Optimal Migration Contracts in Virtual Networks: Pay-as-You-Come vs Pay-as-You-Go Pricing

Xinhui Hu[1], Stefan Schmid[2], Andrea Richa[1], and Anja Feldmann[2]

[1] SCIDSE, Arizona State University, Tempe, AZ 85287, USA
{xinhui.hu,aricha}@asu.edu
[2] TU Berlin & T-Labs, Germany
{stefan,anja}@net.t-labs.tu-berlin.de

Abstract. Network virtualization realizes the vision of an Internet where resources offered by different stakeholders are used and shared by multiple coexisting virtual networks. The abstraction introduced by network virtualization opens new business opportunities. We expect that in the near future, infrastructure providers (or resource brokers and resellers) will offer flexibly specifiable and on-demand virtual networks over the Internet, similarly to the elastic resources in today's clouds.

This paper initiates the discussion on the optimal resource allocations in such an economic environment. We attend to a scenario where a flexible service (such as a web service or an SAP database) is implemented over a virtual network. This service can be seamlessly migrated closer to the current locations of the (mobile) users. We assume that a virtual network provider offers different contracts to the service provider, and we distinguish between two fundamentally different pricing models: (1) a *Pay-as-You-Come* model where the service provider needs to decide in advance which time-based contracts to buy in order to implement the service, and a (2) *Pay-as-You-Go-model* where the service provider is charged only when the service terminates and only for the amount of resources actually used. In both cases, the virtual network provider may offer a *discount* if more resources are bought, e.g., buying a resource contract of double duration or of twice as much bandwidth only costs fifty percent more than a simple contract. We describe two optimal migration algorithms PAYC (for the Pay-as-You-Come model) and PAYG (for the Pay-as-You-Go model), provide a quantitative comparison of the two pricing models, and discuss their implications. Finally, extensions to online algorithms are discussed.

1 Introduction

The Internet becomes more and more virtualized and programmable (or "software-defined"), and we witness a trend towards extending the cloud paradigm to the *network*. Researchers in the field of *network virtualization* develop prototype architectures that herald flexibly specifiable, fully *virtual networks* (VNets) (also known as *CloudNets*): virtual networks that can be requested at short notice (and even be migrated arbitrarily within the specification constraints), while providing isolation guarantees (e.g., in terms of QoS or security). This paradigm has the potential to open a network infrastructure

D. Frey et al. (Eds.): ICDCN 2013, LNCS 7730, pp. 285–299, 2013.

for a wide range of new and innovative services, and it is believed that new economical entities will emerge that lease (or re-lease) infrastructure parts to service providers.

We expect that in the near future, such virtual networks connecting arbitrary locations (and spanning multiple autonomous systems and providers) in the Internet can be leased similarly to the resource leasing models of today's clouds. This paper attends to a use case for such dynamic VNets where a service provider offers a flexible and latency-critical service (for instance a web service, an SAP server or a game server) to its mobile customers whose demand and locations changes over time (e.g., due to time-zone effects or commuting). We assume that the service provider itself uses the resource services of a substrate infrastructure provider (e.g., a physical infrastructure provider or a virtual network provider) in order to offer a low-latency access to a server which can be migrated seamlessly in the VNet (i.e., without reconfiguration or changes of routable network addresses). The service provider is faced with the challenge that while moving the server closer to the customers improves QoS (and/or reduces roaming costs), frequent migrations come at service interruption and bulk data transfer costs. We initiate the study of optimal offline and online migration strategies for the service provider under two different pricing models.

Our Contribution. This paper initiates the study of the virtual server migration problem from an economical perspective. We compare the two most basic pricing policies *Pay-as-You-Come* and *Pay-as-You-Go* (see, e.g., [14]), in which a service provider has to pay in advance for time-based contracts respectively in retrospect for the resources actually used. The service provider receives a discount when buying larger contracts, e.g., a contract of twice the resource volume only costs 50% more. As a first step, we, in this paper, design offline migration algorithms for different settings and discount functions. We find that optimal offline solutions can indeed by computed in polynomial time by using non-trivial dynamic programs. We theb use these algorithms to quantify and compare the two pricing models by simulation. We discuss the implications of these models and find that, as expected, Pay-as-You-Come pricing yields higher costs on the service provider side than Pay-as-You-Go pricing, especially for moderate discounts. Interestingly, the distribution and structure of the costs and the used contracts differ significantly for the two pricing schemes, and it turns out that the QoS guarantees under the Pay-as-You-Go model are much better due to the efficient resource investments.

Note that offline algorithms are particularly interesting if demand patterns can be predicted well (e.g., if it depends on time-zone effects or commuter behavior). However, offline algorithms can also serve as a yardstick to evaluate the performance (i.e., the so-called *competitive ratio*) of online algorithms in simulations. This paper also initiates exploring online migration strategies.

2 Economical Service Migration

A virtual network topology can be modeled as a graph $G = (V, E)$ where $V(G)$ denotes the set of nodes and $E(G)$ the set of links. We assume that a service provider can place its service (i.e., the server) on any location in the virtual network. Requests can originate from different access points in $V(G)$, and the access cost is given by the shortest path (depending on some given metric D) to the server location in $V(G)$. In order to reduce

the access cost, the virtual server can be migrated along the links in $E(G)$. To do so, the service provider needs to purchase bandwidth along the migration path.

We attend to a scenario where a virtual network provider offers the service provider a choice of contracts of different durations in which dedicated resources can be leased in the virtual network (e.g., for migration), i.e., $\mathfrak{D} = \{d_1, d_2, \ldots, d_k\}$ (we assume $d_1 < d_2 < \ldots < d_k$). In addition to the contract durations, the service provider can choose between different bandwidths along the links, i.e., it can choose among the following set of bandwidths for each link: $\mathfrak{B} = \{b_1, b_2, \ldots, b_q\}$ (we also assume that $b_1 < b_2 < \ldots < b_q$).

We consider two different pricing models. Under *Pay-as-You-Go* pricing, a customer only needs to pay for the used resources after the actual consumption (or at regular time intervals T), and the best contract is determined according to the usage pattern *a posteriori*. Pay-as-You-Go pricing is often used in the context of cloud resource leasing. In contrast, in the *Pay-as-You-Come* model, a customer needs to decide *in advance* which kind of *time-based* contracts she is interested in, and needs to buy them before the actual resource usage. Examples for this model can be found, e.g., in the context of private Internet access where users often pay in advance and independently of the actual usage pattern.

In this paper, in order to focus on the main tradeoffs, we initiate the study of these pricing models in a simplified scenario where the virtual network consists of two locations only (e.g., one in the U.S. and one in Asia); we will refer to these locations by L (left) and R (right) respectively and normalize their distance to one unit. The server can be migrated arbitrarily between the two locations if a corresponding resource contract is present for the bulk-data transfers. Concretely, a contract in the *Pay-as-You-Come* model consists of a duration d_i and the bandwidth b_j to lease the virtual link between the two sites for d_i units (e.g., days) and at a bandwidth of b_j (e.g., Mbit/s). The price of the contract is given by a function $f(d_i, b_j)$, where $f(\cdot, \cdot)$ describes a monotonic increasing discount over the contract duration and over the amount of reserved resources. For example, a twice as long contract may cost only 50% more, and doubling the reserved bandwidth may cost only 30% more. In the *Pay-as-You-Go* model, the customer only needs to pay when the service is finished or after a given duration, i.e., every T time units (e.g., a month), and only for the resources (and bulk data transfers) that are actually used. Concretely, if μ_i migrations are performed during the time period T at a bandwidth of $b_j \in \mathfrak{B}$, the overall costs amount to $f(\mu_i, b_j)$.

The main objective is to minimize the migration and contracting costs (denoted by MigCost and ConCost) while providing good Quality-of-Service (QoS) guarantees (minimize access cost AccCost). Hence, we seek to minimize the following cost function:

$$\text{Cost} = \text{AccCost} + \text{MigCost} + \text{ConCost}$$

We assume there are n requests total, denoted by a set $< r_1, r_2, \cdots, r_n >$ at respective times $< t_1, t_2, \cdots, t_n >$. The access cost is given by the latency of the requests $r_i \in V(G)$ to the location of the server $s_i \in V(G)$, i.e., AccCost $= \sum_i D[r_i, s_i]$ where r_i and s_i denote the ith request node and the server location at time t_i. The migration cost MigCost is given by the service interruption time (see also [5]), i.e., the time to transfer

the server which is determined by the bandwidth of the weakest link along the migration path. (In a system supporting live migration, this cost can be negligible and set to zero.) Concretely, the migration cost is computed as MigCost $= \sum_i S \cdot D[s_{i-1}, s_i]/b_i$, where S is the server size, $D[s_{i-1}, s_i]$ denotes whether the locations $s_{i-1} \in \{L, R\}$ and $s_i \in \{L, R\}$ differ (recall that $D[s_{i-1}, s_i]$ is 1 if $s_{i-1} \neq s_i$, and 0 otherwise), and $b_i \in \mathfrak{B}$ is the (minimal) bandwidth along the migration path. Finally, the contract cost is computed as described above, i.e., ConCost $= \sum_i f(d_i, b_i)$ for the Pay-as-You-Come model and as ConCost $= f(\mu, b_i)$ for Pay-as-You-Go model, where $d_i \in \mathfrak{D}$, $b_i \in \mathfrak{B}$ and μ is the total number of migrations.

3 Migration Strategies

This section presents optimal algorithms to compute the best set of contracts and optimal migration strategies for the two presented pricing models. We will first present an algorithm PAYC for the Pay-as-You-Come model and prove its optimality, and then extend this algorithm to a PAYG algorithm which solves the Pay-as-You-Go model. Both our algorithms PAYC and PAYG are based on dynamic programming, and fill out matrices such that optimal substructures are reused.

3.1 Pay-as-You-Come

Let us now turn our attention to the first, time-based pricing model. Our PAYC algorithm stores intermediate *minimum total cost* results (access, migration and contract costs) in a 3-dimensional matrix $C_{n \times n \times 4}$ where n is the total number of requests. $C[i, j, k]$ denotes an entry of the matrix, where $i, j \in [1, n]$ and $k \in \{(s, s') | s, s' \in \{L, R\}\}$. $C[i, j, (s, s')]$ denotes the minimum total cost for satisfying all requests from r_i to r_j for a scenario where at the beginning of the ith request the server is at node s and at the end of request j the server is at node s'. We also need a matrix $(AM_m)_{n \times n \times 4}$ for each bandwidth $b_m \in \mathfrak{B}$. For a fixed bandwidth b_m during the entire interval $[t_i, t_j]$, entry $AM_m[i, j(s, s')]$ stores the combined access and migration costs for the best migration strategy that satisfies the sequence of requests from r_i to r_j, assuming that the server is located at node s at the start of request r_i and at node s' at the end of request r_j. The contract costs, given by the function f, are not included in the entries of AM_m.

Given these data structures, we can describe algorithm PAYC (Algorithm 1) for the Pay-as-You-Come model. PAYC exploits that the optimal contract from request time t_i to request time t_j can either be decomposed into two consecutive subperiods with no overlapping contracts, or be obtained by buying a contract of long duration d_v and bandwidth b_m if $d_{v-1} < t_j - t_i + 1 \leq d_v$, where $d_v, d_{v-1} \in \mathfrak{D}$.

PAYC starts by initializing the optimal costs if we were to serve only one request r_i, for all possible combinations of starting server location s and ending server location s' at time t_i. According to our model, the access cost is equal to the distance between the current requesting node r_i and the server location s' at the end of time t_i, denoted by $D[r_i, s']$. If the request at time t_i comes from the server location s', then no access cost is needed since $D[r_i, s']$ is 0; otherwise the access cost is positive. Recall that the migration cost for request r_i is computed as $S \cdot D[s, s']/b_m$, where $b_m \in \mathfrak{B}$ is the selected bandwidth and S is the migrated server size. We store the respective optimal cost

Algorithm 1. Algorithm PAYC

Input: Requests $<r_1, r_2, ..., r_n>$ at respective times $<t_1, t_2, ..., t_n>$.
Output: Minimum cost.

1: **for** $i = 1$ **to** n **do**
2: **for** all pairs $(s, s') \in \{L, R\}^2$ **do**
3: **for** $m = 1$ **to** q **do**
4: $AM_m[i, i, (s, s')] \leftarrow D[s', r_i] + S \cdot D[s, s']/b_m$
5: $C[i, i, (s, s')] \leftarrow \min_{1 \le m \le q}\{AM_m[i, i, (s, s')] + f(d_1 * D[s, s'], b_m)\}$
6: **for** $l = 2$ **to** n **do**
7: **for** $i = 1$ **to** $n - l + 1$ and **pairs** $(s, s') \in \{L, R\}^2$ **do**
8: $j \leftarrow i + l - 1$
9: $C[i, j, (s, s')] \leftarrow \min_{i \le u < j; s'' \in \{L,R\}}\{C[i, u, (s, s'')] + C[u + 1, j, (s'', s')]\}$
10: **if** $d_{v-1} < t_j - t_i + 1 \le d_v$, for some $v = \{1, \cdots, k\}$ **then**
11: **for** $m = 1$ **to** q **do**
12: $AM_m[i, j, (s, s')] \leftarrow \min_{s'' \in \{L,R\}}\{AM_m[i, i, (s, s'')] + AM_m[i + 1, j, (s'', s')]\}$
13: **if** $C[i, j, (s, s')] > \min_{1 \le m \le q}\{AM_m[i, j, (s, s')] + f(d_v, b_m)\}$ **then**
14: $C[i, j, (s, s')] \leftarrow \min_{1 \le m \le q}\{AM_m[i, j, (s, s')] + f(d_v, b_m)\}$
15: **return** $\min_{s_{\text{final}} \in \{L,R\}} C[1, n, (s_{\text{init}}, s_{\text{final}})]$

of satisfying request r_i (which may or may not incur a non-zero access cost $D[r_i, s']$, depending on whether $r_i \neq s'$ or not) using bandwidth b_m, with starting and ending positions of the server s and s' respectively, in $AM_m[i, i, (s, s')]$. We choose a bandwidth $b_m \in \mathfrak{B}$ such that the total cost, including the contract cost $f(d_1, b_m)$ if a migration accurs, is minimized, and store the optimal total cost in $C[i, i, (s, s')]$.

Next, we consider the total costs for sequences of more than one request. Note that there are l requests occurring between time t_i and t_j, where $i < j$ are defined in Lines 7 and 8 of the algorithm and $l\ (= j - i + 1) > 1$. We have two alternative options: (i) we can split the interval $[t_i, t_j]$ at the time t_u of request r_u, where $i \le u < j$, and buy contracts for the intervals $[t_i, t_u]$ and $[t_{u+1}, t_j]$ independently for the two possible locations s'' of the server at time t_u; or (ii) we can buy a long contract of duration $d_v \in \mathfrak{D}$ and some bandwidth $b_m \in \mathfrak{B}$ to cover all the l requests if the period $t_j - t_i + 1$ is between d_{v-1} and d_v. The smaller cost of these two cases gives the optimal cost for the interval $[t_i, t_j]$.

We also update $AM_m[i, j, (s, s')]$, for all possible bandwidths b_m. Basically we extend the intervals already considered by one request (r_i), and we store in $AM_m[i, j, (s, s')]$ the migration strategy that minimizes the total access and migration costs for satisfying requests r_i through r_j using bandwidth b_m for starting and ending positions of the server s and s' respectively. Note that by taking into account all possible positions of the server at the end of request r_i, we consider all the possibilities of adding r_i to all the best possible strategies already computed for the subsequence $r_{i+1}, \ldots, r_j$ (ending at node s').

We process the previous steps in increasing order of l until l spans all the requests. Thus, the optimal cost is given by $\min_{s_{\text{final}} \in \{L,R\}} C[1, n, (s_{\text{init}}, s_{\text{final}})]$, where s_{init} is the initial server location.

Theorem 1. PAYC *(see Algorithm 1) computes the optimal contracts for Pay-as-You-Come model. The time complexity of* PAYC *is* $O(n^2(n+kq))$*, where* n *is the number of requests,* k *is the number of contract durations and* q *is the number of different bandwidth contracts.*

Proof. The correctness follows by induction over the number of request l and by the optimal substructure property. Due to space constraints, we only sketch the proof. The claim is trivially true for sequences of one request (Lines 1–5). Consider the time interval from t_i to t_j with l requests, where $1 \leq i \leq j \leq n$ and $2 \leq l(= j-i+1) \leq n$. This interval is split into two subintervals (Case I), or a long contract is bought that covers the entire interval (Case II). In *Case I*, we split the cost at time t_u with the server located at s'' such that the total cost $C[i,u,(s,s'')] + C[u+1,j,(s'',s')]$ is minimized, where $i \leq u \leq j$ and $s'' \in \{L,R\}$. Since the number of requests in the two subintervals, $u-i+1$ and $j-u$, are shorter than l, by the induction hypothesis, $C[i,u,(s,s'')]$ and $C[u+1,j,(s'',s')]$ already store the optimal costs for these two intervals respectively. In *Case II*, we buy a long contract to cover the whole interval. Given a certain server location s'' at the start of the time t_{i+1}, $AM_m[i+1,j,(s'',s')]$ already stores the optimal access and migration strategy cost for bandwidth b_m for interval $[t_{i+1},t_j]$. Therefore, an optimal migration strategy for interval $[t_i,t_j]$ using bandwidth b_m can be obtained by adding r_i to the optimal strategies selected for the interval $[t_i+1,t_j]$ and optimizing over the choice on whether to migrate the server to serve r_i or not (resulting in the two possible choices for s'', the position of the server right after satisfying request r_i).

Now we consider the time complexity of the PAYC algorithm. Clearly, the first phase of the algorithm requires time $O(nq)$. The second phase consists of three nested loop and has a complexity of $O(n^2 \cdot (n+kq))$. □

3.2 Pay-as-You-Go

Optimal solutions can also be computed for the Pay-as-You-Go model, and the algorithm PAYG is similar to the algorithm PAYC. As discussed above, in the Pay-as-You-Come model we need to decide when to migrate, which contracts to buy, and how much bandwidth to use. In the Pay-as-You-Go model, we still need to make a decision on when to migrate and how much bandwidth should be reserved, but we do not have to explicitly decide on a time contract. However, unlike the Pay-as-you-Come model, in the Pay-as-you-Go model, a bandwith b_m has to be chosen and fixed for satisfying the entire sequence of requests $r_i,\ldots,r_j$. Also, the contract cost in this model is directly dependent on the number of migrations of the server, and hence we explicitly have to keep track of this number.

Algorithm PAYG is listed in Algorithm 2. PAYG uses a new matrix $(A_m)_{n\times n\times 4}$ to store the access cost under a certain bandwidth b_m, $1 \leq m \leq q$, and another matrix $(N_m)_{n\times n\times 4}$ is used to store the migration number for bandwidth b_m. A matrix $(C_m)_{n\times n\times 4}$ stores the total cost for bandwith b_m. In the entries of the new matrices, the elements $A_m[i,j,(s,s')]$ and $N_m[i,j,(s,s')]$ store the access cost and the number of migrations, respectively, for the optimal solution between time t_i and t_j with an initial server location s and a final server location s', where $s,\ s' \in \{L,R\}$. The entry $C_m[i,j,(s,s')]$ stores the total optimal cost within this time period for bandwith b_m.

Algorithm 2. Algorithm PAYG

Input: Requests $<r_1, r_2, ..., r_n>$ at respective times $<t_1, t_2, ..., t_n>$.
Output: Minimum Cost.

1: **for** $i = 1$ **to** n **do**
2: **for** all pairs $(s, s') \in \{L, R\}^2$ and $1 \leq m \leq q$ **do**
3: $A_m[i, i, (s, s')] \leftarrow D[s', r_i]$
4: $N_m[i, i, (s, s')] \leftarrow D[s, s']$
5: $C_m[i, i, (s, s')] \leftarrow A_m[i, i, (s, s')] + S \cdot N_m[i, i, (s, s')]/b_m + f(D[s, s'], b_m)$
6: **for** $l = 2$ **to** n **do**
7: **for** $i = 1$ **to** $n - l + 1$ **do**
8: $j \leftarrow i + l - 1$
9: **for** all pairs$(s, s') \in \{L, R\}^2$ and $1 \leq m \leq q$ **do**
10: $C_m[i, j, (s, s')] \leftarrow \min_{i \leq u < j; s'' \in \{L,R\}} \{A_m[i, u, (s, s'')] + A_m[u + 1, j, (s'', s')] + S \cdot (N_m[i, u, (s, s'')] + N_m[u + 1, j, (s'', s')])/b_m + f((N_m[i, u, (s, s'')] + N_m[u + 1, j, (s'', s')]), b_m)\}$
11: Let (u, s'') be the parameter and location of request r_u at t_u that minimized Line 10.
12: $A_m[i, j, (s, s')] \leftarrow A_m[i, u, (s, s'')] + A_m[u + 1, j, (s'', s')]$
13: $N_m[i, j, (s, s')] \leftarrow N_m[i, u, (s, s'')] + N_m[u + 1, j, (s'', s')]$
14: **return** $\min_{s_{\text{final}} \in \{L,R\},\ 1 \leq m \leq q} C_m[1, n, (s_{\text{init}}, s_{\text{final}})]$

The basic idea behind PAYG is to compute the optimal solution for a scenario where all the requests require the same bandwidth, and then choose the smallest cost among all the bandwidth options. PAYG starts off by computing the optimal costs for satisfying one request (Lines 1-5). Given the request r_i and the starting and ending server locations s, s', the access cost is given by $D[s', r_i]$ which is 0 if the final server location s' and the request location r_i coincide, and 1 otherwise. Meanwhile $D[s, s']$ will indicate that the server migrates to the other location to serve the current request if $D[s, s']$ is 1. Otherwise, there is no migration, and the starting and ending server locations s, s' describe the same node. We store the optimal solution in $C_m[i, i, (s, s')]$ for each bandwidth b_m, where $C_m[i, i, (s, s')] = D[s', r_i] + S \cdot D[s, s']/b_m + f(D[s, s'], b_m)$.

Now PAYG iterates over the number of requests l (Line 6). For each value of l, we compute all the possible cases, as in Lines 7-13. First, we select from $[1, n - l - 1]$ the value i denoting the index of the first of these l requests. Obviously, the index of the last of the l requests (denoted by j) would be $i + l - 1$, as in Line 8. Assume that the server is located at node s at the time when the ith request occurs, and located at node s' at the end of the jth request, where $s, s' \in \{L, R\}$. We look for a way to split the duration such that the total cost $C_m[i, j, (s, s')]$ is minimized, as shown in Line 10. We use u, m, and s'' to denote the index of the request occurring at the chosen split point, the chosen bandwidth, and the location of the server (Line 11). Therefore, the total cost consists of the summation of the access costs of two subintervals, the summation of the migration costs of two subintervals, and a long contract cost covering the whole period. Here, the access cost is computed as $A_m[i, u, (s, s'')] + A_m[u + 1, j, (s'', s')]$, the migration cost is computed as $(N_m[i, u, (s, s'')] + N_m[u + 1, j, (s'', s')])/b_m$ and the contract cost

is computed as $f(N_m[i,u,(s,s'')]+N_m[u+1,j,(s'',s')],b_m)$, for a certain bandwidth b_m. We store the access cost in $A_m[i,j,(s,s')]$ (Line 12) and the number of migrations in $N_m[i,j,(s,s')]$ (Line 13) for the current duration.

For each bandwidth b_m, we store the optimal solution to serve all the requests in C_m matrix. Thus the optimal cost is hence obtained by computing $\min_{s_{\text{final}}\in(L,R),1\leq m\leq q} C_m[1,n,(s_{\text{init}},s_{\text{final}})]$ (Line 14).

The following claim follows by simple induction over the number of requests.

Theorem 2. PAYG *(see Algorithm 2) computes the optimal contracts for the Pay-as-You-Go model. The time complexity is $O(qn^3)$, where n is the number of requests and q is the number of different contract bandwidths.*

Proof. We argue by induction on the number of the requests l considered. In base case, when there is just one request ($l = 1$), lines 1-5 will give the optimal solutions under each bandwidth. As for the inductive step, we follow a similar strategy as for PAYC. We split at time t_u with the server located at s'' such that the access cost and the migration cost of two sub intervals will minimize the total amount for the current duration. Since we consider all possible splits at all times within the whole interval as well as all the server migrations (Line 10), we choose the best option for the longer interval.

Regarding the time complexity, Lines 3, 4, and 5 each take $O(1)$ time, respectively. Since Lines 3-5 are executed $4nq$ times, the total running time of Lines 1-5 is $O(nq)$. Considering that Lines 10-13 are executed $O(n^2q)$ times and $l \leq n$, we know that the running time of Lines 6-13 is $O(qn^3)$. Therefore, the time complexity of Algorithm 2 is $O(qn^3)$. □

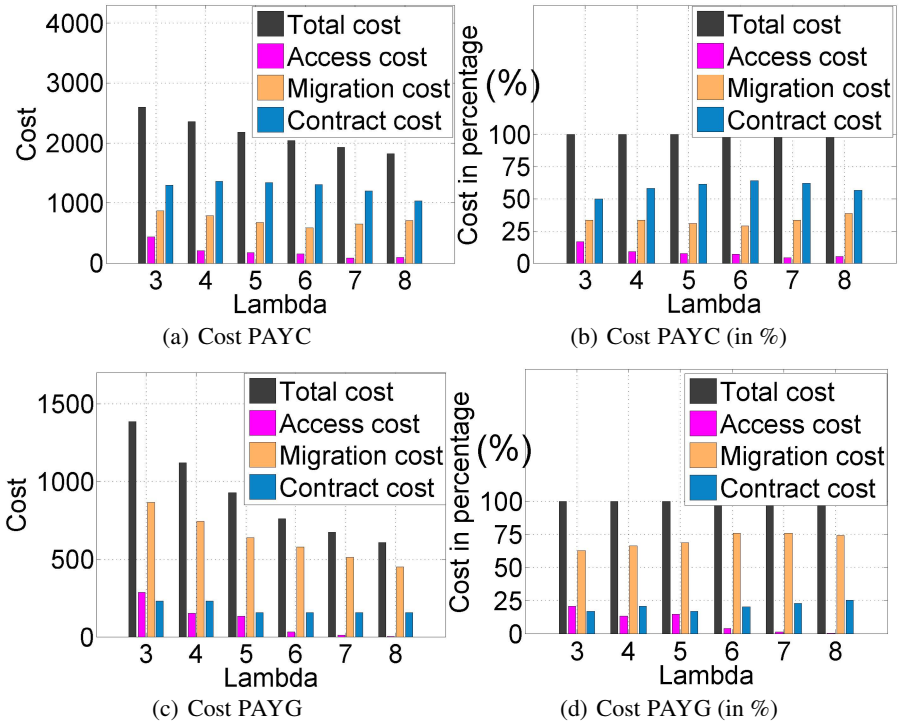
4 Quantitative Comparison

The presented economical migration algorithms allow us to shed light on the properties of the two pricing models. We study three different discount functions $f_{\text{lin}}, f_{\text{sqrt}}, f_{\log}$ which offer cheaper contracts if longer (in terms of days) or larger (in terms of leased bandwidth) contracts are bought: f_{lin} is linear ("get twice as much for a 50% higher price"), f_{sqrt} grows according to a square root function and hence describes a steeper discount, and $f_{\log}$ even gives an even steeper logarithmic discount. For all three discount functions, the cost of a one-day contract with 50 Mbit/s bandwidth is the same, namely $f_i(1,50) = 6$ for $i \in \{\text{lin, sqrt, log}\}$. Concretely, we use $f_{\text{lin}}(d_i,b_j) = 1.5 \cdot f_{\text{lin}}(d_i/2,b_j) = 1.5 \cdot f_{\text{lin}}(d_i,b_j/2)$=$1.5^{(\lfloor \log d_i \rfloor + b_j/50-1)} \cdot f_{\text{lin}}(1,50)$, $f_{\text{sqrt}}(d_i,b_j) = \sqrt{d_i b_j/50} \cdot f_{\text{sqrt}}(1,50)$, and $f_{\log}(d_i,b_j) = \log(d_i b_j/50) \cdot f_{\log}(1,50)$. We assume a server of size $S = 250$ MB, and we assume that the access cost for one remote request is five units (a request originating at the node where the service is located is free). We study a scenario where the provider offers two different bandwidth capacities, namely 50 Mbit/s and 100 Mbit/s, and four types of contract durations, namely 1, 30, 60 and 100 days (i.e., $\mathfrak{B} = \{50,100\}$ and $\mathfrak{D} = \{1,30,60,100\}$).

We study a simple request pattern where requests originate from L and R in turn, e.g., requests originating in Asia alternate with requests originating in the U.S..

Simplified Demand Scenario: We assume that requests alternate infinitely between the two sites L and R in the following manner: requests originate from one site (one per round) for a time interval duration which is chosen according to an exponential distribution with parameter λ, before requests originate from the opposite side again (according to the same distribution).

We simulate $n = 1500$ requests, and present the average over five runs for each experiment.



(a) Cost PAYC (b) Cost PAYC (in %)

(c) Cost PAYG (d) Cost PAYG (in %)

Fig. 1. Cost distribution for PAYC and PAYG

We discuss the following simulations in more detail.

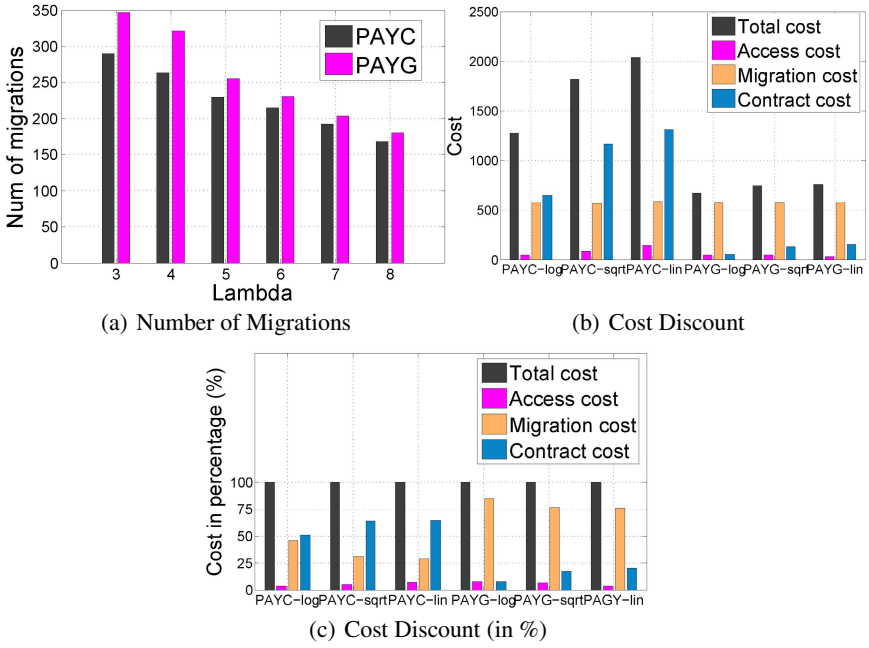
Cost Distribution and Number of Migrations. We analyze how the cost distributes among the access cost, the migration cost, and the contract cost for the two algorithms PAYC and PAYG. All experiments discussed here are conducted under the natural f_{lin} discount function. Figure 1(a) shows the absolute costs of PAYC as a function of λ. We observe that the total cost and the access cost decrease for larger λ while the migration and contract stay much more stable. This is clear as requests originating from one site for longer time periods render it worthwhile to migrate and buy longer contracts.

The contract increases firstly and then decrease after some point, since the total migration numbers decrease and hence the contract cost is reduced. As the number of migrations decrease, the average number of migrations within a contract is also decreased. Therefore, PAYC will buy smaller bandwidth for such contract, which will result in larger migration costs(also shown in Table 2). Figure 1(b) presents the relative shares of the three costs. While the access costs approach zero for larger λ since the server is often at the right location, the contract costs and the migration costs stay stable since PAYC migrates a lot even for larger λ. The same results for PAYG are shown in Figures 1(c) and 1(d), respectively. As a first takeaway, we see that the cost distribution of Figure 1(c) defers from Figure 1(a) in that the total costs are lower, i.e., Pay-as-You-Go is always the cheaper option than Pay-as-You-Come pricing for the customer. Also note that in contrast to the Pay-as-You-Come model, the migrations constitute a larger share of the overall costs, since the contract cost is given by the number of migrations and the amount of leased bandwidth under the discount function; hence the contract cost is lower than the one of PAYC for the same number of migrations. Moreover, there are relatively more frequent migrations under the PAYG model, see Figure 2(a), which also explains the lower access costs (i.e., this improves QoS experienced by the users). Regarding the relative cost shares (Figure 1(d)), we can see that the percentage for the access cost is decreasing while the percentages for the migration cost and the contract cost are increasing slowly. Again, when λ is large enough and the requests become more local, since migrations only occur at the beginning of each interval, the number of migrations (as well as all three cost components) eventually decreases.

Contract Distribution. Different pricing models and scenarios result in different types and combinations of contracts, and it is interesting to study the frequency (or popularity) distribution of the contracts. Table 1 reports on the average number of the contracts as a function of λ, for different contract durations and bandwidths, under the PAYC algorithm and for f_{lin}. We see that when λ is small and migrations are dense, longer duration contracts occur frequently since the server migrates often. However, as λ increases, all lengths of contracts decrease. As λ increases, the average number of migrations in a contract decreases and hence the smaller bandwidth will benefit more than the larger one. Therefore, it turns out to buy more contracts with smaller bandwidth. This can also be seen in Table 2 which records the average number of migrations in different contracts accordingly (average over five runs).

Table 1. Distribution of purchased contracts (discount function f_{lin})

Dur-Bw \ λ	3	4	5	6	7	8
1-50	11.2	8	15.4	13.8	18.4	39.2
60-50	0	0	0	0	2.4	0.8
60-100	1.4	2	1.4	2.8	1	0.4
100-50	0	0	0	0.6	2	5.4
100-100	11	11	11.2	10	7.6	3.4



(a) Number of Migrations (b) Cost Discount

(c) Cost Discount (in %)

Fig. 2. Number of migrations and effect of discount function

Table 2. Number of migrations for each contract (discount function f_{lin})

Dur-Bw \ λ	3	4	5	6	7	8
1-50	1	1	1	1	1	1
60-50	0	0	0	0	8,5	0
60-100	17.67	14	13.5	11.5	0	0
100-50	0	0	0	13	13	12.57
100-100	27.33	23.58	19.45	17.33	15	14.5

Impact of Discount Function. Finally, let us compare the different discount functions in more detail. Figure 2(b) and Figure 2(c) explore the absolute and relative (in %) cost distributions for PAYC and PAYG under different discount functions. Clearly, the higher the discount, the smaller the total cost. Moreover, not surprisingly the performance of PAYG is always better than that of PAYC since the total cost is less for PAYG compared to that for PAYC. However, the difference of the costs for the two models is smaller for higher discounts, i.e., the difference for the logarithmic discount function is smaller than for a discount function which follows a square root.

5 A First Look at Online Migration

Although the main focus of this paper is on predictable demand scenarios and offline algorithms, in this section, we want to initiate the discussion of online algorithms. The online discussion builds upon our offline results in two respects: First, some algorithmic techniques from the offline variant may be used also for the online variants. For example, an online algorithm may try to predict the future from the past, and apply an optimal offline algorithm on a sequence of recent past requests in order to make decisions on how to deal with upcoming requests. Second, offline algorithms are often needed to evaluate the performance of an online algorithm. The ratio of the cost of an online algorithm divided by the cost of an optimal offline algorithm is also known as the *competitive ratio* [3].

Both online algorithms presented in the following are inspired by the (optimal) offline variants and seek to amortize costs over time. To simplify the presentation, we assume a constant bandwidth scenario.

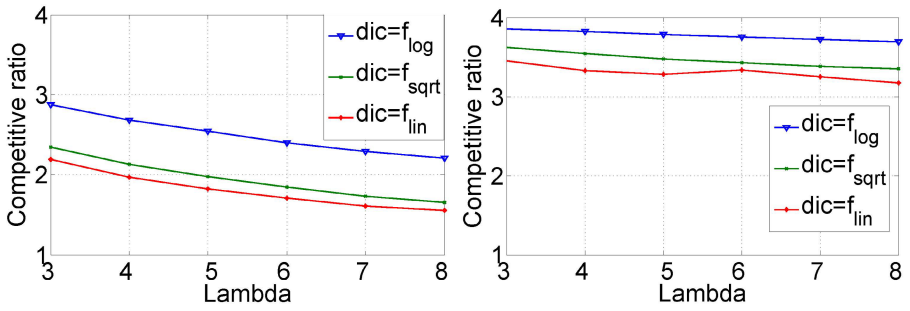
ONC: The online Pay-as-You-Come algorithm ONC tracks the access costs it incurs at the current location using a counter C. Once the counter exceeds the migration cost (given by the server size divided by the bandwidth), ONC migrates the server and resets C. If there is currently no contract available for migration, ONC checks whether a contract longer than the most recently used contract would have been better *for the past requests*. Concretely, ONC checks longer contracts one by one (in increasing order of length) and compares their costs in the corresponding intervals (starting from the last migration) to the cost ONC incurred during that time period. As soon as a better contract is found, it is chosen. Otherwise, ONC checks whether a contract shorter than the most recent contract should be chosen. The following heuristic is applied: ONC checks whether during the last contract, the number of migrations was larger in the first half or the second half of the contract time interval. In case of the first half, ONC will buy the shorter contract; otherwise, ONC chooses the same contract as last time.

Now let us discuss a simple online algorithm ONG for the Pay-as-You-Go model. Since the customers only need to pay for the resources actually consumed, ONG just needs to decide when to migrate.

ONG: Let the counter C_1 record the number of the migrations performed so far and let the counter C_2 denote the total access costs. If the access cost C_2 reaches the migration cost plus marginal migration contract costs (i.e., $f(C_1+1,b)-f(C_1,b)$, for bandwidth b), ONG migrates the server, increments counter C_1, and resets counter C_2.

Given our optimal offline algorithms, it is interesting to study the *competitive ratio* of ONC and ONG. We conduct simulations with the same three discount functions $f_{\log}, f_{\text{sqrt}}$ and f_{lin}, the same contract set and the same access cost as in Section 4. The bandwidth used in our experiments is 50 Mbit/s.

The competitive ratios for ONC and ONG are presented in Figure 3. We observe that the ratios for both algorithms are relatively small (between 1.5 and 4) and decrease for larger λ (lower dynamics). This can be explained by the fact that with higher λ, requests



(a) Competitive ratio for ONC. (b) Competitive ratio for ONG.

Fig. 3. Effect of discount function on competitive ratio. We simulate 1500 requests and present the average over five runs.

remain more local and migration patterns more obvious. A second takeaway is that the competitive ratio for the lowest discount function f_{lin} is best, while higher discounts like f_{log} are handled worse by our online algorithms. Especially in the Pay-as-You-Come model, our online algorithm has more difficulties to deal with high discounts, as it tends to buy too many short contracts (ONC migrates more often than the offline algorithm). Also under Pay-as-You-Go pricing, the offline algorithm can exploit discounts relatively better, although to a lesser extent. (The offline algorithm migrates relatively more frequently for higher discounts.)

6 Related Work

Our work is motivated by the advent of first network virtualization prototype architectures such as GENI. For a good overview of the network virtualization field, see [7]. Theoretical research on network virtualization often focuses on the problem of how to *embed* VNets, e.g., [6,19,15] (and especially the survey [4]), while benefitting from specification flexibilities [13]. Naturally, there are also many papers and results on migration (e.g., [1,3,11,18]): the possibility to migrate is one of the key advantages of the virtualization abstraction; it is due to the decoupling of services from the physical infrastructure. Indeed, it has been shown that it can make sense to migration a Samba front-end server closer to the clients even for bulk-data applications [12]. Our work builds upon the formal migration model studied in [3] and ports it to an economical setting.

Economical aspects of network virtualization are much less well-understood, but there exist strong ties with related problems in, e.g., cloud computing. For example, Armbrust et al. [2] made an effort to understand cloud computing economical models for long-term hosting a service in the cloud. Dash et al. [9] proposed an economic model for self-tuned cloud caching targeting the service of scientific data. Recently, Pal and Hui [14] devised and analyzed three inter-organizational economic models relevant to cloud networks, and formulated non-cooperative price and QoS games between

multiple cloud providers existing in a cloud market. In the context of network virtualization, Schaffrath et al. [16] identified stakeholders and economical roles in a network virtualization environment. The authors distinguish between a physical infrastructure provider, a virtual network provider (i.e., resource reseller), a virtual network operator and a service provider. In terms of pricing, Even et al. [10] presented an online algorithm which decides which VNets to accept and embed such that the overall provider *benefit* is maximized. The benefit threshold of when to accept a VNet can be seen as a simple form of pricing. Migration is not considered in [10].

Finally, a description of our own network virtualization prototype (currently using VLANs) which is developed at Telekom Innovation Laboratories and NTT DoCoMo Eurolabs and which motivates our work can be found in [16]. Currently, migration is seamless (i.e., without the need for reconfigurations) but not live. See [8] for a migration demo.

7 Conclusion

There is a large body of literature on economical aspects of cloud computing, but much less is known about efficient (virtual) network pricing. Interestingly, while cloud (or node) resources are often priced according to a flexible *per-use* or pay-as-you-go policy, networking services such as MPLS connectivity are often charged according to usage-independent, time-based policies. [17] This is particularly surprising as network demand is likely to exhibit a higher variance over time than, e.g., storage resources. For instance, distributed SAP systems may be fully synchronized only sporadically (but then lead to high network loads), whereas the resource requirements of, e.g., a mail service normally grows monotonically over time.

We understand this paper as a first step to study the effect of virtual network pricing policies on *service migration*. We focused on the offline setting where demand patterns are given (e.g., describe regular time-of-day or commuter effects). Such online algorithms can also be useful to evaluate the competitive ratio of online algorithms in simulations. We presented two optimal algorithms for efficient service migration in different economic settings. We believe that the used algorithmic techniques are relatively general and can be extended to more complex scenarios, e.g., to networks supporting live migration or more complex virtual network topologies.

Acknowledgments. The authors would like to thank the anonymous reviewers for their valuable comments.

References

1. Agarwal, S., Dunagan, J., Jain, N., Saroiu, S., Wolman, A., Bhogan, H.: Volley: automated data placement for geo-distributed cloud services. In: Pro. 7th USENIX NSDI (2010)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM 53(4), 50–58 (2010)

3. Arora, D., Bienkowski, M., Feldmann, A., Schaffrath, G., Schmid, S.: Online strategies for intra and inter provider service migration in virtual networks. In: Proc. IPTComm (2011)
4. Belbekkouche, A., Hasan, M., Karmouch, A.: Resource discovery and allocation in network virtualization. IEEE Communications Surveys Tutorials (99), 1–15 (2012)
5. Bienkowski, M., Feldmann, A., Jurca, D., Kellerer, W., Schaffrath, G., Schmid, S., Widmer, J.: Competitive analysis for service migration in VNets. In: Proc. ACM VISA, pp. 17–24 (2010)
6. Chowdhury, K., Rahman, M.R., Boutaba, R.: Virtual network embedding with coordinated node and link mapping. In: Proc. IEEE INFOCOM (2009)
7. Chowdhury, N.M.K., Boutaba, R.: A survey of network virtualization. Computer Networks 54, 862–876 (2010)
8. CloudNet. Migration Demo (2012), `http://www.youtube.com/watch?v=llJce0F1zHQ`
9. Dash, D., Kantere, V., Ailamaki, A.: An economic model for self-tuned cloud caching. In: Proc. IEEE International Conference on Data Engineering, pp. 1687–1693 (2009)
10. Even, G., Medina, M., Schaffrath, G., Schmid, S.: Competitive and Deterministic Embeddings of Virtual Networks. In: Bononi, L., Datta, A.K., Devismes, S., Misra, A. (eds.) ICDCN 2012. LNCS, vol. 7129, pp. 106–121. Springer, Heidelberg (2012)
11. Hajjat, M., Sun, X., Sung, Y.-W.E., Maltz, D., Sripanidkulchai, S.R.K., Tawarmalani, M.: Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud. In: Proc. ACM SIGCOMM (2011)
12. Hao, F., Lakshman, T.V., Mukherjee, S., Song, H.: Enhancing dynamic cloud-based services using network virtualization. SIGCOMM Comput. Commun. Rev. 40(1), 67–74 (2010)
13. Ludwig, A., Schmid, S., Feldmann, A.: The price of specificity in the age of network virtualization (short paper). In: Proc. 5th IEEE/ACM UCC (2012)
14. Pal, R., Hui, P.: Economic Models for Cloud Service Markets. In: Bononi, L., Datta, A.K., Devismes, S., Misra, A. (eds.) ICDCN 2012. LNCS, vol. 7129, pp. 382–396. Springer, Heidelberg (2012)
15. Schaffrath, G., Schmid, S., Feldmann, A.: Optimizing long-lived cloudnets with migrations. In: Proc. 5th IEEE/ACM UCC (2012)
16. Schaffrath, G., Werle, C., Papadimitriou, P., Feldmann, A., Bless, R., Greenhalgh, A., Wundsam, A., Kind, M., Maennel, O., Mathy, L.: Network virtualization architecture: proposal and initial prototype. In: Proc. ACM VISA (2009)
17. Schönherr, M.: T-labs berlin. Personal Communication (2012)
18. Wood, T., Shenoy, P., Ramakrishnan, K., der Merwe, J.V.: Cloudnet: Dynamic pooling of cloud resources by live wan migration of virtual machines. In: Proc. ACM VEE (2011)
19. Zhang, S., Qian, Z., Wu, J., Lu, S.: An opportunistic resource sharing and topology-aware mapping framework for virtual networks. In: Proc. IEEE INFOCOM (2012)