Dynamic Session Management for Static and Mobile Users: A Competitive On-Line Algorithmic Approach

Yigal Bejerano* Israel Cidon
Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa 32000, Israel
bej@tx.technion.ac.il,
cidon@ee.technion.ac.il

Joseph (Seffi) Naor†
Bell Laboratories, Lucent Technologies
600 Mountain Ave.
Murray Hill, NJ 07974.
naor@research.bell-labs.com

ABSTRACT

Most modern communication systems that support real-time communication (ATM, ISDN, Frame-Relay, etc.) are based on connection oriented technologies. This is also true for cellular and Personal Communication Systems, where the users are mobile. Consequently, selecting good routes for Virtual Channels (VCs) is considered as one of the most important problems in modern networks. Usually, the VC route is determined during the session initialization along the shortest path (with respect to some metric) between the users and remains the same during the entire session. However, the user mobility and the fact that the cost of a session may also depend on its duration, motivate the development of *dynamic session management* algorithms. Such algorithms may reroute the VCs dynamically during the sessions for improving session cost and for supporting handoff operations of mobile users.

This work deals with the general *Session Management* problem and proposes several on-line algorithms for managing session between static or mobile users. In the case of static users, we present a 4-competitive algorithm for the situation where the path cost functions are concave. In the presence of mobile users we prove that for a general graph, mobile users and arbitrary link cost functions with positive setup cost, the competitive ratio of the best on-line algorithm is at least $\Omega(\log n)$, where n is the number of nodes in the network. We also present constant competitive on-line algorithms for several practical topologies and movement assumptions.

*Research supported by the Eshkol Fellowship from the Israeli Ministry of Science.

†On leave from the Computer Science Department, Technion - Israel Institute of Technology, Haifa 32000, Israel.

1. INTRODUCTION

Most modern communication systems that support real-time communication are based on connection oriented technologies. These include the standard telephone network and its ISDN extensions [1], as well as the emerging Frame-Relay [2], ATM [2] and QoS support of Internet such as RSVP [3, 4]. Connections are also used in cellular and Personal Communication Systems [5, 6, 7], where the users are mobile and may change their point of attachment to the system's infrastructure during a session. Such changes are termed *handoff* operations. Connection-oriented networks require the establishment of a virtual channel (VC) between the session participants before sending information, and maintaining it during the session. As a result, selecting good routes for VCs is considered as one of the most important problems in modern communication networks, and there are numerous papers on this subject, e.g., [8, 9].There are several optimization issues related to VC routing, such as maximizing the overall network revenue, or minimizing the call blocking probability. In many cases, the selected route may be required to optimize the session cost according to a known metric (delay, throughput, or a nominal cost).

Usually, the VC route is determined during the session initialization along the shortest path connecting the users and does not change during the entire session. Even in cases where mobility is supported, most cellular systems do not compute an entirely new route, but make incremental changes to the route's links [5, 10]. It is however suggested that rerouting may achieve better system performance in such cases [11, 12]. The mobility of the users and the fact that the cost of a session also depends on its duration, motivate the development of *dynamic session management* algorithms. Such algorithms may change the VC routes dynamically during the sessions for both improving the network utilization [12, 13, 14, 15, 16], and supporting handoff operations of mobile users [17, 18, 19]. However, these works cannot guarantee upper bounds on the cost of a specific session.

Our work considers the general *session management* problem for both static and mobile users. There are multiple alternative paths between session users. Each path is characterized by a cost function for using this path for a given duration of time. This function represents both the setup cost and the hold cost of the given path. The *setup cost* represents the cost associated with the establishment operation, in particular signaling cost and setup latency. The *hold cost* determines the expense related to the use of network resources held by the VC. Our goal is to manage a session such that its total cost will be as low as possible. We allow our algorithms to change the VC route dynamically during the session in order to reduce its cost. To the best of our knowledge, this is the first work that takes into account both session setup cost and hold cost and guarantees analytically upper bounds on the session overall cost.

Consider the following example where two users can be connected by one of two feasible paths. Suppose that the hold cost of the first path is one unit per minute, while the hold cost of the second path is ten units for the entire session with no time limit. In this situation there is no single optimal path between the two users. VCs of short sessions should be routed over the first path while VCs of long ones should be routed over the second path. However, if the session duration is not known in advance, there does not exist a way to choose an optimal path at the session initialization. Thus, we are in an on-line dynamic decision setting, where decisions need to be made without prior knowledge of future events.

A common way for measuring the performance of an on-line algorithm is *competitive analysis*. The performance of an on-line algorithm is compared with the performance of an optimal off-line algorithm that knows the sequence of events in advance. The maximum ratio between their respective performances, taken over all sequences, is called the *competitive ratio*. Extensive work has been done in recent years for finding the competitive ratio for different problems such as paging, call admission and circuit routing, scheduling and load balancing, and finance. The reader is referred to the recent book of Borodin and El-Yaniv [20] for an extensive survey of this area.

For the above example, the best deterministic on-line strategy for this problem is routing the session VC over the first path for a duration of ten minutes and then rerouting it over to the second path. This strategy guarantees that in the worst case the session cost will be at most twice the cost of an optimal off-line strategy. This example is known in the on-line literature as the *ski rental* problem [20, 21]. We note that the general session management problem is related to on-line replacement problems [20, 22, 23]. Our work is the first to address the general session management problem using a competitive analysis approach. In [24], handoff rerouting algorithms that are not allowed to hold unused resources are considered and analyzed using competitive analysis.

1.1 Our Results

We first consider the case of static users. We derive conditions under which competitive on-line algorithms exist. Intuitively, we require that the cost functions of all possible paths be concave with respect to the duration. For this case we are able to provide a 4-competitive algorithm. This is achieved by first considering the case of linear cost functions, and then generalizing the algorithm to any concave cost functions. Our on-line algorithm uses the idea that a rerouting decision can only be made if an optimal off-line algorithm has already incurred sufficient cost with respect to the current session duration. We note that the results of [22] imply an algorithm with a competitive ratio of $4 + 2\sqrt{2} \approx 6.83$ for the case of a linear cost function.

Next, we consider the case of mobile users. Finding competitive algorithms for managing a session between mobile users is a much harder task. Here, the unknown information is not only the session duration, but also future movements of the mobile participants. We first present a lower bound of $\Omega(\log n)$ on the competitive ratio for general graphs, mobile users, and arbitrary link cost functions (including linear functions), where n is the number of nodes in the network. We note that the mobile case is related to the on-line Steiner tree problem which has similar lower bounds [25, 26].

Under the assumption of linear cost functions, we consider several important cases where competitive on-line algorithms exist. First, we present a 2-competitive on-line algorithm for managing a session in a tree network. The difficulty here is how to avoid repeated allocations of the same session resources which may increase the session cost with respect to the cost achieved by an optimal off-line algorithm. We also provide a matching lower bound on the competitive ratio. We next consider a ring and present a 4-competitive algorithm. In this case the algorithm is required to determine both the VC path at each time step during the session, and when to release unused link resources. Finally, we consider general graphs with the following restrictions: (a) There is a correlation between the setup cost and the hold cost. For every graph links, the ratio between its hold cost and its setup cost is bounded by two constants c_1 and c_2, where $c_1 \leq c_2$. (b) The user movement rate is limited. With these restrictions we present a $(\frac{c_2}{c_1} + 2)$-competitive algorithm. When $c_1 = c_2$, the algorithm is 3-competitive. This case is of practical importance since it reflects environments of current mobile networks.

2. THE NETWORK MODEL

We assume a connection-oriented network modeled by an undirected graph $G(V, E)$, where the nodes and links represent communication switches and full duplex links, respectively. Users are attached to the network switches, and can either be static or mobile. A mobile user may move from node to node and change its attachment point. A session between two users requires the establishment of a virtual channel (VC) between the corresponding nodes and maintaining it during the session. This means allocating session resources at each edge over the VC path and holding them during the session. The cost of a session depends on both the edges used for the VC and on the duration they are used for. Each edge $e \in E$ is associated with a cost function $f_e(\tau)$ that defines the cost of using edge e for a duration of τ time units, where τ is measured from the time the VC resources are allocated until they are released. Therefore, the entire cost of a session, whose VC is routed over path p and its duration is τ, is given by $f_p(\tau) = \sum_{e \in p} f_e(\tau)$. Note that the function $f_p(\tau)$ represents both setup cost and hold (maintenance) cost of the VC for the given duration. We assume that the time of establishing a VC is negligible with respect to the session duration.

Now, consider a session σ between two static users. We use the term *session duration* for denoting the time period that elapsed from the session initialization at time zero till the termination time τ. The session σ is represented by the triplet $\sigma = (u, v, \tau)$, where u and v are the session end nodes, and τ is the session duration. Let $W_{u,v}$ denote the set of all possible paths between u and v, and let $f_p(\tau)$ be the cost of establishing and holding a VC across path p for a period of τ time units. We use the term *lower envelope function* between u and v for $L_{u,v}(\tau) = \min_{p \in W_{u,v}}\{f_p(\tau)\}$. This function defines the minimum cost of a session between nodes u and v for duration τ, assuming that the session VC route is fixed during the session. The lower envelope function also specifies the *optimal path* for each duration. We say that path p is optimal if there is a duration τ' such that $f_p(\tau') = L_{u,v}(\tau')$. Denote by $P_{u,v}$ the set of all optimal paths between nodes u and v. Figure 1 depicts a set $P_{u,v}$ that contains four paths and the corresponding lower envelope function. In this example, the function $L_{u,v}(\tau)$ is divided into seven time intervals, where each interval starts when a path in $P_{u,v}$ becomes optimal and it ends when some other path becomes optimal. Note that a path may be optimal at more than one time interval.

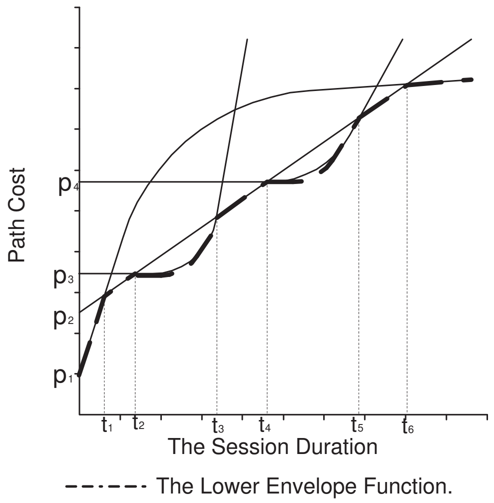

Figure 1: An example of a lower envelope function.

We assume that for each pair of nodes u and v in the graph, the set $P_{u,v}$ and the lower envelope function $L_{u,v}(\tau)$ are given. The issue of computing $P_{u,v}$ and $L_{u,v}(\tau)$ is deferred to the full version of the paper [28].

In the case of mobile users, a session σ is defined by a sequence of m triplets (u_i, v_i, t_i), $\sigma = \{(u_i, v_i, t_i)\}_{i=0}^{m}$, where m is the number of movements of the user during the session. The i-th movement of the user is represented by triplet (u_i, v_i, t_i) that indicates the movement of the user from node u_i to node v_i at time t_i. In our model, the mobile users may move freely from node to node without any limitations on the movement distances or rates.

3. COMPETITIVE ON-LINE ALGORITHMS FOR STATIC USERS

In this section we handle the case where the users are static. Initially, we determine the conditions under which competitive on-line algorithms exist. In [28] we prove that if the optimal off-line algorithm can gain from rerouting the current VC over a different route, or, if it can gain from releasing the current VC and reestablishing a new VC over the same route, then a competitive on-line algorithm does not exist. Therefore, we restrict ourselves to the case where the optimal off-line algorithm makes a single routing decision at the session initialization, and this decision is not changed during the session. However, this restriction is not sufficient for guaranteeing the existence of a competitive on-line algorithm. We also require that the cost function between every pair of nodes is non-negative, monotonically non-decreasing, and concave. Under this condition, the following property is satisfied.

PROPERTY 1. *If, for each path $p \in P_{u,v}$ the cost function is a non-negative, monotonically non-decreasing, and concave, then $L_{u,v}(t)$ is a non-negative, monotonically non-decreasing and concave cost function.*

The proof of the above is deferred to [28]. With these fairly realistic restrictions we present a 4-competitive algorithm for the case of linear cost functions in Section 3.1. In Section 3.2 we generalize this algorithm to handle arbitrary concave cost functions.

3.1 A competitive on-line algorithm for linear cost functions

We now assume that the cost function of any path $p \in P_{u,v}$ is linear, and we denote it by $f_p(t) = s(p) + h(p) \cdot t$, where both the setup cost, $s(p)$, and the hold cost, $h(p)$, are non-negative. For the case where $P_{u,v}$ contains only two possible paths a 2-competitive algorithm is implied by [21]. There, this problem is known as the *ski rental problem*. In the sequel we present a 4-competitive algorithm for the case of an unlimited number of possible paths.

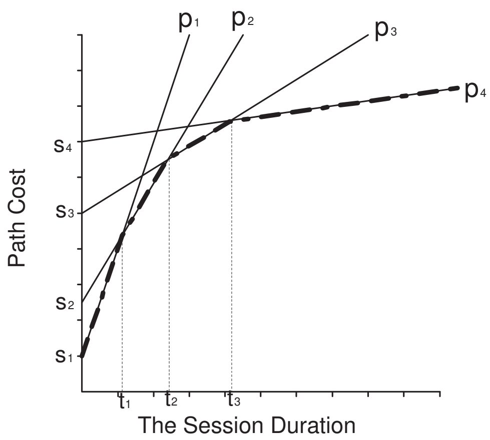

Figure 2: A lower envelope function derived from linear cost functions.

We first sort the paths $p \in P_{u,v}$ in increasing order according to the setup cost $s(p)$, as depicted in Figure 2. Hence, for each pair of paths $p_i, p_j \in P_{u,v}$, $i < j$ if and only if

$s(p_i) < s(p_j)$. Each path $p_i \in P_{u,v}$ is represented by a triplet $(s(p_i), h(p_i), t(p_i))$, where $s(p_i)$ is the setup cost, $h(p_i)$ is the hold cost, and $t(p_i)$ is the time when p_i becomes non-optimal. The following properties of the sequences $\{s(p_i)\}$, $\{h(p_i)\}$ and $\{t(p_i)\}$ are immediate.

PROPERTY 2. *$\{s(p_i)\}$ and $\{t(p_i)\}$ are monotonically increasing sequences, and $\{h(p_i)\}$ is a monotonically decreasing sequence.*

PROPERTY 3. *A path $p_i \in P_{u,v}$ is optimal for sessions whose duration $\tau \in [t(p_{i-1}), t(p_i))$.*

The proposed on-line algorithm uses the doubling technique, see e.g., [27]. During a session, the algorithm may reroute the session's VC so as to reduce the total cost of the session. The algorithm is iterative. An iteration starts when the algorithm determines a new route for the session's VC. This is determined using two principles: the *credit principle* and the *persistence principle*. Suppose that the algorithm starts a new iteration (i.e., makes a routing decision) at time t'. The credit principle states that the selected path p_j should be chosen from the set $P_{u,v}$ such that its setup cost, $s(p_j)$, is the maximal in $\{s(p_i)\}$ such that $s(p_j) \leq \alpha \cdot L_{u,v}(t')$ for a given constant $\alpha > 1$. The persistence principle states that the algorithm should maintain path p_j until time $t(p_j)$ when this path becomes non-optimal, i.e., there is a path $p_{j'}$, $j' > j$, which is optimal at time $t(p_j)$.

```
Session-Management-proc (L_{u,v}, P_{u,v})
    k = 0
    t = 0
    while ( the session has not terminated )
        k = k + 1
        If k > 1 then release path p_{i_{k-1}}
        Establish path p_{i_k} s.t.
            s(p_{i_k}) = max({s(p_j) | s(p_j) ≤ α · L_{u,v}(t)})
        Wait until t = min(τ, t(p_{i_k}))
    End while
End
```

Figure 3: The on-line algorithm $\mathcal{A}$ for linear cost functions.

In Figure 3 we give a formal description of the algorithm. Let t be a clock that measures the session duration, let k be the iteration index, let p_{i_k} denote the path selected at the kth iteration, and let $(s(p_{i_k}), h(p_{i_k}), t(p_{i_k}))$ denote the path characteristics. In addition, let $t(p_{i_0}) = 0$, $L_{u,v}(0) = s_1$, and assume that the session duration is τ.

In the sequel we show that the competitive ratio of algorithm $\mathcal{A}$ is bounded by 4. For each selected path p_{i_k}, let $c(p_k)$ denote its cost at the point of time when it becomes non-optimal, hence $c(p_k) = s(p_{i_k}) + h(p_{i_k}) \cdot t(p_{i_k})$ and it is equal to $L_{u,v}(t(p_{i_k}))$.

LEMMA 1. *For each $k > 1$, $c(p_k) > \alpha \cdot c(p_{k-1})$.*

Proof: Consider the path selected at iteration k, p_{i_k}. It is established at time $t(p_{i_{k-1}})$ and at that time its setup cost $s(p_{i_k})$ is the maximal that satisfies

$$s(p_{i_k}) \leq \alpha \cdot L_{u,v}(t(p_{i_{k-1}})) = \alpha \cdot c(p_{k-1})$$

Hence, $s(p_{i_k+1}) > \alpha \cdot c(p_{k-1})$. The path p_{i_k} is optimal until time $t(p_{i_k})$, then path p_{i_k+1} becomes optimal. By the continuity of the function $L_{u,v}(t)$ it follows that

$$c(p_k) = s(p_{i_k}) + h(p_{i_k}) \cdot t(p_{i_k+1}) =$$
$$= s(p_{i_k+1}) + h(p_{i_k+1}) \cdot t(p_{i_k}) \geq s(p_{i_k+1}) > \alpha \cdot c(p_{k-1}) \quad \square$$

COROLLARY 1. *For each $k, j \geq 1$, $c(p_{k+j}) > \alpha^j \cdot c(p_k)$.*

COROLLARY 2. *For each $n \geq 1$,* $\sum_{k=1}^{n} c(p_k) < c(p_n) \cdot \frac{\alpha}{\alpha-1}$.

Proof: $$\sum_{k=1}^{n} c(p_k) \leq \sum_{k=0}^{n-1} c(p_n) \cdot \left(\frac{1}{\alpha}\right)^k < c(p_n) \cdot \frac{\alpha}{\alpha - 1} \quad \square$$

Let $a(p_{i_k})$ be the actual cost that the algorithm $\mathcal{A}$ pays for using path p_{i_k}.

LEMMA 2. *For each $k \geq 1$, $a(p_{i_k}) \leq c(p_k)$.*

Proof: Since $\mathcal{A}$ uses the path p_{i_k} for only a duration of $t(p_{i_k}) - t(p_{i_{k-1}})$, the following inequality is satisfied,

$$a(p_{i_k}) = s(p_{i_k}) + h(p_{i_k}) \cdot (t(p_{i_k}) - t(p_{i_{k-1}})) \leq$$
$$\leq s(p_{i_k}) + h(p_{i_k}) \cdot t(p_{i_k}) = c(p_k) \quad \square$$

THEOREM 1. *For any session σ,* $Cost_{\mathcal{A}}(\sigma) \leq 4 \cdot Cost_{\mathcal{OPT}}(\sigma)$.

Proof: Consider a session between nodes u and v with duration τ, and suppose that $t(p_{i_n}) \leq \tau < t(p_{i_{n+1}})$. Hence,

$$Cost_{\mathcal{A}}(\sigma) = \sum_{k=1}^{n+1} a(p_{i_k}) =$$

$$= (\sum_{k=1}^{n} a(p_{i_k})) + s(p_{i_{n+1}}) + h(p_{i_{n+1}}) \cdot (\tau - t(p_{i_n}))$$

By Corollary 2, $\sum_{k=1}^{n} c(p_k) < c(p_n) \cdot \frac{\alpha}{\alpha-1}$. By the credit principle, $s(p_{i_{n+1}}) \leq \alpha \cdot c(p_n)$, and from the concavity of $L_{u,v}(t)$, $h(p_{i_{n+1}}) \cdot (\tau - t(p_{i_n})) \leq L_{u,v}(\tau) - L_{u,v}(t(p_{i_n}))$. Therefore,

$$\begin{aligned}
Cost_{\mathcal{A}}(\sigma) &= \sum_{k=1}^{n+1} a(p_{i_k}) \\
&= \left(\sum_{k=1}^{n} a(p_{i_k})\right) + s(p_{i_{n+1}}) + h(p_{i_{n+1}}) \cdot (\tau - t(p_{i_n})) \\
&\leq \left(\sum_{k=1}^{n} a(p_{i_k})\right) + \alpha \cdot c(p_n) + (L_{u,v}(\tau) - L_{u,v}(t(p_{i_n}))) \\
&\leq \frac{\alpha}{\alpha - 1} \cdot c(p_n) + \alpha \cdot c(p_n) + (L_{u,v}(\tau) - L_{u,v}(t(p_{i_n}))) \\
&\leq \frac{\alpha^2}{\alpha - 1} \cdot c(p_n) + (L_{u,v}(\tau) - L_{u,v}(t(p_{i_n}))) \\
&\leq \frac{\alpha^2}{\alpha - 1} \cdot L_{u,v}(\tau) = \frac{\alpha^2}{\alpha - 1} \cdot Cost_{\mathcal{OPT}}(\sigma)
\end{aligned}$$

For $\alpha > 1$, the expression $\frac{\alpha^2}{\alpha-1}$ is minimized for $\alpha = 2$. Plugging this value back into the expression yields that $Cost_{\mathcal{A}}(\sigma) \leq 4 \cdot Cost_{\mathcal{OPT}}(\sigma)$. □

3.2 A competitive on-line algorithm for concave cost functions

In the case of concave cost functions, a naive employment of algorithm $\mathcal{A}$ does not yield a competitive algorithm. This happens since the credit principle assumes that the hold cost of the paths in $P_{u,v}$ becomes lower as their setup cost becomes higher. This assumption is not valid in the concave case, as shown by Figure 4-(a). The usage of the persistence principle is also problematic since a a path may be optimal in more than one time interval, as depicted in Figure 4-(b), making it unclear how should a new path be selected. In the following, we present a new on-line algorithm, referred as algorithm $\mathcal{B}$. It maps each concave cost function to a set of upper bounding linear functions. These linear functions are used as input for algorithm $\mathcal{A}$. We show that the resulting algorithm is 4-competitive with respect to the original (concave) cost functions.

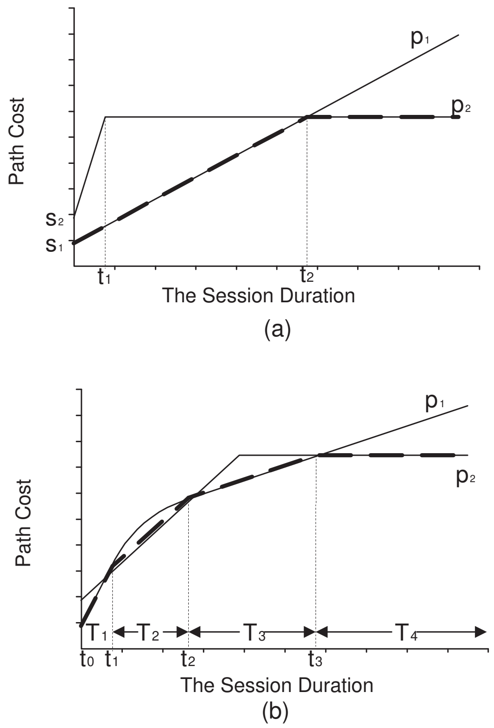

Figure 4: Examples of concave cost functions.

We assume without loss of generality that each $p \in P_{u,v}$ is optimal only during a single time interval. If a path $p \in P_{u,v}$ is optimal in several time intervals, like the case of Figure 4-(b), then we replicate this path, and define each copy to be optimal at a different time interval. The time intervals are enumerated in increasing order starting from 1 to m, where m is the index of the last time interval. The first interval starts at time $t_0 = 0$, each time interval $k < m$ starts at time t_{k-1} and terminates at time t_k, and the last time interval is marked by m ($t_m = \infty$). Hence, the sequence $\{t_k\}_{k=1}^{m-1}$ is monotonically increasing. Let $\hat{p_k}$ be the optimal path during the kth time interval. For each time interval $k < m$ and path $\hat{p_k}$, we define a linear function $\hat{f_k}(t)$ which is the left tangent of $L_{u,v}$ at the point t_k. The linear function $\hat{f_m}(t)$ of the last time interval is defined as the right tangent of $L_{u,v}$ at the point t_{m-1}. An example of such a mapping is given in Figure 5. There, p_1 is optimal during the first and third time intervals, and p_2 is optimal during the second and fourth time intervals. Therefore, $\hat{p_1} = \hat{p_3} = p_1$ and $\hat{p_2} = \hat{p_4} = p_2$. Algorithm $\mathcal{B}$ routes a session VC as algorithm $\mathcal{A}$ would route the VC of a session with the same duration, given a set of paths $\{\hat{p_k}\}$ and corresponding cost functions $\{\hat{f_k}(t)\}$.

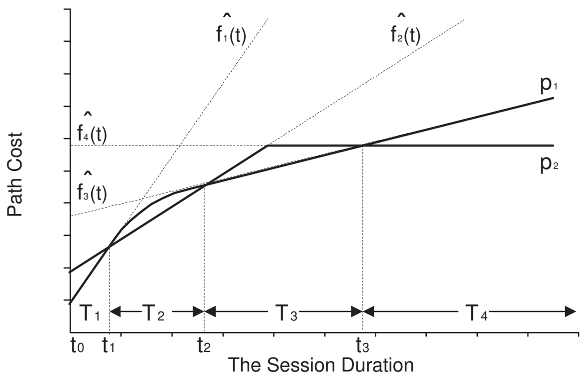

Figure 5: Example of mapping concave cost functions to linear functions.

We now prove that algorithm $\mathcal{B}$ is 4-competitive. Consider a session σ which has duration τ. We denote by $Cost_{\mathcal{OPT}}(\sigma)$ the cost of a session with respect to the optimal off-line algorithm, $Cost_{\mathcal{OPT}}(\tau) = L_{u,v}(\tau)$. Let $Cost_{\mathcal{A}}(\sigma)$ be the cost of a session σ according to algorithm $\mathcal{A}$, when the cost function of each path $\hat{p_k}$ is given by the linear function $\hat{f_k}(t)$, and let $Cost_{\mathcal{B}}(\sigma)$ be the cost of a session σ according to algorithm $\mathcal{B}$, when the cost function of each path $\hat{p_k}$ is given by the concave cost function $f_k(t)$,

LEMMA 3. *For any k and $t \geq 0$, $\hat{f_k}(t) \geq f_k(t)$. For $k < m$, $\hat{f_k}(t_k) = f_k(t_k)$*

Proof: For $k < m$, $\hat{f_k}(t)$ is the left tangent of $L_{u,v}$ at the point t_k. Since, $f_k(t) = L_{u,v}(t)$ for any $t \in [t_{k-1}, t_k]$, $\hat{f_k}(t_k) = f_k(t_k)$. Moreover, $f_k(t)$ is a concave function, hence $\hat{f_k}(t) \geq f_k(t)$. In a similar fashion, $\hat{f_m}(t) \geq f_m(t)$. □

LEMMA 4. *For any session σ, $Cost_{\mathcal{A}}(\sigma) \leq 4 \cdot Cost_{\mathcal{OPT}}(\sigma)$.*

Proof: The proof is identical to the proof of Theorem 1. □

THEOREM 2. *For any session σ, $Cost_{\mathcal{B}}(\sigma) \leq 4 \cdot Cost_{\mathcal{OPT}}(\sigma)$.*

Proof: For each session σ, algorithm $\mathcal{B}$ uses the same paths as algorithm $\mathcal{A}$ and for the same duration. Since, $\hat{f}_k(t) \geq f_k(t)$ during each time interval (according to Lemma 3) $Cost_{\mathcal{B}}(\sigma) \leq Cost_{\mathcal{A}}(\sigma)$. According to Lemma 4, $Cost_{\mathcal{A}}(\sigma) \leq 4 \cdot Cost_{\mathcal{OPT}}(\sigma)$. Therefore, $Cost_{\mathcal{B}}(\sigma) \leq 4 \cdot Cost_{\mathcal{OPT}}(\sigma)$. □

4. COMPETITIVE ON-LINE ALGORITHMS FOR MOBILE USERS

In this section we consider session management with mobile users. In this case, the unknown information is not only the session duration, but also the future movements of the mobile participants. We start by proving that for general graphs and arbitrary edge cost functions with positive setup cost, the competitive ratio of the best on-line algorithm is at least $\Omega(\log n)$, where n is the number of nodes in the network (Section 4.1). In the sequel, we limit the discussion to linear cost functions of the form $f_e(\tau) = s_e + h_e \cdot \tau$ for each edge $e \in E$. We consider several special cases where constant competitive on-line algorithms do exist. To this end, let us first examine what are the properties of an optimal off-line algorithm in this case. A session management algorithm is called *lazy* if it changes the VC route only as a response to a movement of a mobile user, and at the time of the movement.

THEOREM 3. *If the edge cost functions are linear, then there is a lazy optimal off-line algorithm.*

The special cases we consider are: tree topology (Section 4.2), ring topology (Section 4.3), and general graphs with limited movement rate (Section 4.4).

4.1 Lower bound for general graphs

The mobile case in general graphs can be related to the on-line Steiner tree problem by viewing each movement as corresponding to a node leaving the tree and another node joining it. The proof of the following theorem is similar to the one presented in [25]. The details are deferred to [28].

THEOREM 4. *Consider a general graph, mobile users, and arbitrary edge cost functions with positive setup cost. Then, the competitive ratio of the best on-line algorithm is at least $\Omega(\log n)$, where n is the number of nodes in the network.*

4.2 An on-line algorithm for a tree topology

Managing a session in a graph with a tree topology may not seem a difficult task, since there is a unique path between every pair of nodes. However, the following example shows that session resources need to be released judiciously, otherwise the competitive factor is not bounded.

EXAMPLE 1. *Consider a graph with two nodes u and v and a single edge e between them. Now refer to a session between two users. One of them is static and is attached to node u. The other is mobile, and it moves as follows: At the session initialization it is located at node v. Each time a VC is established over edge e it moves to node u. When the VC resources are released the mobile user returns to node v.*

The session algorithm may use one of the following strategies: (a) release the edge resources whenever they are not in use; (b) maintain unused resources in case the mobile user returns to node v. It is not hard to see that both strategies are not competitive.

Our algorithm, referred to as Algorithm $\mathcal{T}$, uses the *postponement principle* for determining the time for releasing unused resources. The release of VC resources over an edge e is postponed by a time period of s_e/h_e with respect to the last time they were in use. This period is called the *postponed period*, and the edge is called a *postponed edge*. If, during that period, the edge resources are required again, then there is no setup cost. The only cost incurred is the cost of maintaining the resources during the time they are not used. Thus, the edge resources are released precisely when the maintenance cost is equal to the setup cost. In this case, the postponed period is called *redundant*. In the following, we use the term *path activation* for establishing a VC over a given path p, which may include postponed edges as well as edges without any allocated resources. The activation cost of each edge $e \in p$ at a given time t is calculated as follows. Let $\tau_e(t)$ be the period that has elapsed since the last time edge e was in use until time t, or $\tau_e(t) = \infty$ if it was never used before. If e is a postponed edge then its activation cost is $h_e \cdot \tau_e(t)$. Otherwise, the activation cost is the sum of two components: the edge setup cost s_e and the cost of the redundant postponed period before the edge resources are released. The cost of this period is $h_e \cdot (s_e/h_e) = s_e$. As a result, the cost of activating path p at time t is:

$$Active\_Cost(p,t) = \sum_{e \in p} \begin{cases} h_e \cdot \tau_e(t) & \tau_e \leq s_e/h_e \\ 2 \cdot s_e & \text{otherwise} \end{cases} \tag{1}$$

The hold cost of each edge remains unchanged. The proofs of the following theorems can be found in [28].

THEOREM 5. *Algorithm $\mathcal{T}$ is 2-competitive.*

THEOREM 6. *The competitive ratio of any on-line algorithm is at least 2.*

We note that Algorithm $\mathcal{T}$ is also suitable for sessions with multiple participants (multicast session). Theorems 5 and 6 still hold in this case.

4.3 An on-line algorithm for a ring topology

In this section we present a 4-competitive algorithm, referred to as Algorithm $\mathcal{R}$, for session management in a ring topology with linear cost functions. In such a graph, at every step, the algorithm is required to determine both the VC path and when to release unused edge resources. The proposed algorithm uses the *postponement principle* for handling unused resources, as described in Section 4.2, where a path activation cost is defined by Equation 1. For determining the VC path during the session the algorithm uses a retrospective approach [21]. It keeps track of the past and routes the VC in accordance with the routing decisions made by $\mathcal{OPT}$.

Our algorithm uses Theorem 3 for estimating the possible decisions of algorithm $\mathcal{OPT}$. This theorem states that there is a lazy off-line algorithm $\mathcal{OPT}$ for optimal session management. Moreover, $\mathcal{OPT}$ knows the future movements of the users, and when edge resources become unused it can determine whether to release them immediately or to maintain them as postponed resources until they are needed again. Hence, the activation cost of a path p at time t according to $\mathcal{OPT}$ is

$$Active\_Cost^*(p,t) = \sum_{e \in p} \begin{cases} h_e \cdot \tau_e(t) & \tau_e \leq s_e/h_e \\ s_e & Otherwise \end{cases} \qquad (2)$$

Algorithm $\mathcal{R}$ maintains a *decision tree* that represents all possible routing options and their cost. A *routing option* is a sequence of routing decisions that defines the VC path after the movement of each participant until a given time t. Thus, each possible routing option is a path in the decision tree from the root to a leaf. At the session initialization the tree contains only two routing options. After each movement of a user, each path in the decision tree splits into two new paths which correspond to the two new routing options. For every routing option z, let $p(z,t)$ be its VC path at time t, and let $c(z,t)$ be its accumulated cost until time t, calculated according to Equation 2. A routing option is called *optimal* at time t if it has the minimal accumulated cost until that time. We denote by $\tilde{z}(t)$ the optimal routing option at time t, and let $\tilde{p}(t)$ and $\tilde{c}(t)$ be its VC path and its accumulated cost at time t correspondingly. At each given time t, algorithm $\mathcal{R}$ routes the session VC over the path $\tilde{p}(t)$ defined by optimal routing option. It is clear from this description, that Algorithm $\mathcal{R}$ may follow several routing options during a session, adapting the behavior of new routing option when it becomes optimal, as describe by the next example.

We remark that in practice, the decision tree is required to represent only routing options that may become optimal in the future. Thus, there is no need to maintain all possible routing options.

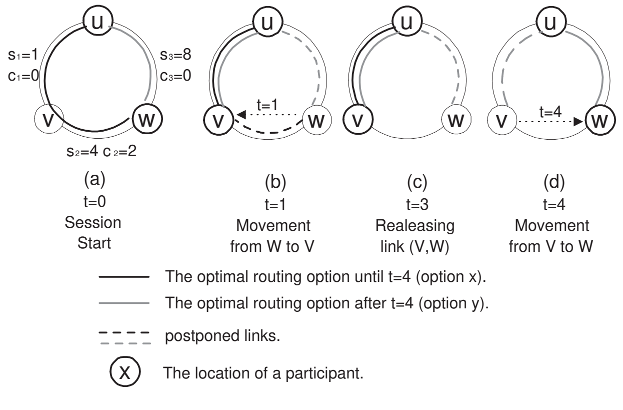

Figure 6: An example of a session in a ring.

EXAMPLE 2. *In this example, a routing option becomes optimal due to an activation of a postponed edge. Consider a ring with three nodes as depicted in Figure 6, when the setup cost and hold cost are denoted over the edges. Now,*

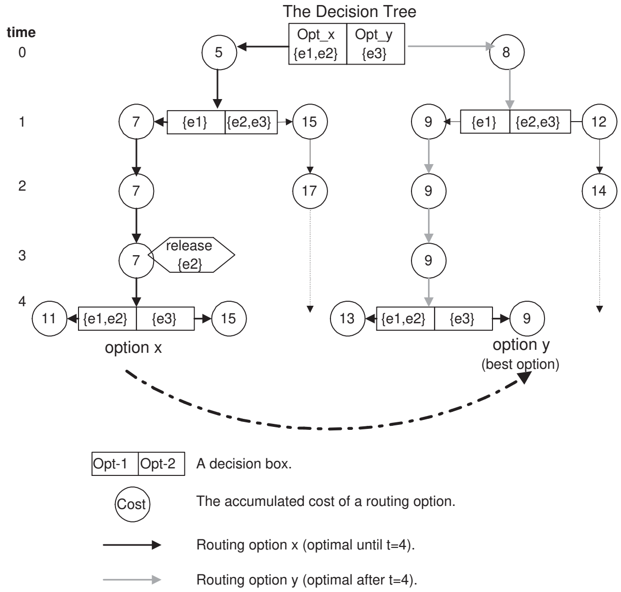

Figure 7: The decision tree of the session.

refer to a session between two users. One of them is static and it is attached to node u. The other is mobile, and it is located at node w at time $t_0 = 0$ when the session starts. At time $t_1 = 1$, the mobile user moves to node v, and at time $t_2 = 4$ it returns to node w. It is clear from the decision tree in Figure 7, that establishing a VC over edge (u,w) at time t_0, and a VC over edge (u,v) at time t_1 is the best routing option (option y), but this is revealed only at time $t_2 = 4$ when the mobile users returns to node w.

Example 2 shows us that two different routing options may both be optimal, one before time t and the other after time t, even if they both route the VC over the same path just before time t. This results from using two different VC paths in the ring in the past. We should be aware of this fact during the cost analysis.

In the following we prove that algorithm $\mathcal{R}$ is 4-competitive. This proof contains two steps. First we assume that $\mathcal{R}$ is not required to pay for redundant postponed periods. Hence, the cost of activating a path is given by Equation 2. Later on, we calculate the contribution of these redundant postponed periods to the session cost.

Let T be a sequence of time intervals that specifies when an edge e was used during a given session. We denote by $Cost^*_T(e)$ the accumulated cost of using edge e during these time intervals, where the activation cost is defined by Equation 2.

LEMMA 5. *Let T_1 and T_2 be two sequences of time intervals such that $T_1 \subseteq T_2$, then for every edge $e \in E$, $Cost^*_{T_1}(e) \leq Cost^*_{T_2}(e)$*

Consider a session σ that starts at time zero. We denote by $Cost^*_{\mathcal{R}}(\sigma,t)$ the session cost with respect to algorithm $\mathcal{R}$

until time t, when the activation cost of a path is calculated according to Equation 2. The proofs of the following two lemmas can be found in [28].

LEMMA 6. *For every session σ and a given time $t_{end} \geq 0$,* $Cost^*_{\mathcal{R}}(\sigma, t_{end}) \leq 2 \cdot \tilde{c}(t_{end})$.

LEMMA 7. *For every session σ and a given time $t_{end} \geq 0$,* $Cost_{\mathcal{R}}(\sigma, t_{end}) \leq 4 \cdot \tilde{c}(t_{end})$.

THEOREM 7. *For every session σ,* $Cost_{\mathcal{R}}(\sigma) \leq 4 \cdot Cost_{\mathcal{OPT}}(\sigma)$.

Proof: On one hand, from Lemma 7, we get that

$$Cost_{\mathcal{R}}(\sigma, t_{end}) \leq 4 \cdot \tilde{c}(t_{end})$$

On the other hand, the routing option of $\mathcal{OPT}$ is also included in the decision tree of the session. Therefore, for any given time $t \geq 0$, $Cost_{\mathcal{OPT}}(\sigma, t) \geq \tilde{c}(t)$. As a result, $Cost_{\mathcal{R}}(\sigma) \leq 4 \cdot Cost_{\mathcal{OPT}}(\sigma)$. □

4.4 An algorithm for general graphs with limited movement rate

Finally, we present an on-line algorithm for general graphs which have the following two properties. First, For every edge $e \in E$ the ratio h_e/s_e is bounded by two constants c_1 and c_2, such that $c_1 \leq h_e/s_e \leq c_2$. Second, the user movement rate is limited. Each mobile user must remain at each visited node for at least a time period of $1/c_1$ before it may move to another node.

In this part we present a lazy on-line algorithm for managing a session, which is called Algorithm $\mathcal{H}$. The algorithm updates the session VC each time a user changes its location. Since the user movement rate is limited, edge resources are released when they become unused. Maintaining resources for a period longer then $1/c_1$ would cost more than releasing them and reallocating them again when needed. Consider a path p. Let $h(p) = \sum_{e \in p} h_e$ and $s(p) = \sum_{e \in p} s_e$ be the *hold cost* and the *setup cost* of path p, respectively. For every pair of nodes u and v, let the *shortest path* between them be the path which has *minimum setup cost.* Denote this path by $\tilde{p}_{u,v}$ and its setup cost by $\tilde{s}(u, v)$. In addition, let $\tilde{h}(u, v)$ be the *minimum hold cost* between nodes u and v. Note that the hold cost of the shortest path, $\tilde{p}_{u,v}$, may be more than $\tilde{h}(u, v)$. However, in most practical networks, there is a correlation between the setup cost and the hold cost of all the graph edges and constants c_1 and c_2 are close. Here, we assume that the hold cost of the shortest path between a pair of nodes is close to the minimal hold cost.

Algorithm $\mathcal{H}$ works as follows. During the session initialization it establishes a VC over the shortest path between the users. Now, suppose that during a session one of the users moves from node w to node u, while the other user is attached to node v. The algorithm finds the path p which is obtained by concatenating the shortest path between nodes w and u, $\tilde{p}_{w,u}$, to the current VC. If the setup cost of p is not more than α times the setup cost of the shortest path between the two users, $\tilde{s}(u, v)$, then the path $\tilde{p}_{w,u}$ is established and it becomes part of the VC route. Otherwise, the current VC is released and a new VC over the shortest path $\tilde{p}_{u,v}$ is established. A formal description of the algorithm is given in Figure 8, where $p_{w,v}$ is the VC path before the movement.

Upon the session initialization between nodes u and v do:
  Establish a VC over the path $\tilde{p}_{u,v}$

Upon a movement from node w to u when the second user is attached to node v do:
  $p \leftarrow p_{w,v} \cup \tilde{p}_{w,u}$
  If $(s(p) \leq \alpha \cdot \tilde{s}(u, v))$ then
    Establish a VC over the path $\tilde{p}_{w,u}$
    Add $\tilde{p}_{w,u}$ to the VC path $p_{u,v}$
  Else
    Release the current VC
    Establish a new VC over the path $\tilde{p}_{u,v}$

Figure 8: A formal description of Algorithm $\mathcal{H}$.

The algorithm uses the *credit principle.* It attempts to minimize the setup cost of each VC update operation under the constraint that the VC total setup cost is at most α times the setup cost of the shortest path between the users. We call α the *credit parameter.* The credit principle guarantees that the hold cost of $\mathcal{H}$ will not exceed $\frac{c_2}{c_1} \cdot \alpha$ times the hold cost of $\mathcal{OPT}$. In the sequel we show that a proper selection of the parameter α yields a small competitive ratio.

We turn to prove that the competitive ratio of the algorithm is $1 + 2 \cdot \frac{c_2}{c_1}$. Consider a session σ that starts at time zero and is defined by a sequence of m triplets, $\sigma = \{(w_i, u_i, t_i)\}_{i=0}^m$, where the i-th triplet represents a movement of a mobile user from node w_i to node u_i at time t_i, as described in Section 2. For simplifying the analysis of the competitive ratio, we also consider both the initialization and the termination of a session as movements. We assume that before a session starts both users are attached to node u_0, and at time zero one of them moves to node v_0. Similarly, the session terminates at time τ, when one of the users moves to the node attached by the other, and they both do not change their attachment point anymore.

We denote by $Hold\_Cost_{\mathcal{H}}(\sigma)$ and $Hold\_Cost_{\mathcal{OPT}}(\sigma)$ the total hold cost of the session σ according to the algorithms $\mathcal{H}$ and $\mathcal{OPT}$ correspondingly. In addition, we denote by $Setup\_Cost_{\mathcal{H}}(\sigma)$ and $Setup\_Cost_{\mathcal{OPT}}(\sigma)$ the total cost of the VC setup operations of both algorithms.

LEMMA 8. *For every session σ,* $Hold\_Cost_{\mathcal{H}}(\sigma) \leq \frac{c_2}{c_1} \cdot \alpha \cdot Hold\_Cost_{\mathcal{OPT}}(\sigma)$.

Proof: For every edge $e \in E$, $c_1 \leq h_e/s_e \leq c_2$. Therefore, for every pair u and v, $\tilde{s}(u, v) \leq \tilde{h}(u, v)/c_1$. In addition, for every path p, $h(p) \leq c_2 \cdot s(p)$. By the credit principle, at any time during the session, the VC route, p, satisfies $s(p) \leq \alpha \cdot \tilde{s}(u, v)$, where the users are attached to nodes u and v. Hence,

$$h(p) \leq c_2 \cdot s(p) \leq c_2 \cdot \alpha \cdot \tilde{s}(u, v) \leq \frac{c_2}{c_1} \cdot \alpha \cdot \tilde{h}(u, v)$$

proving the lemma. $\square$

Consider the i-th movement from node w_i to node u_i. Let $s(M_i) = \tilde{s}(w_i, u_i)$ be the setup cost of the shortest path between these nodes, called the *movement cost*, and let $s(M) = \sum_{i=0}^{m} s(M_i)$.

LEMMA 9. *For every session σ,*
$Setup\_Cost_{\mathcal{OPT}}(\sigma) \geq \frac{s(M)}{2}$.

Proof: First, assume that $\mathcal{OPT}$ pays for allocating the resources of an edge in two installments: the first half is paid for at the time of allocation, and the second half is paid for when the edge resources are released. Now consider a user's movement from node w_i to node u_i, and let us bound its contribution to the total setup cost of $\mathcal{OPT}$. As a result of this movement, part of the VC route between node w_i and some node x is released, and a new path is established between nodes x and u_i. Hence, the setup cost of this movement is at least $(\tilde{s}(w_i, x) + \tilde{s}(x, u_i))/2$. By the triangle inequality, we get that $\tilde{s}(w_i, x) + \tilde{s}(x, u_i) \geq \tilde{s}(w_i, u_i)$. Therefore, the total cost is $Setup\_Cost_{\mathcal{OPT}}(\sigma) \geq \sum_{i=0}^{m} \frac{\tilde{s}(w_i,u_i)}{2} = \frac{s(M)}{2}$ $\square$

LEMMA 10. *For every session σ,*
$Setup\_Cost_{\mathcal{H}}(\sigma) \leq \frac{\alpha}{\alpha-1} \cdot s(M)$.

Proof: We partition the session into phases so as to calculate the total setup cost of the session. The first phase, called phase 0, begins at the session initialization. A new phase begins each time the algorithm decides to release the current VC and to establish a new one over the shortest path. Suppose that the session contains K connection reestablishment operations ($K + 1$ phases). Let $s(p_k)$ be the setup cost of the VC path that is established at the beginning of phase k, and let $s(M_k)$ be the sum of all the movement costs that are made during phase k. Note that $s(p_0) = 0$, since we consider the session initialization as a movement. According to the credit principle,

$$\begin{aligned}
s(p_k) &\leq \frac{1}{\alpha} \cdot [s(M_{k-1}) + s(p_{k-1})] \\
&\leq \frac{1}{\alpha} \cdot s(M_{k-1}) + \frac{1}{\alpha^2} \cdot [s(M_{k-2}) + s(p_{k-2})] \\
&\leq \frac{1}{\alpha} \cdot s(M_{k-1}) + \frac{1}{\alpha^2} \cdot s(M_{k-2}) + \\
&\quad + \frac{1}{\alpha^3} \cdot [s(M_{k-3}) + s(p_{k-3})] \\
&\leq \frac{1}{\alpha} \cdot s(M_{k-1}) + \frac{1}{\alpha^2} \cdot s(M_{k-2}) + \frac{1}{\alpha^3} \cdot s(M_{k-3}) + \\
&\quad + \cdots + \frac{1}{\alpha^k} \cdot s(M_0) \\
&\leq \sum_{j=0}^{k-1} \frac{1}{\alpha^{k-j}} \cdot s(M_j)
\end{aligned}$$

Thus, the total cost of the VC's that are established at the connection reestablishment operations is

$$\sum_{k=1}^{K} s(p_k) = \sum_{k=1}^{K} \sum_{j=0}^{k-1} \frac{1}{\alpha^{k-j}} \cdot s(M_j) = \sum_{j=0}^{K-1} s(M_j) \cdot \sum_{k=1}^{K-j} \frac{1}{\alpha^k} \leq$$

$$\leq \left(\sum_{j=0}^{K-1} s(M_j) \right) \cdot \left(\sum_{k=1}^{\infty} \frac{1}{\alpha^k} \right) \leq \frac{1}{\alpha - 1} \cdot s(M)$$

Hence, the total setup cost is

$$Setup\_Cost_{\mathcal{H}}(\sigma) \leq \sum_{k=0}^{K} [s(p_k) + s(M_k)] \leq$$

$$\leq \sum_{k=0}^{K} s(p_k) + \sum_{k=0}^{K} s(M_k) \leq$$

$$\leq \frac{1}{\alpha - 1} \cdot s(M) + s(M) \leq \frac{\alpha}{\alpha - 1} \cdot s(M)$$

$\square$

THEOREM 8. *Algorithm $\mathcal{H}$ is $(2 + \frac{c_2}{c_1})$-competitive for*
$\alpha = 1 + 2 \cdot \frac{c_1}{c_2}$.

Proof: The total cost of a session σ is the sum of two components, the setup cost and the hold cost. According to Lemma 8, the competitive ratio of the hold cost is

$$\frac{Hold\_Cost_{\mathcal{H}}(\sigma)}{Hold\_Cost_{\mathcal{OPT}}(\sigma)} \leq \frac{c_2}{c_1} \cdot \alpha$$

According to Lemma 9 and Lemma 10, the competitive ratio of the setup cost is

$$\frac{Setup\_Cost_{\mathcal{H}}(\sigma)}{Setup\_Cost_{\mathcal{OPT}}(\sigma)} \leq \frac{\frac{\alpha}{\alpha-1} \cdot s(M)}{\frac{1}{2} \cdot s(M)} = \frac{2 \cdot \alpha}{\alpha - 1}$$

The value of α that minimizes the competitive ratio of both components is defined by the equation

$$\frac{2 \cdot \alpha}{\alpha - 1} = \frac{c_2}{c_1} \cdot \alpha$$

Hence, the best competitive ratio is obtained by $\alpha = 1+2\cdot\frac{c_1}{c_2}$, and the its value is $2 + \frac{c_2}{c_1}$. $\square$

COROLLARY 3. *If $c_1 = c_2$ then Algorithm $\mathcal{H}$ is 3-competitive for $\alpha = 3$.*

5. SUMMARY

We presented dynamic algorithms for session management when users may be both static or mobile. In the case of static users, previous works [11, 12] have shown that the network throughput can be increased by employing dynamic session management algorithms that reroute VC's during the sessions. In addition methods for performing a transparent VC rerouting without losing packets or changing their order was proposed in [15]. This paper shows that VC rerouting techniques can also be used for reducing the total cost of sessions by using on-line approach. We presented a 4-competitive algorithm for the situation where the path cost functions are concave.

In the presence of mobile users, dynamic session management schemes are essential for maintaining continuous connections between mobile users. However, in this case, finding an efficient method is a much harder task. We showed that the session management problem resembles the on-line Steiner tree problem. We proved that for a general graph and arbitrary edge cost functions with positive setup cost, the competitive ratio of the best on-line algorithm is at least $\Omega(\log n)$, where n is the number of nodes in the network. However, there are competitive on-line algorithms for several practical cases, where the edge cost functions are linear. We presented a 2-competitive algorithm for a tree topology, and a 4-competitive for a ring topology. We also gave a $(\frac{c_2}{c_1}+2)$-competitive algorithm for general graphs, where for every edge e, the ratio between its setup cost, s_e, and its hold cost, h_e, is lower and upper bounded by c_1 and c_2 respectively, and the user movement rate is at most $1/c_1$. This algorithm is especially important since it fits certain practical cellular systems.

6. REFERENCES

[1] W. Stallings. ISDN and broadband ISDN. 2nd edition, *Macmillan Publishing Company*, 1992.

[2] D. E. McDysan and D. L. Spohn. ATM theory and application. *McGraw-Hill*, 1994.

[3] C. Huitema. Routing in the Internet. *Prentice-Hall*, 1995.

[4] R. Braden, D. Clark and S. Shenker. Integrated services in the Internet architecture: an overview. RFC 1633, June 1994.

[5] . Mouly and M. B. Pautet. The GSM system for mobile communication. M. Mouly, 49 rue Louise Bruneau, Palaiseau, France 1992.

[6] D. C. Cox. Wireless personal communication : what is it? IEEE Personal Communication, April 1995.

[7] E. Padgett, C. G. Gunther and T. Hattori. Overview of wireless personal communications. IEEE communication magazine, January 1995 Vol 33 No 1.

[8] . E. Streenstrup. Routing in communication networks. *Prentice-Hall*, 1995.

[9] D. Bertsekas, R. Gallager. Data networks. 2nd Edition, *Printice-Hall*, 1992.

[10] I. F. Akyildiz, J. S. M. Ho and M. Ulema. Performance analysis of the anchor radio system handover method for personal access communications system. Proc. IEEE INFOCOM, 1996.

[11] M. H. Ackroyd. Call repacking in connecting networks. IEEE Trans. on Commun. Vol. 27, No. 3, March 1979.

[12] E. W. M. Wong, A. K. M. Chan and T. S. P. Yum. Re-routing in circuit switched networks. Proc. IEEE INFOCOM, 1997.

[13] A. Girard and S. Hurtubise. Dynamic routing and call repacking in circuit-switched networks. IEEE Trans. on Commun. Vol. 31, No. 12, December 1983.

[14] K. C. Lee and V. O. K. Li, A circuit rerouting algorithm for all-optical wide-area networks. Proc. IEEE INFOCOM, 1994.

[15] R. Cohen. Smooth intentional rerouting and its applications in ATM networks. Proc. IEEE INFOCOM, 1994.

[16] B. Awerbuch, Y. Azar, S. Plotkin and O. Waarts. Competitive routing of virtual circuits with unknown duration Proc. of 5th SODA (1994), 321-327.

[17] B. A. J. Banh, G. J. Anido and E. Dutkiewicz. Handover re-routing schemes for connection oriented services in mobile ATM networks. Proc. IEEE INFOCOM, 1998.

[18] K. Keeton, B. A. Mah, S. Seshan, R. H. Katz and D. Ferrari. Providing connection-oriented network services to mobile hosts. Proc. of the USENIX Symp. On Mobile and Location-Independent Computing, Cambridge Massachusetts, August 1993.

[19] C. K. Toh The design and implementation of a hybrid handover protocol for multi-media wireless LANs. Proc. ACM MOBICOM 95, 1995.

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

[21] S. Irani and A. R. Karlin. On Online Computation. Chapter 13 in "Approximation algorithms for NP-hard problems", Edited by D. S. Hochbaum, *PWS Publishing Company*, 1996.

[22] Y. Azar, Y. Bartal, E. Feuerstein, A. Fiat, S. Leonardi and A. Rosen. On capital investment. Algorithmica, Vol. 25, no. 1, pp. 22-36, 1999.

[23] R. El-Yaniv, R. Kaniel and N. Linial. Competitive optimal on-line leasing. Algorithmica, Vol. 25, no. 1, pp. 116-140, 1999.

[24] Y. Bejerano, I. Cidon, and J. Naor. Efficient handoff rerouting algorithms: a competitive on-line algorithmic approach, Proc. IEEE INFOCOM-2000. March 2000.

[25] M. Imaze and B. M. Waxman. Dynamic steiner tree problem. SIAM Journal on Discrete Mathematics 4:369-384, 1991.

[26] N. Alon and Y. Azar. On-line steiner trees in the Euclidean plane. Proc. of 8th Computational Geometry (1992), 337-343.

[27] Y. Azar. On-line load balancing. Chapter 8 in "Online Algorithms, State of the Art", Edited by A. Fiat and G. J. Woeginger, Lecture Notes in Computer Science, Vol. 1442, Springer-Verlag, 1998.

[28] Y. Bejerano, I. Cidon, and J. Naor. Dynamic Session Management for Static and Mobile Users: A Competitive On-Line Algorithmic Approach. Research Report, Center for Communication and Information Technologies, Technion Haifa, Israel. CC PUB #291, August 1999.