Competitive Analysis for Service Migration in VNets*

Marcin Bienkowski
Institute of Computer Science
University of Wrocław, Poland
mbi@
ii.uni.wroc.pl

Anja Feldmann
T-Labs / TU Berlin
Berlin, Germany
anja@
net.t-labs.tu-berlin.de

Dan Jurca
DOCOMO Labs Europe
Munich, Germany
jurca@
docomolab-euro.com

Wolfgang Kellerer
DOCOMO Labs Europe
Munich, Germany
kellerer@
docomolab-euro.com

Gregor Schaffrath
T-Labs / TU Berlin
Berlin, Germany
grsch@
net.t-labs.tu-berlin.de

Stefan Schmid
T-Labs / TU Berlin
Berlin, Germany
stefan@
net.t-labs.tu-berlin.de

Joerg Widmer
DOCOMO Labs Europe
Munich, Germany
widmer@
docomolab-euro.com

ABSTRACT

Network virtualization promises a high flexibility by decoupling services from the underlying substrate network and allowing the virtual network to adapt to the needs of the service, e.g., by migrating servers or/and parts of the network. We study a system (e.g., a gaming application) where network virtualization is used to support thin client applications for mobile devices to improve their QoS. To deal with the dynamics of both the mobile clients as well as the ability to migrate services closer to the client location we advocate, in this paper, the use of competitive analysis. After identifying the parameters that characterize the cost-benefit tradeoff for this kind of application we propose an online migration strategy. The strength of the strategy is that it is robust with regards to any arbitrary request access pattern. In particular, it is close to the optimal offline algorithm that knows the access pattern in advance.

In this paper we present both an optimal offline algorithm based on dynamic programming techniques to find the best migration paths for a given request sequence, and a $O(\mu \log n)$-competitive migration strategy MIG where μ is the ratio between maximal and minimal link capacity in the substrate network for a simplified model. This is almost optimal for small μ, as we also show that there are networks where no online algorithm can achieve a ratio below $\Omega(\log n / \log \log n)$. In contrast, the optimal solution without migration can only achieve a competitive ratio that is linear in the network diameter. Our simulations indicate that the competitive ratio of MIG is robust to the network size, and that the ratio is small if the request dynamics are limited and the requests are correlated.

*Part of this work was performed within the 4WARD project, which is funded by the European Union in the 7th Framework Programme (FP7), the Virtu project, funded by NTT DOCOMO Euro-Labs, and the Collaborative Networking project, funded by Deutsche Telekom AG. We would like to thank our colleagues in these projects for many fruitful discussions. M. Bienkowski is supported by MNiSW grant number N N206 1723 33, 2007–2010.

VISA 2010, September 3, 2010, New Delhi, India.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Algorithms, Design

Keywords

Network Virtualization, Online Algorithms, Competitive Analysis

1. INTRODUCTION

In 2008, the total number of mobile web users outgrew the total number of desktop computer with respect to Internet users [11] for the first time. Providing high quality-of-service (QoS) respective an excellent quality of experience (QoE) to mobile Internet clients is much more challenging, e.g., due to user mobility. However, many applications, including such popular applications as gaming, need a reliable, continuous network service with minimal delay.

Network virtualization [9] is an emerging technology which allows a service specific network to be embedded onto a substrate network in a dynamic fashion. This includes migration of virtual nodes and links as well as virtual servers to meet the applications demands for connectivity and performance. However, to take advantage of this flexibility it is often necessary to know future application demands. Yet, this information is typically not readily available and therefore neither the network resources can be used in an optimal manner nor do the users receive the best possible service.

In this paper, we take a first step towards tackling the general problem by concentrating on a system where thin clients, e.g., on mobile devices, access Internet services, such as a *game server* [14]

via a virtual network. We assume that the distribution of thin clients and therefore the request pattern changes over time. For instance, in the early morning, many requests may originate in Asian countries, then more and more requests come from European users and finally from the US. In this setting it can be beneficial to *migrate* (i.e., *re-embed*) the service closer to the users, e.g., to minimize access delays for the users and to minimize network costs for the providers [13]. Network virtualization allows us to realize such networks. Figure 1 illustrates our setup.

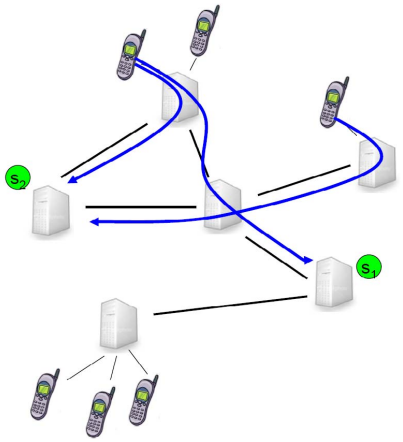

Figure 1: System architecture: Thin mobile clients accessing services s_1 and s_2 along the virtual network.

While moving services close to clients can reduce latency, migration also comes at a cost: the bulk-data transfer imposes load on the network and may cause a service disruption. The cost of migration depends on the available bandwidth in the substrate network [7]. To gain insights into this tradeoff we identify the main costs involved in this system. Intuitively, the benefits from virtualizations are higher the lower the migration cost are relative to the latency penalty. Therefore, a predictable access pattern may ease migration. However, in practice user arrival patterns are hard to predict and thus, we, in this paper, explicitly incorporate uncertainty about future arrivals.

The classic formal tool to study algorithms that deal with inputs (or more specifically: request accesses) that arrive in an online fashion and cannot be predicted is the *competitive analysis framework*. In competitive analysis, the performance of a so-called online algorithm is compared to an optimal offline algorithm that has complete knowledge of the input *in advance*. In effect, the competitive analysis is a *worst case performance analysis* that does not rely on any statistical assumptions. We apply this framework to network virtualization and propose—for a simplified model where the main access cost is delay to the server, and the main migration cost is the available bandwidth between migration source and destination—a competitive migration algorithm whose performance is close to the one of the optimal offline algorithm.

1.1 Related Work

The mobile web today provides browser-based access to the Internet or web applications to millions of users connected to a wireless network via a mobile device. There exists a vast amount of related work on the subject, and we refer the reader to the introductory books, e.g., [24]. In this project, we tackle the question of how *network virtualization* can be used to improve the quality of service for mobile devices.

Network virtualization has gained a lot of attention recently [25] as it enables the co-existence of innovation and reliability [23] and promises to overcome the "ossification" of the Internet [10]. For a more detailed survey on the subject, please refer to [9]. Virtualization allows to support a variety of network architectures and services over a shared substrate, that is, a *Substrate Network Provider (SNP)* provides a common substrate supporting a number of *Diversified Virtual Networks (DVN)*. OpenFlow [18] and VINI [3] are two examples that allow researchers to (simultaneously) evaluate protocols in a controllable and realistic environment. Trellis [4] provides such a software platform for hosting multiple virtual networks on shared commodity hardware and can be used for VINI.

A major challenge in this context is the *embedding* [19] of VNets, that is, the question of how to efficiently and on-demand assign incoming service requests onto the topology. Due to its relevance, the embedding problem has been intensively studied in various settings, e.g., for an offline version of the embedding problem see [16], for an embedding with only bandwidth constraints see [12], for heuristic approaches without admission control see [27], or for a simulated annealing approach see [22]. Since the general embedding problem is computationally hard, Yu et al. [17] advocate to rethink the design of the substrate network to simplify the embedding; for instance, they allow to split a virtual link over multiple paths and perform periodic path migrations. In last year's VISA workshop, Lischka and Karl [15] presented an embedding heuristic that uses backtracking and aims at embedding nodes and links concurrently for improved resource utilization. Such a concurrent mapping approach is also proposed in [8] with the help of a mixed integer program. Finally, several challenges of embeddings in wireless networks have been identified by Park and Kim [20].

In contrast to the approaches discussed above we, in this paper, tackle the question of how to dynamically embed or migrate virtual servers [21] in order to efficiently satisfy connection requests arriving online at any of the network entry points, and thus use virtualization technology to improve the quality of service for mobile nodes. The relevance of this subproblem of the general embedding problem is underlined by the VISA'09 paper by Hao et al. [13] which shows that under certain circumstances, migration of a Samba front-end server closer to the clients can be beneficial even for bulk-data applications.

To the best of our knowledge, this is the first paper to study the embedding of virtual servers in such a system from a competitive analysis perspective. The formal competitive migration problem is related to several classic optimization problems such as facility location, k-server problems, online page migration or metrical task systems [6]. For instance, in the field of *facility location*, researchers aim at computing optimal facility locations that minimize building costs and access costs (see, e.g., [1] for an online algorithm). In the field of *k-server problems* (e.g., [6]), an online algorithm must control the movement of a set of k servers, represented as points in a metric space, and handle requests that are also in the form of points in the space. As each request arrives, the algorithm must determine which server to move to the requested point. The goal of the algorithm is to reduce the total distance that all servers traverse. In contrast, in our model it is possible to access the server remotely, that is, there is no need for the server to move to the request's position. The *page migration problem* (e.g., [2]) occurs in managing a globally addressed shared memory in a multiprocessor system. Each physical page of memory is located at a given processor, and memory references to that page by other processors are

charged a cost equal to the network distance. At times, the page may migrate between processors, at a cost equal to the distance times a page size factor. The problem is to schedule movements on-line so as to minimize the total cost of memory references. In contrast to these page migration models, we differentiate between access costs that are determined by latency and migration costs that are determined by network bandwidth.

Finally, it remains to mention that there is an intriguing relationship between server migration and *online function tracking* [5, 26]. In online function tracking, an entity Alice needs to keep an entity Bob (approximately) informed about a dynamically changing function, without sending too many updates. The online function tracking problem can be transformed into a chain network where the function values are represented by the nodes on the chain, and a sequence of value changes corresponds to a request pattern on the chain. In particular, it follows from [5] that already for some very simple linear substrate networks of size $n = \Theta(\beta)$, where β is the migration cost, no online algorithm can achieve a competitive ratio smaller than $\Omega(\log n / \log\log n)$.

1.2 Our Contributions

This paper makes the following contributions: We study a mobile network virtualization architecture where thin clients on mobile devices access a service that can be migrated closer to the access points to reduce user latency. We identify the main costs tradeoffs in this system (Section 2) and present a competitive analysis framework for its analysis (Section 3). We then describe an optimal offline algorithm for computing the best migration strategy at hindsight. In particular, we derive the following theorem.

THEOREM 1.1. *The optimal offline migration policy* OPT *can be computed in $O(n^3 + n^2 \sum_{t\in\Gamma} |\sigma_t|)$ time, where n is the network size, σ_t is the set of terminal requests at time t, and Γ is the set of rounds in which events occur.*

Moreover, we present the online algorithm MIG that performs close to optimal if the bandwidths in the substrate network do not differ too much between the different links. Concretely, we show the following result.

THEOREM 1.2. MIG *is $O(\mu \cdot \log n)$-competitive, where n is the network size and μ is the maximal ratio between the bandwidths in the substrate network. The competitive ratio of an optimal static strategy (without migration) is linear in the network diameter.*

For small μ, this is asymptotically optimal, as it follows from online function tracking literature [5] that there are networks for which no online algorithm achieves a competitive ratio lower than $\Omega(\log n / \log\log n)$. Our formal insights are complemented by simulation results (Section 4) which indicate that the competitive ratio of MIG is small even in large networks, especially if the request dynamics is low and correlation between requests high.

2. ARCHITECTURE

Our setting is based on the virtualization architecture proposed in [23] for which we are in the process of developing a prototype implementation. The main roles from this architecture related to this work are: The *(Physical) Infrastructure Provider (PIP)*, which owns and manages an underlaying physical infrastructure (called "substrate"); the *Virtual Network Provider (VNP)*, which provides bit-pipes and end-to-end connectivity to end-users; and the *Service Provider (SP)*, which offers application, data and content services to end-users.

We assume that a service provider is offering a service to mobile clients which can benefit from the flexibility of network and service virtualization. The goal of the service provider is to minimize the round-trip-time of its service users to the servers, by triggering migrations depending, e.g., on its (latency) measurements. Concretely, VNP and/or PIP will react on the SP-side changes of the requirements on the paths between server and access points, and re-embed the servers accordingly.

Formally, we consider a substrate network $G = (V, E)$ managed by one or multiple substrate providers (PIP). Each substrate node $v \in V$ has certain properties and features associated with it (e.g., in terms of operating system or CPU power); in particular, we assume that it has a computational capacity $c(v)$. Similarly, each link $e = (u, v) \in E$, with $u, v \in V$, has certain properties, e.g., it is characterized by a bandwidth capacity $\omega(e)$, and it offers the latency $\lambda(e)$. Links between different PIPs are typically less well-provisioned or more expensive than links within a PIP.

In addition, to the substrate network, there is a set T of external machines (the mobile thin clients or simply *terminals*) that access G by issuing requests to virtualized services S hosted on a set of virtual servers by G. Each server $s \in S$ has a certain resource or capacity requirement $r(s)$ that needs to be allocated to s on the substrate node where it is hosted. Henceforth, we will assume that each server in S offers a different service, and each request is targeted to exactly one server. That is, we do not consider server replication.

In order for the machines in T to access the servers S, a fixed subset of nodes $A \subseteq V$ serve as *Access Points* where machines in T can connect to G. Due to the movement of machines in T, the access points can change frequently, which may trigger the migration algorithm. We define R_t to be the multi-set of requests at time t, where $r_t \in R_t$ is a tuple $(a \in A, s \in S)$ specifying the access point and the requested service. Our main objective is to shed light onto the trade-off between the access costs $Cost_{acc}$ of the mobile clients to the current service locations and the server migration cost $Cost_{mig}$: while moving the servers closer to the requester may reduce the access costs and hence improve the quality of service, it also entails the overhead of migration.

We can identify the following main parameters which influence the access and migration costs. A major share of $Cost_{acc}$ is due to the request latency, i.e., the sum of the requests' latencies to the corresponding servers. Observe that the routing of the requests occurs along the shortest paths (w.r.t. latency) on the substrate network. In addition, the access cost depends on the server load, that is, the access cost depends on the capacity $c(v)$ of the hosting node v and the resource demands $c(s)$ of the servers s hosted by v. The correlation between load and delay can be captured by different functions, and is not studied further here. In this paper, we assume that requests are relatively small, and hence, we do not explicitly model bandwidth constraints in $Cost_{acc}$. In conclusion, at time t and for some function f,

$$Cost_{acc}(t) = \sum_{R_t} f\left(\text{delay}(r_t), \text{load}(r_t)\right).$$

In contrast to the requests, which are rather light-weight, the server state is typically large, and hence the traffic volume of migration cannot be neglected. The main cost of migration are service outage periods and the migration itself. The migration cost $Cost_{mig}$ of a virtual server $s \in S$, or the outage period, hence depends to a large extent on the available bandwidth $\omega(p)$ on the migration path $p : src \rightsquigarrow dst$ (along the substrate network) between migration source node src and destination node dst, and the size size(s) of the application s to be migrated. Another major cost factor is the *transit costs*, namely the number k of PIPs on the path.

In summary:

$$Cost_{mig}(t) = \sum_{s \in S} f(\omega(p), k, \text{size}(s))$$

for some function f, where the migration cost is zero if $src = dst$.

Our model so far lacks one additional ingredient: *terminal dynamics* (or mobility). A conservative approach is to assume arbitrary request sets R_t, where R_t is completely independent of R_{t-1}. However, for certain applications it may be more realistic to assume that the mobile nodes move "slowly" between the access points. Note that while users typically travel between different cities or countries at a limited speed, these geographical movements may not translate to the topology of the substrate network. Thus, rather than modeling the users to travel along the links of G, we consider on/off models where a user appears at some node $v_1 \in V$ at time t, remains there for a certain period Δt, before moving to another arbitrary node $v_2 \in V$ at time $t + \Delta t$.

Traditional models may assume that Δt is exponentially distributed. However, in our formal analysis we assume a worst-case perspective and consider arbitrary distributions for Δt. Often, it is reasonable to assume some form of correlation between the individual terminals' movement. For example, in a planetary-scale substrate network, demographic aspects have to be taken account in the sense that during a day, first many requests will originate from Asian countries, followed by an active period in Europe and finally America. However, as it is rather hard to describe and characterize such movement accurately we, in this paper, perform a worst case analysis (w.r.t. latency) that does not use any statistical assumptions.

To what extent the system can benefit from virtual network support and migration depends on several factors, e.g., how frequently the thin clients change the access points. Given rapid changes it may be best to place the server in the middle of the network and leave it there. On the other hand, if the changes are slower or can be predicted, it can be worthwhile to migrate the server to follow the mobility pattern. This constitutes the trade-off studied in this paper.

3. COMPETITIVE ANALYSIS

As already discussed, competitive analysis asks the question: How well does the system perform compared to an optimal offline strategy which has complete knowledge of the entire request sequence in advance? In the following, we present an online migration strategy that is "competitive" to any other online or offline solution for virtual network supported service migration. In order to focus on the main properties and trade-offs involved in the virtualization support of thin clients, we assume a simplified online framework for our formal analysis. We assume a synchronized setting where time proceeds in time slots (or *rounds*).[1] In each round t, a set of σ_t terminal requests arrive in a worst-case and online fashion at an arbitrary set of access nodes A.

Thus the embedding problem is equivalent to the following synchronous game, where an online algorithm ALG has to decide on the migration strategy in each round t, without knowing about the future access requests. Concretely, in each round $t \geq 0$:

1. The requests σ_t arrive at some access nodes A.
2. The online algorithm ALG decides where in G to migrate the servers S. If positions are changed, it pays migration costs $Cost_{mig}(t)$.
3. The online algorithm ALG pays the requests' access costs $Cost_{acc}(t)$ to the corresponding servers.

Note, that we allow ALG to migrate the virtual servers for all the requests of the current time slot t. However, as we assume that a request is much cheaper than a migration, and if there are not too many requests arriving concurrently, our results also apply for a scenario where the last two steps are reordered.

We aim at devising competitive algorithms ALG that minimize the *strict competitive ratio* ρ: Let $\text{ALG}_t(\sigma)$ be the migration and access costs incurred by ALG in round t under a request arrival sequence σ (a sequence of access points), that is,

$$\text{ALG}_t(\sigma) = Cost_{acc}(t) + Cost_{mig}(t).$$

Let $\text{OPT}(\sigma)$ be the optimal cost of an offline algorithm OPT for the given σ, that is, OPT has a complete knowledge of σ and can hence optimize the sever locations "offline". ρ is the ratio of the costs of ALG and OPT. Thus, our objective is to minimize:

$$\rho = \max_{\sigma} \frac{\sum_t \text{ALG}_t(\sigma)}{\text{OPT}(\sigma)}$$

For an online algorithm that uses randomization, we consider the expected costs against an oblivious adversary without access to the outcome of the random coin flips of the algorithm.

In the remainder of this paper, let σ_t denote the set of request issued in round t. We make the following simplifications for our analysis: (1) We focus on a single substrate provider (i.e., we do not consider multiple PIPs). (2) There is only one server. (3) We neglect the load on the infrastructure nodes. (4) The access costs $Cost_{acc}(t)$ are given by the total number of hops between mobile client and server, that is, we consider the shortest path inside G and add one hop for the access link which may be wireless. Henceforth, for any two nodes $u, v \in V$ on the topology, let $Cost_{acc}(u, v)$ denote the shortest hop distance between u and v. (5) The migration cost $Cost_{mig}(t)$ is given by the bandwidth constraint of the smallest edge capacity on the migration path. Let $Cost_{mig}(u, v)$ denote the migration cost on a path from u to v. (The path will be clear from the context.) Thus, $Cost_{mig}(u, v) = \max_e size(s)/\omega(e)$ where $size(s)$ is the size of the migrated server s and e is a link on the migration path from u to v.

Note, our migration cost model corresponds to a *tree metric*, that is, for any given migration strategy, there is an algorithm achieving the same performance by migrating only along the minimal *Steiner tree* (w.r.t. link latency due to bandwidth constraints). To see this, assume that the migration graph contains a cycle. Consider two arbitrary nodes u and v on that cycle, and let the two paths from u to v be denoted by p_1 and p_2. Then, the migration cost on a path from u to v is given by the weakest link on the corresponding path. Thus, if the link with lowest bandwidth on p_1 has lower or equal capacity than p_2, using migration on p_2 does not come at an extra cost. (Note that for load-balancing reasons in scenarios with multiple parallel migrations, this may change.)

3.1 Optimal Offline Algorithm

The competitive analysis compares the performance of a given online algorithm to an optimal offline strategy. In the following, we present an optimal migration strategy OPT for the case that the request sequence σ is given in advance. As stated above, this serves as a reference for evaluating our online protocol. OPT is based on dynamic programming techniques. It exploits the fact that migration exhibits an optimal substructure property: Given that at time t, the server is located at a given node u, then the most cost-efficient migration path that leads to this configuration consists solely of optimal sub-paths. That is, if a cost minimizing path π to node u at

[1]Note that while this assumption simplifies the analysis, it is not critical for our results.

time t leads over a node v at time $t' < t$, then there cannot be a cheaper migration sub-path that leads to v at time t' than the corresponding sub-path in π.

OPT essentially fills out a matrix opt[time][node] where $opt[t][v]$ contains the cost of the minimal migration path that leads to a configuration where the server satisfies the requests of time t from node v. Assume that initially, the service is located at node v_0. Thus, initially, $opt[0][u] = Cost_{mig}(v_0, u) + \left[\sum_{v \in \sigma_0} Cost_{acc}(v, u)\right]$ as the migration origin is v_0, and as a request needs to travel on the access link from the terminal to v and from there to u (w.l.o.g., we assume that the cost $Cost_{acc}$ contains the first wireless hop from terminal to substrate network).

For $t > 0$, we find the optimal values $opt[t][u]$ by considering the optimal migration paths to any node v at time $t-1$, and adding the migration cost from v to u. That is, in order to find the optimal cost to arrive at a configuration with server at node u at time t:

$$\min_{v \in V} \left[opt[t-1][v] + Cost_{mig}(v, u) + \sum_{w \in \sigma_t} Cost_{acc}(w, u) \right]$$

where we assume that $Cost_{acc}$ includes the first (wireless) hop of the request from the terminal to the substrate network, and where $Cost_{mig}(v, v) = 0 \ \forall v$.

We have the following runtime result.

THEOREM 3.1. *The optimal offline migration policy* OPT *can be computed in* $O(n^3 + n^2 \sum_{t \in \Gamma} |\sigma_t|)$ *time, where* Γ *is the set of rounds in which events occur.*

PROOF. Note that we can constrain ourselves to optimal offline algorithms where migration will only take place in "active" rounds Γ with requests. This is useful in case of sparse sequences with few requests. The $opt[\cdot][\cdot]$-matrix contains $|\Gamma| \cdot n$ entries. In order to compute a matrix entry, we need to consider each node $v \in A$ from which a migration can originate; for each such node, the access cost from all the requests in σ_t need to be computed. Both the shortest access paths and the migration costs can be looked up in a pre-computed table (pre-computation in time at most $O(n^3)$, e.g., by *Floyd-Warshall's algorithm*) and require a constant number of operations only, which implies the claim. □

Finally, we remark that OPT can be generalized for arbitrary cost functions and multiple PIPs.

3.2 QoS without Migration

In order to compare the benefits of migration to a static scenario, we derive the competitive ratio of fixed strategies.

LEMMA 3.2. *A system without migration yields a competitive ratio of*

$$\rho \in \Theta(Diam(G)),$$

where $Diam(G)$ *is the network diameter of substrate network* G.

PROOF. Clearly, $\rho \in O(\text{Diam}(G))$ because for each request, the optimal offline algorithm pays at least 1, and an online algorithm pays at most $\text{Diam}(G) + 1$. On the other hand, we can fix any online algorithm with server at position v. Then there exists a location u such that $Cost_{acc}(v, u) \geq \text{Diam}(G)/2$. If the request sequence σ consists of requests originating from u only, the online algorithm pays at least $(\text{Diam}(G)/2 + 1) \cdot |\sigma|$, while the offline algorithm may place the server at u and pay $|\sigma|$. □

In a fixed scenario, the best location for hosting s is in the network center, i.e., the location which minimizes the distance traveled by the requests, namely at node u for which

$$u := \arg\min_{v \in V} \max_{w \in V} Cost_{acc}(v, w).$$

Note, since there is no migration in the fixed scenario, the competitive ratio does not depend on any bandwidth constraints (i.e., on link weights). This indicates that in networks with highly heterogeneous links or with links whose capacity changes quickly over time, a fixed solution without migration may be appealing.

3.3 Online Migration

In this section we describe our online algorithm MIG for competitive migration. The basic idea of MIG is to strike a balance between the request latency cost $Cost_{acc}^{\text{MIG}}$ and the migration cost $Cost_{mig}^{\text{MIG}}$ it incurs, and to continuously move closer to a possible optimal position. The intuition is that after a small number of migrations only, either MIG is at the optimal position, or an optimal offline algorithm OPT must have migrated as well during this time period. Either way, OPT cannot incur much smaller costs than MIG. In other words, by using MIG for moving to good locations in the network, a possible offline algorithm that migrates less frequently cannot have much lower access costs than MIG; on the other hand, an offline strategy with frequent migrations will have a cost similar to $Cost_{mig}^{\text{MIG}}$.

Let us first consider a scenario with constant bandwidth capacities, i.e., $\omega(e) = \omega \ \ \forall e \in E$ and identify its migration cost $\beta = 1/\omega$.

The algorithm MIG divides time into *epochs*. In each epoch MIG monitors, for each node v, the cost of serving all requests from this epoch by a server kept at v. We denote this counter by L_v. MIG keeps the server at a single node w till L_w reaches β. In this case, MIG migrates the server to a node u chosen uniformly at random among nodes with the property $L_u < \beta$. If there is no such node, MIG does not migrate the server, and the epoch ends in that round; the next epoch starts in the next round and the counters L_v are reset to zero.

LEMMA 3.3. MIG *is* $O(\log n)$*-competitive in networks with constant bandwidth.*

PROOF. Fix any epoch $\mathcal{E}$ and let β denote the migration cost. If OPT migrates the server within $\mathcal{E}$, it pays β. Otherwise it keeps it at a single node paying the value of the corresponding counter at the end of $\mathcal{E}$. By the construction of MIG, this value is at least β, and thus in either case $\text{OPT}(\mathcal{E}) \geq \beta$.

The migrations performed by MIG partition $\mathcal{E}$ into several *phases*. According to our migration strategy, the access cost of MIG in each phase is at most β. Below, we show that the expected number of migrations within one epoch is at most H_n, where H_n is the n-th harmonic number. The number of phases is then $H_n + 1$, and hence $\text{MIG}(\mathcal{E}) \leq \beta \cdot H_n + \beta \cdot (H_n + 1) = \beta \cdot O(\log n)$. This yields the competitiveness of MIG.

Let $\{v_i\}_{i=1}^n$ be the sequence of nodes in order their counters reach the value β (with ties broken arbitrarily). Assume that in a phase MIG keeps the server at node v_i and let T_i be the expected number of server migrations till the end of $\mathcal{E}$. If $i = n$, then the current phase is the last one in $\mathcal{E}$, and thus $T_n = 0$. Otherwise, $i < n$, and at the end of this phase, MIG chooses a next place for the server uniformly at random from the set $\{v_j : i < j \leq n\}$.

Hence, we obtain the recursive formula

$$T_i = 1 + \sum_{j=i+1}^{n} \frac{1}{n-i} \cdot T_j.$$

By a simple induction, one can show that $T_i = H_{n-i}$. Thus, if MIG starts $\mathcal{E}$ with the server at node v_k, the expected number of migrations in $\mathcal{E}$ is $H_{n-k} \leq H_n$. $\square$

For networks with general bandwidths, MIG can be adopted in such a way that it migrates when the counter of the current location v reaches $\min_e \omega(e)$, that is, when $L_v \geq \min_e \omega(e)$. Thus, the cost of the optimal algorithm in each epoch is at least $\min_e \omega(e)$, while the cost of MIG is at most $\max_e \omega(e)$. Thus, by the same arguments as in the proof of Lemma 3.3, we immediately obtain the following result.

THEOREM 3.4. MIG *is* $O(\mu \cdot \log n)$*-competitive in general networks, where* $\mu = \max_{e,e' \in E} \omega(e)/\omega(e')$.

4. SIMULATION

Although the main focus of this paper is on the formal and worst-case competitive analysis, we briefly report on our preliminary simulation results on "average-case", randomized access distributions. We ran experiments on a simple linear substrate network consisting of n substrate nodes are arranged in a chain. We assumed that the duration for which a terminal remains at a specific access point is exponentially distributed (with parameter λ), after which a new access point is chosen uniformly at random from the set of all access points.

The simulations show that the higher the correlation among the requests is, the more a system will benefit from our migration strategy. Moreover, even for relatively high degrees of mobility, MIG outperforms the optimal static solution (without migration), especially in large networks where the available bandwidth does not differ too much between links. Interestingly, the observed competitive ratio is small even for relatively large networks, see Figure 2.

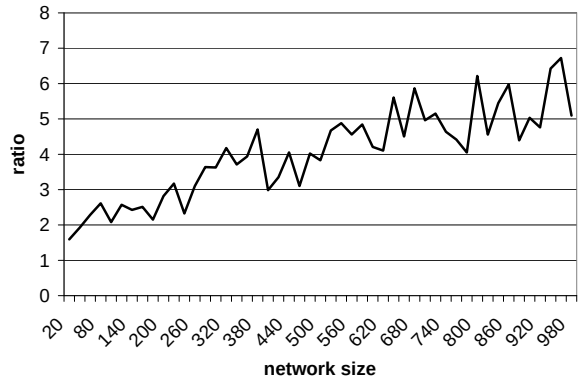

Figure 2: Competitive ratio of MIG for different network sizes. In this simulation, we used $\omega(e) = 1/20\ \forall e$ and considered one request with exponentially distributed sojourn time (with $\lambda = 1/50$). The total runtime is 1000 rounds.

Figure 3 shows three traces of the competitive ratio over time for a network with $n = 100$. The figure illustrates that the larger the correlation of the requests is, i.e., the more requests travel together, the higher the migration benefits.

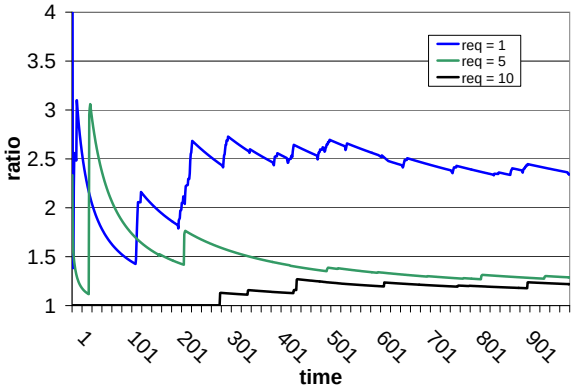

Figure 3: Competitive ratio of MIG over time. In this simulation, we used $\omega(e) = 1/20\ \forall e$ and considered one request with exponentially distributed sojourn time (with $\lambda = 1/50$). The network size was $n = 100$. In the first experiment, a single client moves along the topology in a uniform and random fashion. In the second and third experiment, correlation is increased by moving five and ten clients simultaneously (and perfectly correlated) along the network.

5. DISCUSSION AND EXTENSIONS

5.1 Generalization and Optimization

For the sake of our analysis, we have considered a simplified model for the access and migration costs. While we express the first one as a function of number of hops, the latter is mainly based on the transfer cost (computed as the ration between the virtual server size and the migration path bottleneck bandwidth). In a general setting, both metrics are composed of two components: the transmission cost and service cost for the required information. The service cost depends on the total machine load, where the virtual server is hosted (application queuing delay), and the transmission cost depends on the length of the considered network path (be it for access or migration), the machine load, and the number of traversed infrastructure providers PIPs. The last dependency introduces also a business dimension to the cost computation, expressed via SLAs between PIPs, which provision interfaces only for certain types and sizes of resources. At the same time, in our competitive analysis framework, we can instantiate different optimization problems, in which we incorporate different possible objective functions and constraints. Examples are: minimize the overall access cost under migration cost constraints, or minimize the migration cost under delay constraints, or resource constraints with respect to infrastructure capabilities. Each instance of such an optimization may lead to specialized algorithms which can be derived for the particular problem being addressed.

5.2 Multiple Servers and Replication

So far, we have constrained our design space to systems in which a single virtual server is migrated on an underlying network infrastructure graph. However, in the long run, we envision a system in which multiple virtual servers can be accessed in parallel by the active clients. Moreover, for large network systems, we want to allow virtual server replication as an additional mechanism to server migration. Within this context, various questions must be addressed. An on/off substrate node model, in which each infrastructure node may host a virtual server is no longer sufficient. Node

storage capacity has to be considered and influences the cost of migration/access of/to a given virtual server. Given certain storage resources in the system and migration/access cost as a function of the node load, the overall system must carefully balance the number of replicas of each virtual server that exists at any given time, and decide whether a new replica should be instantiated or an existing replica should be migrated. Also, the competition for resources among different virtual servers must be addressed. To this end, service utility functions which encompass general QoS/QoE and popularity metrics must be incorporated in the overall optimization framework.

5.3 Mobility and Prediction

Our competitive analysis method and OPT and MIG algorithms optimize the placement of the virtual server based on rounds, after observing the active user requests in that round. Our analysis highlights the merits of the algorithm both on a per round basis and over longer periods of time, when requests are completely random. However, when request locations are correlated in time, our approach can be further optimized by the use of additional estimation and prediction mechanisms. Thus, a careful study of user mobility patterns and request location updates may enhance our virtual server location decisions. Time window based mechanisms which incorporate the location of a given request in the recent past and extrapolate possible future requests can be incorporated in our system, either on individual or aggregate basis. Furthermore, user mobility data and activation patterns can be studied and incorporated as request mobility estimation models in our framework. While we perceive such methods as independent of our migration analysis methodology, they are a desirable addition to our framework for the case of large, global systems with stringent resource and QoS constraints.

6. CONCLUSION

At the heart of network virtualization lies the ability to react to changing environments in a flexible fashion. Competitive analysis is the classic mathematical framework to design and study protocols for such systems. This paper proposed an online algorithm that dynamically migrates a server to the locations of highest demand.

When is migration efficient? Our formal analysis and our simulations suggest that our migration strategy is particularly efficient in scenarios where requests are correlated, as MIG is able to identify good migration strategies *online*. Our algorithm achieves a good competitive ratio in systems with low request dynamics but remains efficient also for higher dynamics. In addition, while our formal analysis suggests a logarithmically increasing competitive ratio in the network size, our simulation experiments indicate that MIG scales well in case of randomized access patterns. Compared to a static solution without dynamic migration, MIG is naturally better for larger networks, especially if edge capacities do not differ too much and as long as the request dynamics is limited and still exhibits an exploitable structure.

7. REFERENCES

[1] H.-K. Ahn, S.-W. Cheng, O. Cheong, M. Golin, and R. van Oostrum. Competitive facility location: the voronoi game. *Theor. Comput. Sci.*, 310(1-3):457–467, 2004.

[2] Y. Bartal. Distributed paging. In *Dagstul Workshop on On-line Algorithms*, pages 97–117, 1996.

[3] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In vini veritas: Realistic and controlled network experimentation. In *ACM SIGCOMM*, 2006.

[4] S. Bhatia, M. Motiwala, W. Muhlbauer, V. Valancius, A. Bavier, N. Feamster, L. Peterson, and J. Rexford. Hosting virtual networks on commodity hardware. Technical Report GT-CS-07-10, Georgia Tech, 2008.

[5] M. Bienkowski and S. Schmid. Online function tracking with generalized penalties. In *Proc. 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, 2010.

[6] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

[7] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg. Live wide-area migration of virtual machines including local persistent state. In *Proc. VEE*, 2007.

[8] K. Chowdhury, M. R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. In *Proc. INFOCOM*, 2009.

[9] M. K. Chowdhury and R. Boutaba. A survey of network virtualization. *Elsevier Computer Networks*, 54(5), 2010.

[10] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in cyberspace: Defining tomorrow's Internet. In *Proc. SIGCOMM*, 2002.

[11] Data from 2008. International Telecommunications Union, 2009.

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

[13] F. Hao, T. V. Lakshman, S. Mukherjee, and H. Song. Enhancing dynamic cloud-based services using network virtualization. *SIGCOMM Comput. Commun. Rev.*, 40(1):67–74, 2010.

[14] U. C. Kozat, Y. Gwon, and R. Jain. An overlay server system (oss) platform for multiplayer online games over mobile networks. In *Proc. Global Telecommunications Conference, 2006 (GLOBECOM)*, 2006.

[15] J. Lischka and H. Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proc. VISA*, pages 81–88, 2009.

[16] J. Lu and J. Turner. Efficient mapping of virtual networks onto a shared substrate. In *Technical Report, WUCSE-2006-35, Washington University*, 2006.

[17] J. R. M. Yu, Y. Yi and M. Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review*, 38(2):17–29, Apr 2008.

[18] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, 2008.

[19] B. Monien and H. Sudborough. Embedding one interconnection network in another. In *Computational Graph Theory*, 1990.

[20] K.-m. Park and C.-k. Kim. A framework for virtual network embedding in wireless networks. In *Proc. 4th International Conference on Future Internet Technologies (CFI)*, pages 5–7, 2009.

[21] R. Potter and A. Nakao. Mobitopolo: A portable infrastructure to facilitate flexible deployment and migration of distributed applications with virtual topologies. In *Proc. 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures (VISA)*, pages 19–28, 2009.

[22] R. Ricci, C. Alfeld, and J. Lepreau. A solver for the network

testbed mapping problem. *SIGCOMM Comput. Commun. Rev.*, 33(2):65–81, 2003.

[23] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy. Network virtualization architecture: Proposal and initial prototype. In *Proc. VISA*, 2009.

[24] A. Talukder and R. Yavagal. *Mobile Computing: Technology, Applications, and Service Creation*. McGraw-Hill, 2006.

[25] US National Science Foundation. Global environment for network innovations (geni). *http://www.geni.net/*, 2006.

[26] K. Yi and Q. Zhang. Multi-dimensional online tracking. In *Proc. of the 19th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1098–1107, 2009.

[27] Y. Zhu and M. H. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *Proc. INFOCOM*, 2006.