wang*, Y.C. Tay†, Leana Golubchik*
*Department of Computer Science, University of Southern California
Email: {bochunwa,leana}@usc.edu
†Department of Computer Science, National University of Singapore
Email: dcstayyc@nus.edu.sg

***Abstract*—Network virtualization can potentially overcome Internet ossification. This technology lets multiple virtual networks run on a shared physical infrastructure. A key step lies in mapping a virtual network request to a resource allocation in the network substrate.**

Previous approaches to this network embedding problem assumed the request will ask for specific resources, such as network capacity or computing power. However, the end-user is more interested in performance. This paper therefore considers a different request format, namely a request will ask for a certain quality of service (QoS). The infrastructure provider must then determine the resource allocation necessary for this QoS. In particular, the provider must take into account user reaction to perceived performance and adjust the allocation dynamically.

To this end, we propose an estimation mechanism that is based on analyzing the interaction between user behavior and network performance. This approach can dynamically adjust resource estimations when QoS requirements change. Our simulation-based experiments demonstrate that the proposed approach can satisfy user performance requirements through appropriate resource estimation. Moreover, our approach can adjust resource estimations efficiently and accurately.

***Index Terms*—Virtual Network Embedding; Resource Allocation; Network Virtualization; Traffic Equilibrium; Quality of Service.**

I. Introduction

Although the Internet has been a tremendous success in providing a variety of (packet-delivery based) services, the Internet's rapid growth and deployment have also become obstacles to modification of its current architecture and adoption of new protocols. Specifically, due to the coexistence of multiple ISPs, such modifications require their mutual agreement. However, given their conflicting policies and goals, ISP agreement is hard to achieve.

Network virtualization is viewed as a potential approach to overcome the ossification problem of the Internet [1]. In a network virtualization environment, several virtual networks (VNs) with heterogeneous resources can coexist over a shared physical infrastructure. Service providers can create their own VNs by leasing resources from infrastructure providers and offering customized end-to-end services without significant modifications to the physical infrastructure [2, 3]. That is, one advantage of network virtualization is that VNs can satisfy a variety of customized requirements.

When service providers generate VN requests, they can specify resources for every virtual node and link. Thus, when a *VN embedding* process maps a VN request onto specific physical nodes and links in the substrate network, it must result in satisfaction of constraints on virtual nodes and links.

Since substrate resources shared by VNs are finite, the VN embedding problem is one of the fundamental challenges in network virtualization. That is, an efficient VN embedding is necessary to increase resource utilization. However, the VN embedding problem has been proven to be NP-hard (in offline and online scenarios [4, 5]).

A. Current Approaches

Addressing this problem has been an active area of research, with a number of heuristics having been proposed, e.g., [6–15], where some of the works restrict the problem space in order to enable efficient heuristics and reduce complexity. These limitations include: (i) considering offline versions of the problem and assuming all virtual network requests are known in advance [9, 10, 12]; (ii) assuming that resources are unlimited without the need for admission control [7, 9, 12]; and (iii) focusing on specific topologies [9].

Fan et al. [7] focus on finding optimal reconfiguration policies that can minimize cost. Lu et al. [9] focus on a family of backbone-star topologies and aim at finding the best topology. Szeto et al. [10] use the multi-commodity flow algorithm to solve the VN embedding problem. The works in [6, 8, 11] do not restrict the problem space and consider link and node constraints together with admission control mechanisms. All of these works [6, 8, 11] consider the online version of the problem.

Yu et al. [11] propose a two stage mapping algorithm based on shortest path and multi-commodity flow algorithm.In addition, they allow path splitting and migration. Lischka and Karl [8] consider node and link mapping in one stage. Their approach is faster than the two-stage approach, especially for large virtual networks with high resource consumption.

Such works assume that VN requests indicate the exact amount of required resources (e.g., "the capacity of a link between two nodes should be 10 Mbps"). Then, heuristics are designed to achieve objectives defined from the perspective of *infrastructure providers*, e.g., balancing load [6, 12], maximizing revenue [6, 8, 11], and minimizing cost [6–9].

However, from the perspective of *service providers*, the main concern is having sufficient resources to support a certain

1526-7539/13 $26.00 © 2013 IEEE
DOI 10.1109/MASCOTS.2013.65

level of service quality. Existing efforts are not concerned with this, i.e., *they do not consider what resources are required to support a needed quality of service (QoS), which is an important consideration for service providers.*

B. Our Alternative Formulation

To this end, this paper proposes an alternative approach - namely that of *considering QoS as a constraint*. That is, when VN requests are made, service providers should be able to use the minimum required QoS as constraints of that request, rather than the amount of resources needed. It is typically not straightforward to determine the amount of resources that should be requested in order to satisfy a desired level of QoS. Thus, we reconsider the problem in this light.

There are three roles in virtual networks: infrastructure providers, service providers, and customers, where each role has different goals:

- *Infrastructure providers* typically focus on balancing the load, maximizing revenue, and minimizing cost.
- *Service providers* typically focus on maximizing revenue (e.g., by supporting as many customers or requests as possible); they are the intermediaries between customers and infrastructure providers.
- *Customers* typically focus on quality of service, such as downloading time (in file downloading applications) or bit-rate (in video viewing applications).

Previous efforts largely focus on infrastructure providers. Our work also includes a focus on service providers as well as customers. Specifically, in contrast to previous efforts, we consider and focus on the use of QoS as a constraint in the VN embedding problem.

In this paper, we consider two types of services, web and videos (e.g., YouTube [16]). These two types of traffic correspond to more than 60% of the Internet's traffic in North America [17]. As mentioned above, from the perspective of service providers, revenue maximization is an important goal. Several metrics can be used to evaluate revenue.

For example, service providers can measure how many customers they can support or how many requests (connections) can be completed over a certain time period. In this paper, we use connection completion rate as our QoS constraint, i.e., when generating a VN request, service providers would specify a desired connection completion rate.

Since connection completion rates are affected by network conditions and user behavior, service providers should consider these two factors when requesting resources for a specific connection completion rate. The difficulty with using connection completion rates for provisioning is that users react to QoS, so their demand becomes a moving target.

To model this user reaction, Tay et al. [18] decompose traffic equilibrium into user demand and network supply and then apply this model to web traffic. The user curve and network curve intersect to determine traffic equilibrium. We adopt their model and extend it to support video traffic by considering different user behavior characteristics.

C. Our Contribution

Our contributions in this paper can be summarized as follows. Unlike previous efforts, we focus on the use of QoS as a constraint in the VN embedding problem. We develop models and corresponding techniques that allow determination of resource amounts needed to satisfy QoS constraints.

In this paper, we adopt the model proposed in [18] and extend it to video traffic by considering different user behavior characteristics (see Sec.III-A).

Moreover, in Sec.III-B, we develop a corresponding mechanism to estimate traffic equilibrium when different link capacities are given. The mechanism allows infrastructure providers and service providers to determine the amount of resources needed to satisfy QoS constraints (e.g., connection completion rate) efficiently, particularly when user behavior and QoS constraints change dynamically.

In addition to connection completion rate, we also consider QoS metrics from the perspective of customers (e.g., bit-rate). We propose an algorithm which estimates the amount of resources needed by considering multiple QoS constraints simultaneously. In Sec.III-C, we demonstrate how our approach can be used by infrastructure providers by offering them insight into efficient resource mapping in a network virtualization environment.

Our extensive validation results demonstrate that our estimation mechanism is accurate (see Sec.IV). Our proposed approach is orthogonal to (and can be combined with) existing efforts (see Sec.V).

II. Background

In this paper, we adopt the traffic equilibrium analysis model for web traffic proposed in [18]. For completeness of presentation, we first provide a summary of background information needed in the remainder of this paper; for clarity of presentation, several figures are reproduced here (the corresponding figures in [18] are noted accordingly). We present our proposed model in Sec.III.

Tay et al. consider traffic equilibrium as a balance between an inflow controlled by users and an outflow controlled by the network (e.g., link capacity, congestion control, etc) [18]. That is, the number of active connections is controlled by users, where the network conditions affect how fast a connection can be completed. Since users react to congestion, the interaction between users and the network form a loop:

1) TCP's control mechanism reduces congestion window due to network congestion.
2) The reduced congestion window causes downloading time to increase, so users may generate fewer connections or abort connections.
3) User reaction reduces the number of active connections and, consequently, TCP increases transfer rate per connection.
4) The increased downloading rate encourages users to launch more connections, thus causing congestion to increase, looping back to 1 (above).

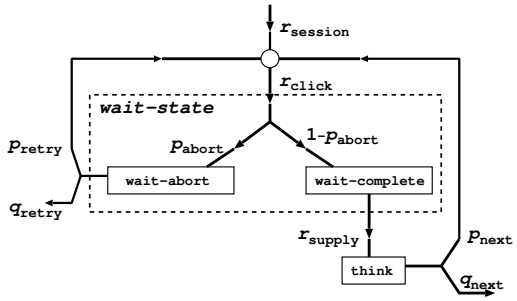


Fig. 1. **Surfing session model (corresponds to Fig.3 in [18]).**

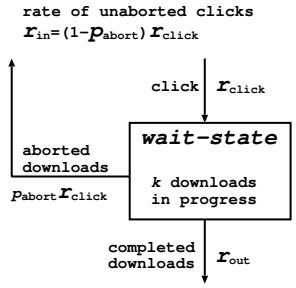


Fig. 2. **Flow of downloads (corresponds to Fig.4 in [18]).**

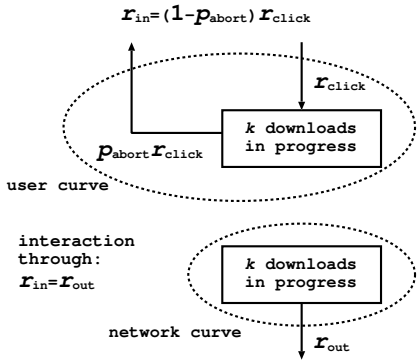


Fig. 3. **Two submodels: a user curve and a network curve (corresponds to Fig.5 in [18]).**

This loop makes it hard to reason about the shift in an equilibrium when there is a flash crowd, or some link breaks.

Tay et al. use a surfing session model for user behavior, as depicted in Fig.1; it is assumed that the session arrival rate, $r_{session}$, is independent of network congestion because users are unaware of network conditions until they arrive.

In each session, a user generates requests, e.g., by clicking on hyperlinks, buttons, etc.[1] r_{click} is defined as the click rate. Each click may launch multiple responses, and the traffic sent to the user is termed a **download**. For simplicity, we use the terms download and connection interchangeably in what follows.

After a click, a user waits for download completion (`wait-state` is Fig.1). When the user enters the wait-state, two different cases are possible, `wait-abort` state and `wait-complete` state, i.e., if the downloading time is too long due to network congestion, the user may decide to abort the download. The probability of aborting a download is p_{abort}. After the user aborts a download, it may retry again with probability p_{retry}. Otherwise, it quits the session with probability $q_{retry} = 1 - p_{retry}$. If the user completes the download, it enters the `think` state (when viewing downloaded content). Then, it may click (generate another request) with probability p_{next} or finish the session with probability $q_{next} = 1 - p_{next}$.

Focusing on the wait-state (in Fig.2), k is defined as the average number of ongoing concurrent downloads in the wait-state. This k is a measure of network congestion.

The wait-state is decomposed into a $user - network$ model (depicted in Fig.3). This model includes a user demand curve r_{in} and a network supply curve r_{out}. The user curve represents the relationship between the unaborted click rate and the congestion level k, and the network curve describes the relationship between the rate of completed downloads and k. The decomposition in Fig.3 gives

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

and

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

where T_{abort}, b_{TCP}, and $S_{completed}$ are the average time spent in the `wait-abort` state, the average bandwidth provided by TCP for a download, and the average size of a completed download, respectively. These two equations describe a pair of user and network curves that determine the traffic equilibrium where they intersect (see Fig. 5(a)).

Analyzing how a flash crowd affects the equilibrium thus reduces to examining how the user curve is affected. Similarly, analyzing the impace of a link failure reduces to examining how the network curve moves. We thus break the feedback loop illustrated earlier for TCP.

III. Proposed Mechanism

Below, we extend the traffic equilibrium analysis model to video traffic (see Sec.III-A). Moreover, based on the traffic equilibrium analysis model, we design a mechanism to estimate the amount of resources needed to satisfy QoS constraints (see Sec.III-B). We then demonstrate how infrastructure providers can use our mechanism (see Sec.III-C). We present the evaluation of our mechanism in Sec.IV.

A. Traffic Equilibrium Analysis Model Extension

The work presented in [18] only considers web traffic. We now extend the model to videos by considering different user behavior characteristics. In this paper, we use YouTube [16] as an example of video traffic. YouTube provides video sharing services and allows users to upload videos of up to 15 minutes

[1]Typing in a URL is also regarded as a click.

in length.[2] We also describe how to apply our model to long videos (e.g., movies).

In web services, a user aborts a download mostly because it is slow. In video sharing services, however, a user may abort a download for other reasons. Gill et al. have studied the YouTube system and found that approximately 24% of video downloads were interrupted [19]. They argued that there were two main reasons for users to abort connections before the video ended: (1) poor performance due to slow downloading rate (i.e., as in web services); (2) poor content quality (e.g., video content is uninteresting, or video resolution is low).

Because user behavior is different in video sharing services, the model described above cannot be applied directly, and the two cases described above need to be reconsidered. We define p_{rate} as the probability that the downloading rate of a connection is too slow and $p_{quality}$ as the probability that users abort a connection because of poor content quality, even if the downloading rate is fast. Then, p_{abort}^{video} in video sharing services can be given as:

$$p_{abort}^{video} = p_{rate} + (1 - p_{rate})p_{quality} \tag{3}$$

This equation gives the user curve a shape that is different from the one for web traffic (compare Figs. 5(a) and 5(b)).

We assume p_{retry} is the same, whether users abort a connection due to having a poor rate or poor quality. Therefore, we can simply replace p_{abort} in Eq.1 by p_{abort}^{video}, which can be calculated using Eq.3. Then, the equation for the user curve (r_{in}) for web services (Eq.1) still works for video sharing services.

We also define T_{rate} as the average amount of time users wait before aborting a connection because of slow downloading rate, and $T_{quality}$ as the average amount of time users wait before aborting a connection because of poor content quality. By Little's Law, the value of k is

$$\begin{aligned} k =& [p_{rate}T_{rate} + (1 - p_{rate})p_{quality}T_{quality} \\ &+ (1 - p_{abort}^{video})t_{completed}]r_{click}, \end{aligned}$$

where $t_{completed}$ is the average amount of time spent in the `wait-complete` state. Then, the equation for r_{click} in video sharing services is:

$$r_{click} = \frac{k}{p_{rate}T_{rate} + (1-p_{rate})p_{quality}T_{quality} + (1-p_{abort}^{video})t_{completed}}$$

and the equation for the network curve, r_{out}, is:

$$\begin{aligned} r_{out} &= (1 - p_{abort}^{video})r_{click} \\ &= \frac{k}{\frac{p_{rate}}{1-p_{abort}^{video}}T_{rate} + \frac{(1-p_{rate})p_{quality}}{1-p_{abort}^{video}}T_{quality} + \frac{S_{completed}}{b_{TCP}}}, \end{aligned} \tag{4}$$

where $t_{completed} = \frac{S_{completed}}{b_{TCP}}$.

Again, this equation gives the network curve a shape that is different from that for web services (see Fig. 5).

The above assumes that users will view the entire video, if they do not abort a video download. Since more than 75% of videos on YouTube are within 300 seconds [20], we believe that this assumption is reasonable. However, this assumption may not be valid when considering Video-on-Demand (VoD) services that provide long videos, such as movies.

Yu et al. have studied user behavior in VoD systems [21], where a typical file is roughly 100 minutes in length. They found that more than 75% of sessions were terminated within 25 minutes. Most of the users just scanned through videos rather than watched the entire movie. In such a case, we can use a threshold to define what is a completed connection. If a connection is killed after this threshold is reached, then we consider this kill as not an abort. The threshold can be defined in terms of time, fraction of video length, etc.

Moreover, compared to short video sharing systems, user behavior in VoD type systems is more complicated, e.g., users may seek forward or backward while viewing a movie [22]. Such user behavior may change time spent viewing a video. In our model, the time users spend viewing a video is required information. Thus, given such information, our model can be applied to long videos.

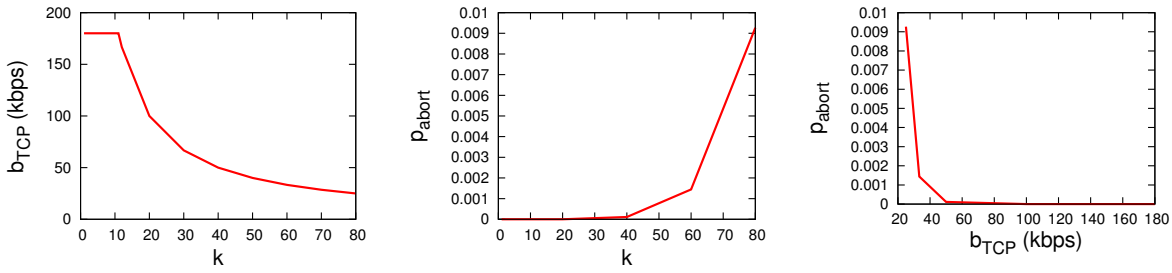
An important consideration in this paper is that users do react to network congestion. It is a reasonable model for web and video traffic when users are monitoring their session's progress. However, when users download a large file, such monitoring may not occur (they may take a break, do something else, etc). Tay et al. have studied such non-reactive connections and found that such traffic may cause a loss of equilibrium and induce a performance collapse [18]. We therefore assume there is a separate mechanism (e.g., admission control) for dealing with non-reactive flows, and focus only on reactive connections in this paper.

B. Performance Estimation Mechanism

In this paper, we consider service providers as well as customers. We have described the traffic equilibrium analysis model for web and video traffic above. This model is based on a $user - network$ model that separates user and network behavior into two curves, and the traffic equilibrium occurs where these two curves intersect. We use connection completion rate at the equilibrium point as a QoS metric from the perspective of service providers. In Section III-B1, we develop a mechanism to estimate the amount of resources needed to satisfy such a QoS constraint. Then, in Section III-B2, we propose an algorithm for estimating the amount of resources needed when different QoS constraints are requested.

1) Connection Completion Rate: To calculate the connection completion rate, we would need to know user behavior characteristics and network conditions. For example, user arrival rate in the daytime is typically higher than at night. Thus, to maintain the same QoS, infrastructure providers should allocate more capacity in the daytime. Moreover, service providers may make requests for higher QoS. For example, when service providers release new services or new videos,

[2]By default, YouTube allows users to upload videos that are 15 minutes long. If users want to upload videos that are longer than 15 minutes, they have to verify their accounts. In this paper, we use the default setting.



(a) b_{TCP} vs. k (b) p_{abort} vs. k (c) p_{abort} vs. b_{TCP}

Fig. 4. **Relationship between average TCP connection bandwidth b_{TCP}, average number of concurrent downloads k, and p_{abort}**

they may expect a higher connection load from customers. If the necessary information is not available (e.g., initially), traffic conditions need to be measured. However, since network equilibrium changes with time, it is not efficient to measure traffic conditions continuously.

One possible approach for infrastructure providers to cater to changing demands of customers and service providers is to increase capacity gradually and then check whether the assigned resources can satisfy the QoS requirements. Although straightforward, this method is inefficient and slow. We therefore use our demand/supply model to develop a mechanism that can estimate the amount of necessary capacity based on measurement data collected by infrastructure providers.

We note that each VN can run its own services. Because user behavior differs from service to service, we assume that the data used for estimation purposes must be collected from the same type of service, e.g., data collected from web services should not be applied to video services. Since, in their basic construction, the models presented above are similar, we use the model for web traffic as an example to demonstrate the proposed mechanism.

Recall that the traffic equilibrium is a balance between an inflow controlled by users (Eq.1) and an outflow due to the network (Eq.2). We can classify parameters in Eq.1 and Eq.2 into three categories based on what causes them to change:

- user: p_{abort}, $r_{session}$, p_{retry}, p_{next}, T_{abort}, k
- network: b_{TCP}, k
- application: $S_{completed}$

Because data are collected from the same service, we can assume that $S_{completed}$ and T_{abort} are unchanged[3]. Since users do not know network conditions before they arrive, we can assume that $r_{session}$ remains constant over a time window. The analysis of collected traces in [23] indicates that p_{retry} is fairly constant and that p_{next} can be represented as a function of p_{abort}. Therefore, our proposed mechanism focuses on the relationship between p_{abort}, k, and b_{TCP}.

The first step is to determine the relationship between k and b_{TCP}. Recall that b_{TCP} is the average bandwidth provided by TCP for a download, and k is the number of concurrent connections.When k is 1, b_{TCP} will be the throughput of that connection. Altman et al. [24] and Barakat et al. [25] have studied TCP throughput in single-hop and multiple-hop paths, respectively. Thus, we can calculate TCP throughput of a connection based on the path type.

However, the TCP throughput calculated by [24] or [25] is an ideal value. Reaching the ideal value requires a large enough maximum congestion window size. If the maximum congestion window size is small, we can estimate TCP throughput by using $\frac{W_m}{RTT}$, where W_m is the maximum congestion window size and RTT is the round trip time [26]. We define $thrp^{ideal}$ as the TCP throughput calculated by models proposed in [24] and [25]. We can then have the following equation of TCP throughput, $thrp$:

$$thrp = min(thrp^{ideal}, \frac{W_m}{RTT})$$

When k increases, the value of b_{TCP} is determined by the capacity C of the virtual link. We therefore have

$$b_{TCP} = \begin{cases} thrp & \text{, if } thrp * k \leq C \\ \frac{C}{k} & \text{, if } thrp * k > C \end{cases} \quad (5)$$

Fig.4(a) illustrates Eq.5, using data from ns2 simulations [27]. In this example, C is 2 Mbps, and users abort a connection if the connection cannot be completed within 20 seconds.

Users will react to network congestion: When b_{TCP} decreases, p_{abort} will increase. It is difficult to determine their relationship directly. Since Eq. 5 already relates b_{TCP} to k, we can relate p_{abort} to b_{TCP} through k instead. There are tools (e.g., [23]) that can extract information from traffic measurements for expressing p_{abort} in terms of k. Fig.4(b) illustrates such a relationship using simulation data.

Given these relationships (b_{TCP}-k and p_{abort}-k), we can determine the value of p_{abort} for different b_{TCP}. Fig.4(c), derived from Figs. 4(a) and 4(b), is an example illustration of the relationship between p_{abort} and b_{TCP}. We can think of this relationship as representing how users react to network congestion. When a connection can be completed with a reasonable downloading time, users will not abort the connection. However, when network congestion becomes worse and b_{TCP} becomes small, users may become impatient when waiting for

[3]The change in T_{abort} (depending on time of day) is a slow change that can be measured leisurely (e.g. offline). Within the time granularity of a connection, T_{abort} can be considered constant. Value of content can determine abort time; this would make abort time a random variable, where we would use the average value, T_{abort}, of this random variable.

Algorithm 1 Calculation of the minimum amount of capacity needed to satisfy multiple QoS constraints simultaneously.

QoS Requirement 1 (R1): connection completion rate
QoS Requirement 2 (R2): bit-rate
//$R1$ and $R2$ are evaluated by intersecting the
//user curve (Eq.1) and network curve (Eq.4); see Fig. 5
//assumes there is a feasible solution for capacity
$C_{low} \leftarrow C_{min}$
$C_{high} \leftarrow 2 * C_{min}$
while true **do**
  if $R1(C_{low})$ AND $R2(C_{low})$ **then**
    return C_{low}
  else if $R1(C_{high})$ AND $R2(C_{high})$ **then**
    if $\frac{C_{high}-C_{low}}{C_{low}} \leq \delta$ **then**
      return C_{high}
    end if
    $C_{temp} \leftarrow \frac{C_{high}+C_{low}}{2}$
    if $R1(C_{temp})$ AND $R2(C_{temp})$ **then**
      $C_{high} \leftarrow C_{temp}$
    else
      $C_{low} \leftarrow C_{temp}$
    end if
  else
    $C_{low} \leftarrow C_{high}$
    $C_{high} \leftarrow 2 * C_{high}$
  end if
end while

connection completion. From Fig.4(c), we can see that p_{abort} increases dramatically when b_{TCP} decreases.

After determining the relationship between b_{TCP} and p_{abort} based on existing data from earlier measurements, we can use it to estimate QoS in an offline manner. For example, when service providers ask for higher QoS, infrastructure providers can use our mechanism to estimate the corresponding amount of resources needed in an offline manner. They can select a new capacity of the virtual link, C, and calculate the corresponding b_{TCP} and p_{abort} based on existing data from earlier measurements. They can then plot the user and network curves, using Eq.1 and Eq.2; the intersection point (Fig.5) gives the connection completion rate.

Next, they can change the value of C iteratively, until they find the new traffic equilibrium that satisfies the QoS requirement. Resources can then be assigned to service providers based on estimated results. This approach allows infrastructure providers to avoid having to take more of an online approach (in contrast to the offline approach described above), i.e., they can avoid having to increase resources gradually and measure traffic continuously.

2) Multiple QoS Constraints: Above, we gave an approach for estimating the amount of resources needed to satisfy connection completion rate requirements. However, from the perspective of customers, different QoS metrics may be of interest, such as downloading time (in file downloading applications) or bit-rate (in video viewing applications). In order to attract customers, service providers need to consider multiple QoS constraints simultaneously.

As an example, we use video services and consider two QoS constraints, connection completion rate and bit-rate, from the perspective of service provider and customers, respectively. We design an algorithm to calculate the amount of resources needed to satisfy multiple QoS constraints.

The equation of the network curve for video, r_{out}, is derived in Eq.4. Initially, we assume there is no user reactions and all probabilities in Eq.4 are 0. In such a case, the equation of the network curve is $r_{out} = \frac{k}{\frac{S_{completed}}{b_{TCP}}}$. Then, we replace b_{TCP} by $\frac{C}{k}$ where C is the capacity of a link, and the equation of the network curve is $r_{out} = \frac{C}{S_{completed}}$.

If the connection completion rate requested by service providers is r_{rate}, the amount of resources needed should be at least C_{min} where $C_{min} = r_{rate} * S_{completed}$. Then, we can use Algo.1 to calculate the amount of resources, $C_{required}$, that is needed to satisfy multiple QoS constraints (file throughput and the video bit rate). The idea here is to set bounds C_{low} and C_{high} for link capacity and iteratively adjust them, to converge on a value C that satisfies the QoS requirements[4].

In Algo.1, Eq.1 and Eq.4 are used to determine the user and network curves; the intersection point (Fig.5) gives the connection completion rate (which corresponds to $R1$ in Algo.1), while the equilibrium k value determines b_{TCP} (Eq.5) (which corresponds to $R2$ in Algo.1).

C. Virtual Network Embedding

We presented the traffic equilibrium analysis model in Sec.II and Sec.III-A and proposed our estimation mechanism for the amount of resources in Sec.III-B. In this section, we explain in detail how infrastructure providers can use our mechanism to calculate the amount of resources needed.

Here, service providers can ask for a certain QoS (e.g. connection completion rate) instead of the amount of resources (e.g. capacity) in virtual network requests. Infrastructure providers need to estimate the amount of resources which should be assigned to satisfy the QoS specifications. Moreover, the infrastructure providers should also be able to modify resource allocations dynamically. Our mechanism allows infrastructure providers to efficiently estimate the amount of capacity necessary to satisfy such requests, based on existing data from earlier measurements.

However, there is a requirement for our resource estimation mechanism: infrastructure providers should have measured p_{abort}, $r_{session}$, p_{retry}, p_{next}, T_{abort}, b_{TCP}, k, and $S_{completed}$ for a particular application. If infrastructure providers do not have such data, they cannot use the estimation mechanism proposed in Sec.III-B. To address this issue, we propose an iterative approach.

Suppose the service provider specifies a value for connection completion rate r_{rate}, and the infrastructure provider must

[4]In Algo.1, we assume there is a feasible solution for capacity. If a feasible solution does not exist, infrastructure providers would reject this request.

determine the capacity allocation C for the corresponding virtual link. The infrastructure provider has to collect data to compute $r_{session}$, b_{TCP}, k, and $S_{completed}$. Since they do not have data on user behavior at the beginning of this process, they initially set all probabilities to 0, and use a lower bound $C = r_{rate} * S_{completed}$, as in Sec.III-B2. After collecting sufficient data for p_{abort}, T_{abort}, etc., the infrastructure provider uses collected data to calculate the user and network curves and adjust C accordingly, as in Algo.1. This can be repeated if better estimates of user behavior data become available.

IV. Evaluation

A. Evaluation Environment

We use measurement results from [28] to generate a substrate network topology. In [28], Spring et al. designed Rocketfuel, a mapping engine, to measure router-level ISP topologies. We use the Tiscali topology of the Rocketfuel's trace. The topology has 276 nodes and around 400 links, a scale that corresponds to a medium-size ISP. The link capacities follow a uniform distribution from 40Mbps to 200Mbps.

We consider two services, web and video. For web services, each download transfers thirty 536-byte packets, and T_{abort} is 20 seconds. The settings are the same as used in [18]. For video services, the file size of a video is 10 MBytes. T_{rate} and $T_{quality}$ are 6 minutes and 30 seconds, respectively. These settings are based on measurement results in [19].

We use the ns-2 simulator [27] to evaluate the network curve. We determine p_{abort} from the network curve simulation. We randomly select two end nodes from the Tiscali topology to form a path. We consider the path as a virtual link between these two nodes. Then, we use dumbbell configurations with k sources sending data to k destinations over this virtual link, where the k sources are attached to one end node, and the k destinations are attached to the other node. Although there is only one topology in the experiments, traffic is generated over multiple virtual links, with each link having its own topology.

To maintain k concurrent connections, all sources are in busy states all the time. After we determine p_{abort}, the user curve (Eq.1) can be plotted as follows: In [18], collected traces are analyzed and p_{retry} is found to be close to 0.97. We can then use p_{abort} determined from the network curve simulation and do a linear regression fit to determine p_{next}.

We give examples of user/network curves for web and video traffic in Fig.5(a) and Fig.5(b), respectively. In Fig.5(a), the $r_{session}$ is fixed at 0.4, and we use two capacity setting for web traffic: 1Mbps and 2Mbps. Fig.5(a) demonstrates that different capacities result in different connection completion rates when the network traffic reaches equilibrium. As expected, the connection completion rate increases as the capacity increases.

In Fig.5(b), the capacity of a virtual link for video traffic is 30Mbps. We use two $r_{session}$ values: 0.02 and 0.03. In this example, when $r_{session}$ increases, the connection completion rate at equilibrium *decreases*. Therefore, to satisfy the same QoS requirement, infrastructure providers should increase the capacity. Fig.5 demonstrates that network conditions (e.g., capacity) and user behavior (e.g., $r_{session}$) affect the network equilibrium.

B. Accuracy of Performance Estimation Mechanism

We presented our performance estimation mechanism in Sec.III-B. Here, we evaluate its accuracy. As mentioned above, measurement data is necessary to carry out the estimation mechanism. We collect data from ns2. The capacity for web traffic and for video traffic is 1Mbps and 20Mbps, respectively. We use this data as measurement data in the estimation mechanism.

First, we fix the value of $r_{session}$ for web and video traffic and use different connection completion rate requirements to simulate service providers changing QoS requirements. For web service, the maximum r_{out} value is about 8, so we pick a QoS value uniformly at random between 4 and 80 for the connection completion rate; for video service, the maximum r_{out} value is about 0.25, and we similarly pick a QoS value between 0.2 and 2.

To satisfy this changing QoS, an infrastructure provider should be able to allocate capacity dynamically. When a connection completion rate is given, we determine the estimated capacity, C^{model}, using our mechanism; we use ns2 to determine the actual amount of capacity needed, C^{sim}. Then, we compare C^{model} with C^{sim} and calculate the error rate = $|\frac{C^{sim}-C^{model}}{C^{sim}}|$. The average error rate is presented in Fig.6.

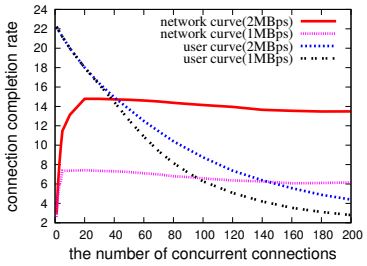
We then fix the connection completion rate (QoS) requirement and change the values of $r_{session}$. This scenario simulates the case where the user arrival rate changes dynamically. For web service, $r_{session}$ is selected uniformly at random between 0.2 and 2; for video service, $r_{session}$ is similarly chosen between 0.01 and 0.1. The infrastructure provider has to adjust resource allocation to satisfy the fixed QoS requirement despite the changing $r_{session}$ We calculate the error rate and depict the results in Fig.6.

Fig.6 shows that our mechanism can estimate the amount of resources needed accurately in both scenarios. In this evaluation, we only use measurement data corresponding to a particular capacity for each service. If infrastructure or service providers have multiple sets of measurement data (e.g., data measured from different bandwidth capacities), they should be able to improve accuracy.

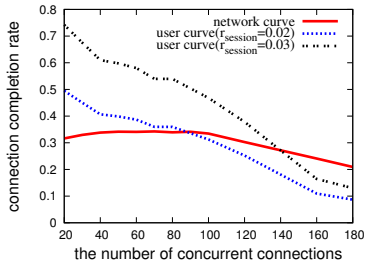
C. Robustness: Effect of Link Delay and buffer Sizes

Infrastructure providers use our mechanism to estimate the amount of resources needed. Till now, the main link information used in this paper is link capacity. Since the data measurements can be collected from different links, we need to consider the different possible settings of each link (e.g., RTT and buffer sizes). Here, we evaluate the impact of link delay and buffer sizes on our performance estimation mechanism.

We use web traffic as an example. The default settings are: capacity of 1Mbps, link delay of 20ms, and buffer size of 50. We increase link delay to 40ms in the first experiment, and we increase buffer size to 100 in the second experiment. Then,



(a) The network curve and the user curve in web services.



(b) The network curve and the user curve in video services.

Fig. 5. **The connection completion rate at equilibrium is where the user demand (Eq.1) and the network supply (Eq.2) curves intersect. Notice how the curves differ in shape between (a) and (b).**

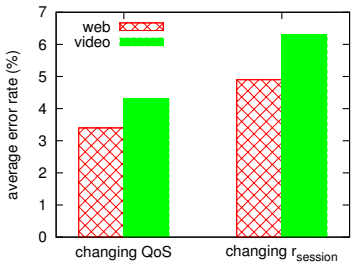


Fig. 6. **The average error rate for changing QoS and changing $r_{session}$**

we observe the network equilibrium under different settings. The results are depicted in Fig.7 where $r_{session}$ is 0.4.

Fig.7(a) shows that link delay affects the network curve when the number of concurrent connections is small, because, in this case, b_{TCP} is dominated by RTT (Eq.5); when the number of concurrent connections is large enough, b_{TCP} is determined by bandwidth capacity, rather than RTT. Moreover, the user curves are almost the same and are not affected by link delay. Thus, variability in link delays does not affect the network equilibrium significantly.

Fig.7(b) shows that buffer sizes affect the network curve when the number of concurrent connections is large, because packets experience longer queuing delays. When congestion starts, long queuing delays make the situation worse, so the network curve decreases more rapidly. Nonetheless, the connection completion rate at equilibrium is robust with respect to congestion level.

To evaluate the effect of link delay and buffer sizes on our mechanism, we randomly select path i from our topology and collect traffic data from this path. Then, we randomly select 50 paths which have different link delays and buffer sizes and assign different QoS requirements to each of them. We use the data collected from path i to estimate the amount of resources needed for these 50 paths. We also collect traffic data from the 50 paths. We compare the estimation results based on data collected from path i with the actual measurement results from the 50 paths. The resulting average error rate is less than 10% (in Fig. 8).

In summary, the results demonstrate the robustness of our mechanism with respect to variability in link delays and buffer sizes.

D. Heterogeneous User Behavior and File Sizes

In the experiments above, we use homogeneous user behavior and file sizes. The values of T_{abort}, T_{rate}, $T_{quality}$, and file size are fixed. However, in real systems, user behavior and file sizes are heterogeneous. Here, we evaluate the accuracy of using average values in estimating performance under heterogeneous environments.

We now describe the settings of the homogeneous and the heterogeneous environments.

- **Homogeneous:** For web services, each download transfers thirty 536-byte packets, and T_{abort} is 20 seconds. For video services, the average file size of a video is 10 MBytes. T_{rate} and $T_{quality}$ are 6 minutes and 30 seconds, respectively.
- **Heterogeneous:** For web services, the number of packets each download transfers follows a uniform distribution from 25 to 35. The size of each packet is 536 bytes. T_{abort} follows a uniform distribution from 15 seconds to 25 seconds. For video services, the video file size follows a uniform distribution from 5 MBytes to 15 MBytes. T_{rate} and $T_{quality}$ also follow uniform distributions, where T_{rate} varies from 5 minutes to 7 minutes, and $T_{quality}$ varies from 20 seconds to 40 seconds.

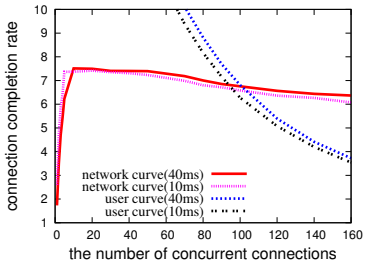
We give examples for each traffic type in Fig.9(a) and Fig.9(b). The capacity for web and video traffic is 1 Mbps and 30 Mbps, respectively. From Fig.9, we can observe that the performance of these two settings is almost the same when the number of concurrent connections is small. When the number of concurrent connections becomes large, the performance of the heterogeneous setting is worse than the performance of the homogeneous setting. This is not unexpected, as the variability in the heterogeneous setting (as compared to the homogeneous one) likely results in degraded performance.

We then use the same settings used in Sec.IV-B to verify the accuracy of our mechanism. We vary the connection completion rate (QoS) requirement and the values of $r_{session}$. The results are shown in Fig.10.

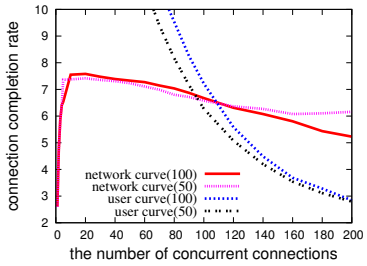
Compared to Fig.6, the error rate of using average values to estimate performance under heterogeneous settings is higher. However, the error rate increases only by about 2%. The results demonstrate that our mechanism still can estimate the amount of resources needed accurately (with a less than 10% error) in the homogeneous and the heterogeneous environments.

V. Discussion

Resource Estimation at Service Providers: Here, we discuss how service providers can use our mechanism. Previous work assumes that service providers will ask for a specific amount of resources when they send infrastructure providers



(a) The network and user curves with different link delay



(b) The network and user curves with different buffer sizes

Fig. 7. **The equilibrium and our estimation mechanism are robust with respect to uncertainty over link delay and buffer sizes.**

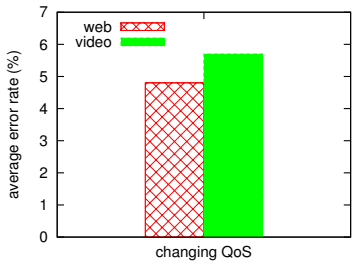
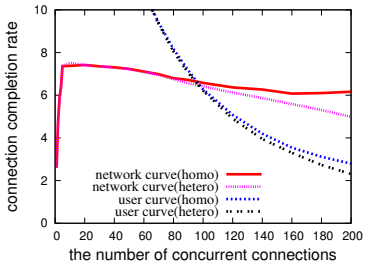
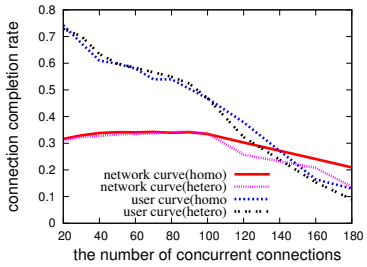


Fig. 8. **The average error rate for changing QoS with different link delay and buffer sizes**



(a) The network curve and the user curve in web services.



(b) The network curve and the user curve in video services.

Fig. 9. **Comparison between homogeneous settings and heterogeneous settings.**

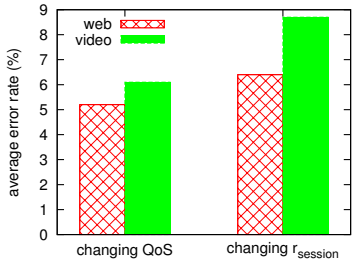


Fig. 10. **The average error rate for changing QoS and changing $r_{session}$**

a virtual network request. However, previous work *does not explain how to calculate the amount of resources needed and does not consider how to satisfy QoS requirements*. Thus, our approach can complement previous efforts. Specifically, when service providers want to create a new virtual network, they can define the minimum QoS which should be achieved. They can then calculate the required capacity allocation. Then, they can send infrastructure providers VN requests and ask for that resource amount. Infrastructure providers can then use other approaches (e.g., [6, 8, 11]) to assign resources.

However, there is a limitation when service providers use our mechanism. In our mechanism, b_{TCP} equals TCP throughput of an isolated connection when the number of concurrent connections is low (Eq.5). There are several factors that can affect TCP throughput, such as RTT and buffer sizes. Since service providers do not know which physical paths will be assigned, the only information they have is the capacity of the virtual link. Other network information (e.g., RTT and buffer sizes) is typically unavailable. Therefore, in our mechanism, the equation used by service providers to calculate b_{TCP} is $\frac{C}{k}$. This estimation of b_{TCP} does not affect our mechanism significantly. Service providers can monitor QoS metrics. If service providers realize that their QoS isn't being satisfied, they can adjust their request for resources, made to infrastructure providers.

Network or application configurations: In this paper, we do not focus on any particular configuration for specific services or customized protocols. However, today, some service providers tune TCP to improve their performance. For instance, Google modifies TCP on their routers for better performance. Some applications may open multiple connections to reduce downloading time. Moreover, Dobrian et al. find that buffering algorithms in video services have an impact on user engagement [29]. These configurations may change network equilibrium by affecting the network curve or the user curve. Since our approach focuses on a generic case, such issues are outside the scope of this paper and are part of future efforts.

Combining with existing efforts: A number of heuristics have been proposed to address the VN embedding problem. Such work assumes that VN requests indicate the exact amount of resources required. Our approach can be combined with these efforts to estimate the amount of required resources.

In [11], Yu et al. propose two mechanisms to simplify virtual network embedding: i) split a virtual link over multiple substrate paths and ii) path migration. These mechanisms have also been considered in [8] and [6].

However, there is no conflict between our approach and path splitting and migration mechanisms, and our approach can also be used together with these mechanisms. We described how to use our mechanism at the infrastructure provider side in Sec.III-C and at the service provider side above. If path splitting and migration are allowed, our mechanism can be used by the infrastructure provider only.

We believe that service providers should ask for specific QoS requirements, rather than for specific capacity. The rational for this is that the sum of QoS metrics obtained from split paths may not be equal to the QoS achieved on a single path. Because service providers do not know whether

infrastructure providers will split paths or not, it is difficult for service providers to ask for appropriate amounts of resources to achieve their QoS goals.

Therefore, service providers should send infrastructure providers virtual network requests with QoS requirements instead. The goal of infrastructure providers would then be to make sure that the QoS achieved over all paths can satisfy these QoS requirements.

VI. Conclusions

We introduced an alternative direction for VN requests, namely that of using QoS as constraints. In contrast to previous efforts that assume a VN request will indicate the amount of resources needed, we suggested that service providers can use QoS as constraints when generating VN requests.

We focused on two popular services, web and videos, and proposed an allocation mechanism based on analysis of interaction between user behavior and network performance in different services. This mechanism can be used to estimate the amount of resources needed. It can also dynamically adjust resource allocations.

Our simulation-based experiments demonstrate that the mechanism can satisfy QoS requirements through appropriate resource allocation. Moreover, our approach can adjust resource allocations efficiently and accurately.

References

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *IEEE Computer*, April 2005.

[2] J. Turner and D. Taylor, "Diversifying the internet," in *IEEE GLOBECOM*, 2005.

[3] N. Feamster, L. Gao, and J. Rexford, "How to lease the internet in your spare time," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 61–64, Jan. 2007.

[4] D. G. Andersen, "Theoretical approaches to node assignment," Dec. 2002, unpublished Manuscript.

[5] S. G. Kolliopoulos and C. Stein, "Improved approximation algorithms for unsplittable flow problems," in *FOCS*, 1997.

[6] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *IEEE INFOCOM*, 2009.

[7] J. Fan and M. H. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies," in *IEEE INFOCOM*, 2006.

[8] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *VISA*, 2009.

[9] J. Lu and J. Turner, "Efficient mapping of virtual networks onto a shared substrate," *Washington University. Technical Report*, 2006.

[10] W. Szeto, Y. Iraqi, and R. Boutaba, "A multi-commodity flow based approach to virtual network resource allocation," in *IEEE GLOBECOM*, 2003.

[11] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, April 2008.

[12] Y. Zhu and M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *IEEE INFOCOM*, 2006.

[13] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *SIGCOMM Comput. Commun. Rev.*, April 2011.

[14] S. Zhang, Z. Qian, J. Wu, and S. Lu, "An opportunistic resource sharing and topology-aware mapping framework for virtual networks," in *IEEE INFOCOM*, 2012.

[15] A. Fischer, J. Botero, M. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. PP, no. 99, pp. 1–19, 2013.

[16] *YouTube*, http://www.youtube.com/.

[17] *Sandvine Global Internet Phenomena Report - Spring 2011*, Sandvine, http://www.sandvine.com/.

[18] Y. C. Tay, D. N. Tran, E. Y. Liu, W. T. Ooi, and R. Morris, "Equilibrium analysis through separation of user and network behavior," *Comput. Netw.*, vol. 52, pp. 3405–3420, Dec. 2008.

[19] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in *IMC*, 2007.

[20] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of YouTube videos," in *IEEE IWQoS*, Jun 2008.

[21] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," in *EuroSys*, 2006.

[22] B. Cheng, X. Liu, Z. Zhang, and H. Jin, "A measurement study of a peer-to-peer video-on-demand system," in *IPTPS*, 2007.

[23] D. N. Tran, W. T. Ooi, and Y. C. Tay, "Sax: A tool for studying congestion-induced surfer behavior," in *PAM*, 2006.

[24] E. Altman, F. Boccara, J. Bolot, P. Nain, P. Brown, D. Collange, and C. Fenzy, "Analysis of the TCP/IP flow control in high-speed wide-area networks," in *IEEE CDC*, 1995.

[25] C. Barakat and E. Altman, "Analysis of TCP with Several Bottleneck Nodes," INRIA, Tech. Rep. RR-3620, 1999.

[26] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 8, pp. 133–145, April 2000.

[27] *ns-2*, http://isi.edu/nsnam/ns/.

[28] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," in *ACM SIGCOMM*, 2002.

[29] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *ACM SIGCOMM*, 2011.