Chapter 11

COMPETITIVE ANALYSIS OF HANDOFF REROUTING ALGORITHMS

Yigal Bejerano*, Israel Cidon† and Joseph (Seffi) Naor†
* *Bell Laboratories, Lucent Technologies*
† *Technion, Israel Institute of Technology*

11.1 INTRODUCTION

Personal Communication Systems (PCS) enable people and devices to communicate independently of their location and while moving from place to place. For providing continuous communication to mobile users every PCS network employs a *mobility management* composed of two components, *location management* and *handoff management*. In contrast to the telephone number in traditional telecommunication systems that specifies the location of the end user, the PCS subscriber number does not provide the location of the mobile user. Therefore, the system must maintain a *location management* mechanism for locating mobile users. This mechanism maps subscriber numbers to the current location of the requested users for call delivery operations. The *handoff management* enables the PCS network to maintain sessions with mobile users while they

change their attachment points with the system's infrastructure. Such changes are called *handoff* or *handover* operations [8, 30]. In this chapter we consider only rerouting algorithms for supporting handoff operations. More general overviews of mobility management and handoff operations can be found in [6, 30, 33]. In our discussion we consider only inter-cell handoff operations that result from user movements to new cells[1], that include actions at both the wireless level and the network infrastructure. From the wireless perspective, the handoff mechanism determines the next serving base station and allocates new wireless resources for the session. The network needs to ensure the continuity of the traffic flow without interruptions or disordering the packets.

Most modern communication systems that are used as an infrastructure for PCS networks [3, 13, 22, 23, 25, 27] are based on connection-based technologies such as telephone and ISDN technologies [28], Frame-Relay [21], ATM networks [21] and more recently MPLS Networks [12]. Such networks require the establishment of a *virtual channel* (VC) between the session participants and maintaining it during the life of the session. In PCS networks, each time a session participant performs a handoff operation, the session VC must be modified for maintaining end-to-end connectivity. Thus, using efficient handoff rerouting algorithms is important for an efficient management of PCS networks. The effectiveness of a handoff management is evaluated by its ability to use efficiently the network resources while ensuring minimal disruption to the ongoing sessions. We classified the criteria for evaluating the handoff algorithms into two components. The *setup cost* represents the cost associated with the handoff operations, in particular signaling cost and handoff latency, and the *hold cost* determines the expense related to the use of network resources held by the VC. The *overall cost* of a session is defined to be the sum of its setup and hold costs.

In this chapter we describe currently used handoff rerouting algorithms and discuss their efficiency in term of overall cost in comparison to an optimal solution. We show that the ratio between the overall session cost achieved by these algorithms and the cost of an optimal solution can be very large (to be defined precisely later) in the worst case. Then, we introduce new handoff rerouting algorithms for which we can bound this ratio in both the worst case and average case. We use analytical

[1] another type of handoff is intra-cell handoff that results from the deterioration of radio cannel quality beyond a determined threshold due to fading effects or re-assignment of wireless cell resources.

proofs for the first result and simulation models for the latter result.

11.1.1 On-Line Algorithms and Competitive Analysis

Our goal, as already mentioned, is to reduce the overall session cost. Optimizing the VC routing for this long term goal, after each movement of a user, is a complicated task, since the handoff algorithm is typically not able to predict the session duration and the future movements of the users. Thus, the problem of minimizing the overall session cost is an on-line dynamic decision problem, where decisions are based on the current state of the network without knowledge of future events.

The difficulties that an algorithm faces are illustrated in the following example. Consider two static users that can be connected by one of two 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, an optimal path cannot be chosen by the algorithm at the session initialization. Nevertheless, our algorithm has to make decisions without prior knowledge of the session duration.

A common way for measuring the performance of an on-line algorithm is *competitive analysis*. The idea is to compare the costs associated with an on-line algorithm with the costs spent by an optimal off-line algorithm that has complete knowledge of the future. The maximum ratio between their respective costs, taken over all possible input sequences, is called the *competitive ratio*. It guarantees an upper bound on the worst case performance with respect to an optimal off-line algorithm. For the above example, the on-line strategy for this problem that yields the best competitive ratio routes the session VC over the first path for a duration of ten minutes and then reroutes 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 [11, 17].

In recent years, competitive analysis was extensively used for analyzing the performance of various algorithms for different communication problems, such as call admission, circuit routing, scheduling and load

balancing. Extensive surveys of this area are given in [11, 17]. An alternative approach to measuring the quality of on-line algorithms is through *average case* analysis, which relies on some hypothesis on the distribution of the input. Each of the two approaches has clear advantages as well as limitations, and the reader is referred to [11] for a related discussion.

11.1.2 Competitive Handoff Rerouting Algorithms

This chapter provides a worst-case analysis of handoff rerouting algorithms for general communication networks. We distinguish between two types of algorithms: *handoff anticipating* versus *handoff non-anticipating* rerouting algorithms. Algorithms of The first type are permitted to hold unused resources anticipating that these resources will be used later on, while algorithms of the latter type are not allowed to do so. By slightly increasing the session hold cost, the handoff anticipating algorithms may significantly reduce the setup cost, as we demonstrate in the following example. Consider a session in which one of the participants frequently moves between two adjacent radio cells. As a result, after each such movement the session VC is modified, and some of the previous VC resources become unused until the next movement. If the link setup costs is significantly higher than their hold costs[2] then the session overall cost can be reduced by keeping the two VC that end at each radio cell, rather than releasing and allocating the same resources again and again. However, holding unused resources may hurt the network utilization and increase the call blocking probability.

Each link e in the network is associated with two independent costs, the setup cost, s_e, and the hold cost, h_e. Initially, for the case of arbitrary graphs, we present a lower bound of $\Omega(\log n)$ on the competitive ratio of any handoff algorithm, where n is the number of nodes in the graph. Then, for the handoff-anticipating model, we consider several important cases where competitive on-line algorithms exist. We present a 2-competitive on-line algorithm for managing a session in a tree network. The difficulty in this case is how to avoid repeated allocations of the same 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

[2] We should take into account also the movement frequency for comparing these costs.

is required to determine both the VC path at each time step during the session and when to release unused link resources.

In the following section we turn our attention to the non-anticipating model and show that if there is a correlation between the setup and the hold costs of the links, then efficient competitive algorithms do exist. We assume that the ratio between the hold cost and the setup cost is bounded by two constants c_1 and c_2, where $c_1 \leq c_2$. With these restrictions, we present two $(\frac{c_2}{c_1} + 2)$-competitive algorithms. When $c_1 = c_2$, the algorithms are 3-competitive. This case is of practical importance since it reflects environments of current mobile networks.

11.1.3 The Chapter Organization

The chapter is organized as follows. Section 11.2 describes the network model and a short survey of different known handoff rerouting algorithms is given in section 11.3. In Section 11.4 we present both lower bounds and competitive algorithms for the handoff-anticipating model. Competitive algorithms for the non-anticipating model are Given in Section 11.5. We strengthen our analytical results by performing simulations and present the results in Section 11.6. Finally, we conclude this study in Section 11.7. For clarity of exposition, some of the proofs are omitted and others are deferred to the appendix.

11.2 THE NETWORK MODEL

We assume an arbitrary connection-oriented network modeled by an undirected graph $G(V, E)$, where the nodes and edges represent communication switches and full duplex links respectively. Users are attached to the nodes and can be either static or mobile. A mobile user may move and change its attachment node. In our discussion we do not consider any specific network architecture and we assume that a switch can serve a single base station or a group of base stations. The set of base stations serviced by a single node (switch) is called the *node coverage area*[3]. We consider only inter-node handoff operations, i.e., a movement of a mobile user from the coverage area of one node to the coverage area of another.

[3]Recall that large PCS networks have a hierarchical architecture. Each network switch connects a group of base station controllers (BSC) to the infrastructure network and each BSC serves a group of base stations.

A session between two users requires the establishment of a *virtual channel* (VC) between the corresponding nodes and holding it during the session. This means allocating resources at each edge over the VC path and holding them during the session. Each edge $e \in E$ is associated with a *linear cost function*, $f_e(\tau) = s_e + h_e \cdot \tau$, that defines the cost of using edge e for a duration of τ time units. The *setup cost*, $s_e \geq 0$, is the cost of allocating resources over edge e, and the *hold cost*, $h_e \geq 0$, is the cost of holding these resources for a single time unit. The hold time, τ, is measured from the time the edge resources are allocated until they are released. Hence, the entire cost of a VC session of duration τ, which is routed over a path p, is given by $f_p(\tau) = \sum_{e \in p} f_e(\tau)$. We use cost as a general term, and it can capture delay, dollar cost, handoff latency, signaling cost, or an aggregation of several measures.

Now, consider a session σ between two users that starts at time zero and terminates at time τ. The session is defined by a sequence of m triplets, $\sigma = \{(u_i, v_i, t_i)\}_{i=0}^{m}$, where the i-th triplet represents a movement of a user from node u_i to node v_i at a time t_i. We also consider the session initialization and the termination 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 t_m, when one of the users moves to the node to which the other attaches, and they both do not change their attachment node anymore.

The session cost depends on the used handoff rerouting algorithm. This cost is composed of the overall setup cost and the overall hold cost. For a given handoff algorithm $\mathcal{A}$ and a session σ, the first term is denoted by $Setup\_Cost_{\mathcal{A}}(\sigma)$ and the second term by $Hold\_Cost_{\mathcal{A}}(\sigma)$. Thus, the overall cost of a session σ results by algorithm $\mathcal{A}$ is,

$$Cost_{\mathcal{A}}(\sigma) = Setup\_Cost_{\mathcal{A}}(\sigma) + Hold\_Cost_{\mathcal{A}}(\sigma)$$

11.3 EXISTING HANDOFF REROUTING SCHEMES

We present the main handoff rerouting algorithms that are described in the literature and evaluate their performance using competitive analysis. Recall that this is a worst case analysis comparing the session cost of the algorithm under discussion with the session cost of an optimal off-line algorithm.

11.3.1 The Algorithm Description

As mentioned above, the objective of handoff rerouting algorithms is to maintain the ongoing sessions with mobile users while they change their attachment points with the system's infrastructure. They should minimize the consumption of network resources while ensuring minimal disruption to the traffic flow. A handoff algorithm consumes network resources by employing signaling and session rerouting each time a session's participant moves to a new place and a handoff operation occurs. In addition, the required bandwidth is reserved at all the links along the session VC. The current existing handoff rerouting algorithms can be broadly divided into the following four categories:

- Connection reestablishment.
- Path extension.
- Connection modification.
- Multicast Connection Rerouting (Handoff anticipation).

These algorithms differ in their resource consumption and their interference with the session flow. A summary of handoff rerouting algorithms and comparisons is given in [8].

The Connection Reestablishment Algorithm

The *connection reestablishment* is a simple handoff algorithm that establishes a completely new VC between the users at each handoff operation and releases the previous VC [18]. This algorithm optimizes the network utilization at the expense of high signaling cost and high handoff latency, making it inefficient in the case of small cell sizes or when the users are distinct.

The Path Extension Algorithm

The *path extension algorithm* [2, 3, 31], also called the *chaining handoff approach* [26], allocates a new segment between the old and the new attachment points and connects it to the existing VC. It never tears off any established VC segment with the exception of cycles. In practical

implementations, optimizations such as detection and releasing of routing loops are usually performed. This is a simple handoff algorithm with low handoff latency and low signaling cost that also preserves the packets' order. The latter eliminates the need for large buffers and packet reorder mechanisms that is required in other schemes. As a result, the setup cost and the service interruption of this scheme are minimal, at the expense of possibly highly stretched routes that increase the session latency and reduce the network available capacity for other connections. This scheme was especially tailored for Wireless ATM-LAN environment where the base stations are directly connected to the ATM-LAN network. The proponents of these schemes claim that in such environments the additional delay that results from the link chaining is negligible, and the increasing hold cost is insignificant relative to the large bandwidth capacity available in the system infrastructure.

The Connection Modification Category

The *connection modification* (CM) category, also called the *partial rerouting* method, contains a large group of algorithms that they are all employ the following approach. When a handoff occurs, a CM algorithm uses part of the existing session VC and establish a new path between the user's new location and a *crossover switch* (COS). Algorithms using this approach differ by the way the COS is selected.

The Fixed Anchor Rerouting Scheme [5, 7, 24] - In this scheme a single node in the VC is selected as an anchor and only the path to the anchor is modified. The scheme is based on the assumption that during the time-life of a session the mobile user usually remains in the switch coverage area. Thus, both the signaling cost and the session interference are kept low and the VC route is near optimal. A similar scheme is also used by GSM networks [13, 23].

The Last Divergent Rerouting Algorithm [19, 20] - This algorithm selects as the COS the first node on the shortest path between the users that is also included in the existing VC. This approach can be viewed as an improvement of the connection reestablishment scheme and it is targeted to minimize the network resources held by the session. Like connection reestablishment scheme, this algorithm gives the best

performance in terms of resource utilization. However, the handoff signaling as well as the session disruption may be large when the variation between the old and the new VCs is significant.

The Minimal Path Update (MPU) Algorithm [29, 19, 4] - This approach, also called the *nearest common node rerouting* (NCNR), selects as a COS the node in the existing VC that is the closest one to the user new location. This algorithm is an improvement of the path extension approach and it minimizes the rerouting cost and the latency of the handoff operations.

All the previous methods use either a *hard* or a *soft* handoff procedure. In a hard handoff procedure, the scheme finds the COS and then releases the VP path between the COS and the previous attachment point before establishing the new VC with the user current attachment point. In a soft handoff procedure, the new VC is allocated before releasing unnecessary resources. Thus, soft handoff reduces the handoff latency experienced by the user. In the following we present a handoff predicting approach that aims to further reduce the handoff latency.

The Multicast Connection Rerouting

The *multicast connection rerouting* category, also called *handoff anticipation* approach establishes a multi-point VC to several adjacent nodes in anticipation of a possible handoff. This approach reduces the handoff latency at the expense of processing cost and network bandwidth utilization. Several variations to this approach have been proposed in the literature.

The Static Virtual Connection Tree Rerouting Scheme (VCT) [1] - In this algorithm, during the connection initialization, a tree structure is established between a root node and a set of nodes in the vicinity of the user current attachment point (usually this tree is established between a switch and the set of base stations in its coverage area). Each packet designated to the user is duplicated and sent to all the leaves of the tree. Thus, while the user moves in the coverage area of the tree its information is always available. Each time the user leaves the coverage area of the current VCT, the algorithm performs inter-VCT handoff where the tree root and its structure are modified.

The Multicast Rerouting Algorithm (MR) [15] - In this method a a set of connections is established from an anchor node to the current attachment point of the mobile user and all its neighboring nodes. When a packet arrives to the anchor node it is duplicated and sent to all the VC-s. Thus, when the user moves to another attachment point its data is already available. The handoff algorithm updates the set of connections by adding a VC to new neighboring nodes and releasing VC-s that are no longer needed. This approach avoids the need to perform inter-VCT handoffs.

The Dynamic Virtual Connection Tree Rerouting Algorithm (DVCT) [32] - Here, the handoff algorithm combines the multicast methods used by the VTC and MR algorithms. It constructs a dynamic virtual tree that spans the user current attachment point and its neighboring nodes. However, network resources are allocated only along the path with the user current location, which is called the active connection. When the user moves to another node, the VC that ends at this node become active and the required resources are allocated along this path.

Summary of the Handoff Algorithms' Properties

In general, most of the above algorithms attempt to optimize the session cost only from the perspective of a single criterion, either the setup cost or the hold cost, at the expense of other criteria. The path extension is a simple algorithm that aims to reduce the setup cost, while the connection reestablishment scheme guarantees the optimal hold cost. The multicast connection rerouting algorithms use a handoff anticipating approach whose goal is to reduce the handoff latency at the expense of high usage of network resources. Connection modification algorithms attempt to reduce the overall session cost, however they cannot guarantee low cost relative to the optimal cost. The properties of these schemes are summarized in Table 11.1, (See also [6]).

11.3.2 Competitive Analysis of the Existing Algorithms

We turn to calculate lower bounds on the competitive ratios of the handoff rerouting algorithms described above. We classify the algorithms

The Schemes	Advanages	Disadvanages
Connection Reestablishment & Last Divergent	Optimal route. Minimal hold cost. Simple.	High handoff latency. High setup cost.
Path Extension and **Minimal Path Update**	Low handoff latency. Keeps packet order. Simple.	Inefficient connection route. High hold Cost.
Connection Modification - Fixed Anchor	Reduced resource usage (setup & hold). Moderate handoff latency.	Complicated.
Multicast Connection	Low handoff latency. Keeps packet order.	Waste of net resources. Very high hold cost. Complicated.

Table 11.1: Comparison of Handoff Rerouting Algorithms.

into two groups. This first group contains *Hold-cost-minimization* algorithms that minimize the session hold cost and ignore its setup cost. For instance, this group contains the connection reestablishment algorithm [18] and the connection modification algorithm presented in [19], where the selected COS is the first node on the shortest path between the users that is also included in the existing VC. The second group includes *Setup-cost-minimization* algorithms that minimize the session setup cost without considering its hold cost. Such algorithms are the path extension algorithm [2],[3], the minimal path update algorithm [29] and the anchor rerouting algorithm [5]. This separation enables us to show that any algorithm that minimizes the cost of only one component may result with high cost of the second component.

Theorem 1: *The competitive ratio of any hold-cost-minimization algorithm is at least $n/2$, where n is the number of nodes in the network, even if the setup cost and hold cost are correlated.*

Proof: Consider the network $G(V, E)$ that is described in Figure 11.1, where for every edge $e \in E$, $s_e = h_e$. For every $i \in [2 \cdots n]$, let $S_{1,i} = (i-1)$ and for every $i \in [3 \cdots n]$, let $S_{i-1,i} = 1 + \epsilon$. We assume a session σ between two users that are located in node 1. Immediately after the session initialization, one of the users moves from node 1 to node n

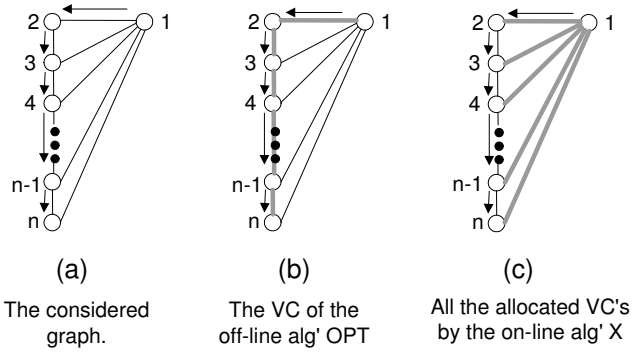

Figure 11.1: The graph $G(V, E)$ used by Theorem 1.

through the path $1, 2, 3, \cdots, n$. As soon as the user reaches node n the session terminates. We assume that the mobile user moves fast enough so that the session duration and hold cost are negligible. Now, consider the best hold-cost-minimization algorithm, denoted by $\mathcal{X}$. After each movement from node $i-1$ to node i the algorithm releases the current VC and establishes a new one between the nodes 1 and i. The cost of this setup operation is $(i-1)$, thus the setup cost of the algorithm is $Setup\_Cost_{\mathcal{X}}(\sigma) = \frac{n\cdot(n-1)}{2}$. However, the optimal off-line algorithm $\mathcal{OPT}$ extend the current VC after each movement. Thus, its setup cost is $Setup\_Cost_{\mathcal{OPT}}(\sigma) = (n-1)\cdot(1+\epsilon)$. As a result,

$$\frac{Cost_{\mathcal{X}}(\sigma)}{Cost_{\mathcal{OPT}}(\sigma)} = \frac{n\cdot(n-1)/2}{(n-1)\cdot(1+\epsilon)} = \frac{n/2}{1+\epsilon}$$

Since ϵ can be any small positive value, the lower bound is at least $n/2$. □

Corollary 1: *The competitive ratio of the connection reestablishment algorithm is at least $n/2$, where n is the number of nodes in the network.*

Proof: The connection reestablishment algorithm will make the same routing decisions that the best setup-cost-minimization algorithm has made in the example given in the proof of Theorem 1. □

Theorem 2: *The competitive ratio of any hold-cost-minimization algorithm is at least n, where n is the number of nodes in the network, even if the setup cost and hold cost are correlated.*

Proof: Consider a ring network $G(V, E)$ as depicted in Figure 11.4, with even number of nodes n and for every edge $e \in E$, $s_e = h_e$. Let $S_{1,n} = 1 + \epsilon$ and let the setup cost of the other edges $e \in E - \{(1, 2)\}$ be $S_e = 1$. We assume a session σ between two users that are located at the nodes 1 and $n/2 + 1$. Immediately after the session initialization, the user from node $n/2 + 1$ moves to node 2 through its left neighbor. We assume that the mobile user moves fast enough so that movement time from node $n/2 + 1$ to node 2 is negligible and the user stays at node 2 for a very long time. Recall that the setup cost of the edges between nodes 1 and $n/2 + 1$ along the left side of the ring (the path $1, 2, \cdots, n/2 + 1$) is $n/2 + \epsilon$ while the cost of the path along the right side is $n/2$. Now, consider the best setup-cost-minimization algorithm, denoted by $\mathcal{X}$. At the beginning, it establish the VC a long the right side of the ring, and after each movement from node $i - 1$ to node i, $i \in [n/2 + 2, n]$, the algorithm adds the edge $(i - 1, i)$ to the existing VC. Thus, $Setup\_Cost_{\mathcal{X}}(\sigma, t) = n - 1$ and the hold cost until time t is $Hold\_Cost_{\mathcal{X}}(\sigma) = (n - 1) \cdot t$. Let us tern to describe the routing decisions of the off-line algorithm $\mathcal{OPT}$. Algorithm $\mathcal{OPT}$ routes the VC along the left side of the ring, and after each movement it removed the unused edge. Thus, $Setup\_Cost_{\mathcal{OPT}}(\sigma) = n/2 + \epsilon$ and the hold cost until time t is $Hold\_Cost_{\mathcal{OPT}}(\sigma, t) = (1 + \epsilon) \cdot t$. Consequentially,

$$\frac{Cost_{\mathcal{X}}(\sigma, t)}{Cost_{\mathcal{OPT}}(\sigma, t)} = \frac{(n - 1) + (n - 1) \cdot t}{n/2 + \epsilon + (1 + \epsilon) \cdot t} \simeq n$$

Thus, for large t and small ϵ the ratio is approximately n. □

Corollary 2: *The competitive ratios of the path extension, the minimal path update, and the anchor rerouting algorithms are at least n, where n is the number of nodes in the network.*

Proof: These algorithms will make the same routing decisions that the best setup-cost-minimization algorithm has made in the example given in the proof of Theorem 2. □

The above theorems prove that the described handoff algorithms may make poor routing decisions that yield high session cost with respect to the cost of an optimal algorithm. In the following we focus on competitive handoff algorithms that balance between the session setup and hold costs for obtaining constant competitive ratios.

11.4 HANDOFF ANTICIPATING ALGORITHMS

In this section we consider handoff anticipating competitive algorithms that are allowed to allocate and hold unused edge resources. As we have shown in Section 11.1.1, these resources may be used for reducing the overall session cost if it is known that these resources will be used later on during the session. It is clear that this ability gives a considerable advantage to the off-line algorithm that knows all the future movements in advance. Therefore, as we prove in Theorem 3, there is a lower bound of $\Omega(\log n)$ for any on-line algorithm for general graphs, where n is the number of nodes in the network. This is not a surprising result, since the connection management problem can be shown to be related to the on-line Steiner tree problem [16] that has a similar lower bound, by viewing each movement of a user as corresponding to a node leaving the tree and another node joining it.

Theorem 3: *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.*

We defer the proof of Theorem 3 to Appendix 11.A In the sequel, we consider two special cases where constant competitive on-line algorithms do exist. In Section11.4.1, we consider a tree topology and Section 11.4.2 intrudes a competitive algorithm for a ring topology. To this end, let us first examine what are the properties of an optimal off-line algorithm. A connection 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 4: *If the edge cost functions are linear, then there is a lazy optimal off-line algorithm.*

Proof: Suppose in contrast that such lazy optimal algorithm does not exist. Consider a non-lazy optimal off-line algorithm and define a lazy algorithm that makes the same rerouting operations but postpones their execution until one of the user moves. Thus, the setup cost of the lazy algorithm is as most as the setup cost of the optimal algorithm. Since the lazy algorithm allocates the used resources after the optimal algorithm,

its hold cost is no more than the hold cost of the optimal Algorithm. Hence, the lazy algorithm yield the same cost or better than the non-lazy off-line optimal algorithm. □

Note that Theorem 4 is satisfied also when considering non-anticipating competitive algorithms.

11.4.1 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:

$$Active\_Cost(p,t) = \sum_{e \in p} \begin{cases} h_e \cdot \tau_e(t) & \tau_e \leq s_e/h_e \\ 2 \cdot s_e & \text{otherwise} \end{cases} \tag{11.1}$$

The hold cost of each edge remains unchanged.

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

Proof: Consider a session σ between mobile users that are controlled by a cruel adversary. The adversary increases the session cost achieved by $\mathcal{T}$ (with minimal effect on the cost of OPT) by forcing the on-line algorithm to repeatedly allocate VC resources immediately following their release. When VC resources become active, the adversary moves the mobile users to new nodes where these resources are not required any more (as described in Example 1). We claim that this is the worst-case scenario from the perspective of Algorithm $\mathcal{T}$. Let us bound the contribution to the cost of the session in both OPT and Algorithm $\mathcal{T}$ of every edge $e \in E$. Suppose that edge e is activated n_e times, and let the total period of time in which edge e is used be θ_e. Algorithm $\mathcal{T}$ incurs a cost of $2 \cdot s_e$ for each activation of edge e, where this cost includes a setup cost and a postponed period cost. Concerning OPT, when edge resources become unused, OPT has two options to choose from. It can either maintain the edge resources until the next activation, i.e., for a period of s_e/h_e, or release them immediately when they become unused, and reallocate them at a later point of time. The activation cost of both options is s_e. Hence,

$$Cost_{\mathcal{A}}(\sigma) = \sum_{e \in E} (n_e \cdot 2 \cdot s_e + \theta_e \cdot h_e) \leq$$

$$\leq 2 \cdot \sum_{e \in E} (n_e \cdot s_e + \theta_e \cdot h_e) = 2 \cdot Cost_{OPT}(\sigma)$$

□

In [9], it is shown that the competitive ratio of any on-line algorithm for a tree topology is at least 2. Thus, Algorithm $\mathcal{T}$ is the optimal on-line algorithm for trees.

11.4.2 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. 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 11.4.1, where a path activation cost is defined by Equation 11.1. For determining the VC path during the session the algorithm uses a retrospective approach [17]. It keeps track of the past and routes the VC in accordance with the routing decisions made by OPT.

Our algorithm uses Theorem 4 for estimating the possible decisions of algorithm OPT. This theorem states that there is a lazy off-line algorithm OPT for optimal session management. Moreover, 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 OPT is

$$Active\_Cost^*(p,t) = \sum_{e \in p} \begin{cases} h_e \cdot \tau_e(t) & \tau_e \leq s_e/h_e \\ s_e & Otherwise \end{cases} \tag{11.2}$$

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 (as described in Example 2). 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 11.2. 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 note 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.

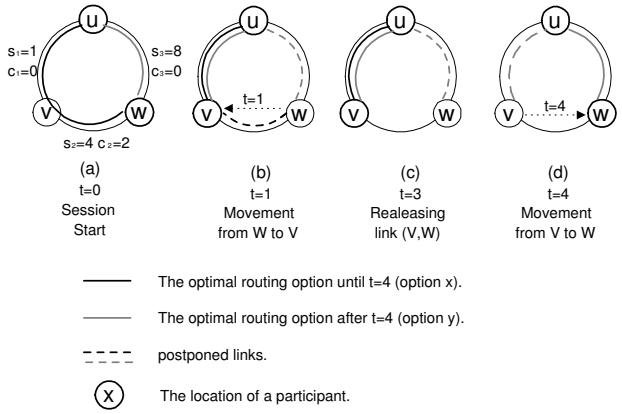

Figure 11.2: An example of a session in a ring.

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 11.2, where the setup cost and hold cost are denoted over the edges. Now, 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 11.3, 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 be both 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.

Theorem 6: *Algorithm $\mathcal{R}$ is 4-competitive.*

The proof of Theorem 6 is given in Appendix 11.B.

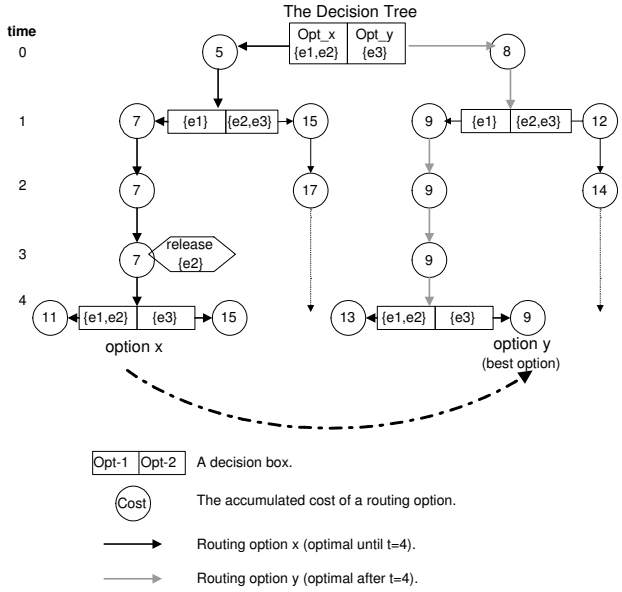

Figure 11.3: The decision tree of the session.

11.5 HANDOFF NON-ANTICIPATING SCHEMES

We turn to the class of *handoff non-anticipating competitive algorithms*. The latter algorithms are considered more practical than the handoff anticipating algorithms from two main reasons. First, the uncertainty of predicting the future movement of users, results in holding unused resources that increases the session hold cost without ensuring any reduction of the setup cost. Second, holding unused resources may hurt the network potential utilization and increase the call blocking probability.

We start by showing lower bounds for this problem. Although, in this case, the offline algorithm is more restricted than the handoff anticipating offline algorithm, the competitive ratio of the best on-line algorithm for general graph is still at least $\Omega(\log n)$, where n is the number of nodes in the graph. Then we restrict our discussion to the case where there is a correlation between the setup cost and the hold cost. For each link $e \in E$, the ratio between the hold cost and the setup cost is bounded by two constants c_1 and c_2, such that $c_1 \leq h_e/s_e \leq c_2$. With these restrictions, we present two $(\frac{c_2}{c_1} + 2)$-competitive algorithms. When $c_1 = c_2$, the

algorithms are 3-competitive. This case is of practical importance since it reflects environments of current mobile networks.

11.5.1 Lower Bounds for Non-Anticipating Algorithms

First, we consider the case of general graphs where there are no restrictions on the setup and hold costs of a link.

Theorem 7: *Consider a general graph and linear cost functions. 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.*

The proof of Theorem 7 can be found in [10]. This bound holds even if all edges have the same setup cost and the mobile users are allowed to move only between adjacent nodes. The proof strongly uses the independence between the setup and hold costs of each edge. Therefore, we restrict our study to a *correlated cost model* where the two costs of each edge are correlated. The graph is associated with two positive constants, c_1 and c_2, that bound the ratio h_e/s_e for every edge $e \in E$, i.e., $c_1 \leq h_e/s_e \leq c_2$. Now, let us also show the lower bound of the correlated cost model.

Theorem 8: *If the setup cost and hold cost are correlated, then the competitive ratio of any on-line algorithm is at least* 2.

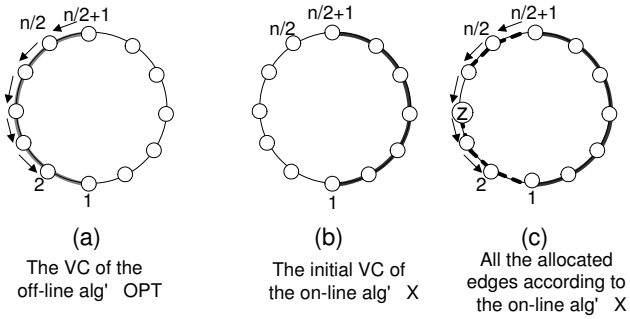

Figure 11.4: The ring for the lower bound proof

Proof: We assume in this proof that $c_2 = c_1$. Suppose, in contrast, that there is an on-line Algorithm $\mathcal{X}$ with a competitive ratio $\gamma < 2$. Consider a ring with n nodes, where n is an even number greater then $\frac{2}{2-\gamma}$, and both the setup and hold costs of each edge are 1. Clearly, $\frac{2\cdot(n-1)}{n} > \gamma$. Let σ be a session between a mobile and a static user which are initially at distance $n/2$ away from each other, e.g., at nodes $n/2+1$ and 1, respectively, as depicted in Figure 11.4. During the session initialization, Algorithm $\mathcal{X}$ establishes a VC connecting the two users, and suppose that the VC is routed through the right side of the ring. Immediately after the VC setup the mobile user moves to node $n/2$, and continues its movement until it reach node 2 through its left neighbor. We assume that the mobile user moves fast enough so that the session duration and hold cost are negligible.

Let us turn to calculate the setup cost of Algorithm $\mathcal{X}$. Initially, it uses the path-extension method until the mobile user reaches some node z, where it decides to reroute the session VC through the left side of the ring. Thus, its setup cost is at least $(n-1)$. Note that node z may be any node in the left side of the ring including nodes 2 and $n/2$. Concerning the off-line algorithm OPT, since it knows the mobile user movements in advance, it routes the session VC through the left side of the ring. After each movement it only releases unused VC resources without allocating new ones. Therefore, its setup cost is $n/2$. The competitive ratio of Algorithm $\mathcal{X}$ is $\frac{Cost_{\mathcal{X}}(\sigma)}{Cost_{OPT}(\sigma)} = \frac{n-1}{n/2} > \gamma$, in contrast to the above assumption. □

11.5.2 Efficient Competitive Algorithms

In the sequel, we present two lazy on-line algorithms for arbitrary graphs and a correlated cost model with a competitive ratio of $1 + 2 \cdot \frac{c_2}{c_1}$. If $c_1 = c_2$, then the competitive ratio is 3. The two algorithms use different techniques for balancing between the setup and hold costs of a session. The first algorithm, denote by $\mathcal{A}$, is the simpler one and we also provide a full analysis of its competitive ratio. The second algorithm, termed Algorithm $\mathcal{B}$, employs more cost-effective methods, and although it achieves the same competitive ratio it yields better results in average as we show in Section 11.6.

Algorithm $\mathcal{A}$

The Algorithm's Description

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

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

Figure 11.5: A formal description of Algorithm $\mathcal{A}$.

The first algorithm, which we denote by $\mathcal{A}$, balances between the path extension and the connection reestablishment algorithms. In our description we use the following notation. For 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 $p^*_{u,v}$ and its setup cost by $s^*(u,v)$. In addition, let $h^*(u,v)$ be the *minimum hold cost* between nodes u and v. Note that the hold cost of the shortest path, $p^*_{u,v}$, may be more than $h^*(u,v)$. However, if the constants c_1 and c_2 are close, then the hold cost of the shortest path between a pair of nodes is close to the minimum hold cost.

Algorithm $\mathcal{A}$ 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, $p^*_{w,u}$, 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, $s^*(u,v)$, then the path $p^*_{w,u}$ is established and it becomes part of the VC route.

Otherwise, the current VC is released and a new VC over the shortest path $p^*_{u,v}$ is established. A formal description of the algorithm is given in Figure 11.5, where $p_{w,v}$ is the VC path before the movement.

The algorithm uses the *credit principle*. It attempts to minimize the setup cost of each handoff 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 $\mathcal{A}$ will not exceed $\frac{c_2}{c_1} \cdot \alpha$ times the hold cost of OPT. In the sequel we show that a proper selection of the parameter α yields a small competitive ratio.

The Algorithm's 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 11.2.

Lemma 1: *For every session* σ,
$Hold\_Cost_{\mathcal{A}}(\sigma) \leq \frac{c_2}{c_1} \cdot \alpha \cdot Hold\_Cost_{OPT}(\sigma)$.

Proof: For every edge $e \in E$, $c_1 \leq h_e/s_e \leq c_2$. Therefore, for every pair u and v, $s^*(u,v) \leq h^*(u,v)/c_1$. 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 $s(p) \leq \alpha \cdot s^*(u,v)$, where the users are attached to nodes u and v. Hence, $h(p) \leq c_2 \cdot s(p) \leq c_2 \cdot \alpha \cdot s^*(u,v) \leq \frac{c_2}{c_1} \cdot \alpha \cdot h^*(u,v)$ proving the lemma. □

Consider the i-th movement from node w_i to node u_i. Let $s(M_i) = s^*(w_i, u_i)$ 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 2: *For every session* σ, $Setup\_Cost_{OPT}(\sigma) \geq \frac{s(M)}{2}$.

Proof: First, assume that 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 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 $(s^*(w_i,x)+s^*(x,u_i))/2$. By the triangle inequality, we get that $s^*(w_i,x)+s^*(x,u_i) \geq s^*(w_i,u_i)$ Therefore, the total cost is

$$Setup\_Cost_{OPT}(\sigma) \geq \sum_{i=0}^{m} \frac{s^*(w_i,u_i)}{2} = \frac{s(M)}{2}$$

□

Lemma 3: *For every session* σ, $Setup\_Cost_{\mathcal{A}}(\sigma) \leq \frac{\alpha}{\alpha-1} \cdot s(M)$.

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,

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

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

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

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

□

Theorem 9: *Algorithm $\mathcal{A}$ 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 1, the competitive ratio of the hold cost is

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

According to Lemma 2 and Lemma 3, the competitive ratio of the setup cost is

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

The value of α that minimizes the competitive ratio of both components is obtained from the equation $\frac{2 \cdot \alpha}{\alpha - 1} = \frac{c_2}{c_1} \cdot \alpha$. Hence, the best competitive ratio is obtained by setting $\alpha = 1 + 2 \cdot \frac{c_1}{c_2}$, and its value is $2 + \frac{c_2}{c_1}$. □

Corollary 3: *If $c_2 = c_1$, then Algorithm $\mathcal{A}$ is 3-competitive (for $\alpha = 3$) for general graphs.*

Algorithm $\mathcal{B}$

The second algorithm, which we denote by $\mathcal{B}$, uses the connection modification approach for improving the first algorithm in terms of the session cost. It is based on the following two improvements in the selection of the crossover switch (COS) at each handoff operation.

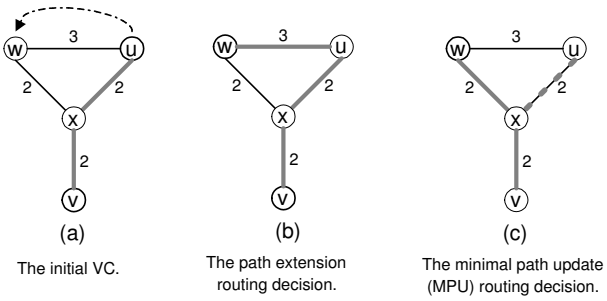

Figure 11.6: Selecting the VC route according to the path extension and the minimal path update algorithms.

The first improvement is achieved by selecting the COS according to *minimal path update* (MPU). The selected COS is the node on the existing VC which is the closest to the user new attachment node. This is the cheapest modification of the existing VC and Figure 11.6 demonstrates that such a selection always yields lower setup and hold costs compared with the path extension algorithm.

The second improvement is achieved by removing exceptionally "heavy" segments from an MPU VC. A segment $p_{a,b}$ between nodes a and b is called an *exceptional segment* if its cost is at least α times the setup cost of the shortest path, i.e., $s(p_{a,b}) \geq \alpha \cdot s^*(a,b)$. If this happens, then the algorithm checks if there are exceptional segments that contain the selected COS, and replaces the most expensive exceptional segment by a shortest path, called a *shortcut*. This improvement considerably reduces the VC cost by establishing low cost shortcuts and releasing unused resources. It is especially useful for users that have local movement patterns, as described by Figure 11.7. In this figure the users are initially attached to the nodes u and v. After the session initialization, the mobile user from node u moves around node u, and creates exceptional segments along its way. Each time such a segment is detected it is replaced in the VC by a shortcut. Thus the VC length during the session is kept close to optimal.

So far we have described the two improvements as two separate steps which are employed sequentially. However, a better COS that further reduces the VC cost is as follows. For each handoff operation we allocate a budget equal to the cost of the allocated paths according to both of the

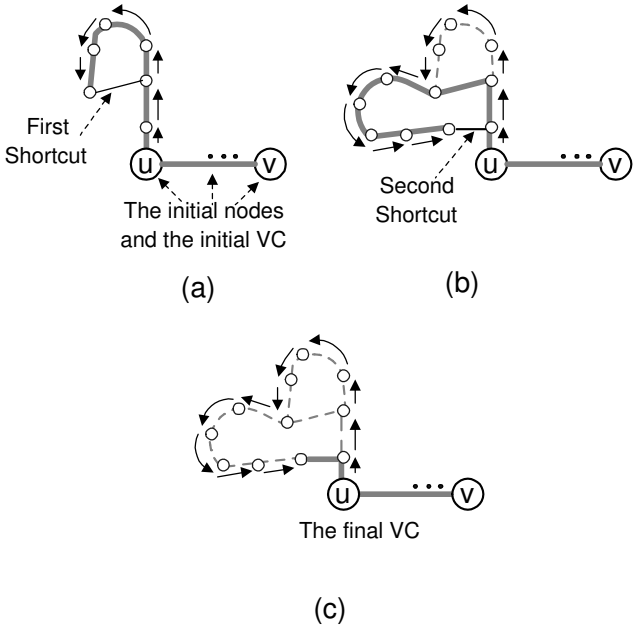

Figure 11.7: An example of session with exceptional segment removal operations.

above improvements. This budget upper bounds the cost of the allocated path. It is used for finding the COS that further reduces the VC cost as much as possible while satisfying the budget constraint. Note that if an exceptional segment is not found, then the selected path is the same as the path selected by an MPU decision. Otherwise, if an exceptional segment exists after the MPU decision, then it is removed at the end of this stage.

The final algorithm works as follows. At the session initialization, it establishes a VC over the shortest path between the session users. Now, suppose that one of the users moves from node w to node u, while the other user is attached to node v, and let $p_{w,v}$ be the VC path before the movement. The algorithm uses three steps for calculating the new path for the VC, $\tilde{p}_{u,v}$. In *Step 1*, it finds the node $x \in p_{w,v}$ which is the closest one to node u, $x = arg\_min_{x \in p_{w,v}}\{s^*(u,x)\}$. This function computes the value of x that minimizes $s^*(u,x)$. Let $\tilde{p}_{u,v}$ be the path that is obtained by concatenating the VC segment between nodes v and

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

Upon a movement from node w to u when the second user is attached to node v do:
$x \leftarrow arg\_min_{x \in p_{w,v}}\{s^*(u,x)\}$
$\tilde{p}_{u,v} \leftarrow p^*_{u,x} \cup p_{x,v}$.
$(a,b) \leftarrow arg\_max_{a \in p^*_{u,x}, b \in p_{x,v}, s(\tilde{p}_{a,b}) \geq \alpha \cdot s^*(a,b)}\{s(\tilde{p}_{a,b})\}$
If $(\tilde{p}_{a,b} \neq \emptyset)$ then
$B \leftarrow s^*(u,a) + s^*(a,b)$
$c \leftarrow arg\_min_{c \in p_{x,v}, s^*(u,c) \leq B}\{s^*(u,c) + s(p_{c,v})\}$
$\tilde{p}_{u,v} \leftarrow p^*_{u,c} \cup p_{c,v}$
Allocate the missing edge resources in $\tilde{p}_{u,v}$.
Release unused edge resources.
Route the VC over the path $\tilde{p}_{u,v}$.

Figure 11.8: A formal description of Algorithm $\mathcal{B}$.

x with the shortest paths between nodes x and u, $\tilde{p}_{u,v} = p^*_{u,x} \cup p_{x,v}$. In *Step 2*, the algorithm finds the most expensive exceptional segment in the path $\tilde{p}_{u,v}$ that includes node x, denoted by $\tilde{p}_{a,b}$. In *Step 3*, the algorithm selects a COS. If such an exceptional segment was found, then let the handoff budget be $B = s^*(u,a) + s^*(a,b)$. the algorithm selects a COS, c, that minimizes the VC total cost, $s^*(u,c) + s(p_{c,v})$, under the budget constrain, $s^*(u,c) \leq B$. The received path is $\tilde{p}_{u,v} = p_{u,c} \cup p^*_{c,v}$. Otherwise, node x remains the COS. Finally, the algorithm allocates the missing edge resources in the received path $\tilde{p}_{u,v}$, routes the session VC over this path, and releases unused resources. A formal description of Algorithm $\mathcal{B}$ is given in Figure 11.8.

Figure 11.9 provides an example of handoff rerouting operation according to Algorithm $\mathcal{B}$ with $\alpha = 3$. Initially, the users are attached to nodes w and v. Then, the user from node w moves to node u (Figure 11.9-a). In Step 1, the algorithm adds the path (x,z,a,u) to the existing VC and let $\tilde{p}$ be the resulting path (Figure 11.9-b). This path contains two exceptional segments, $\tilde{p}_{y,z}$ and $\tilde{p}_{a,b}$, where $s(\tilde{p}_{y,z}) = 120$ and $s(\tilde{p}_{a,b}) = 200$ (Figure 11.9-c). Note that node u by itself is *not* included in any legitimate exceptional segment. The most expensive segment is

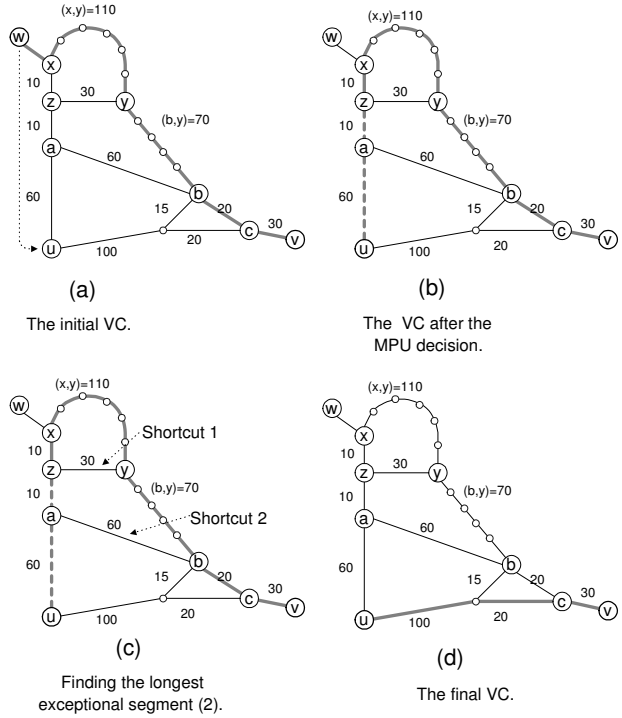

Figure 11.9: An example of handoff operation according to Alg. $\mathcal{B}$.

$\tilde{p}_{a,b}$. Therefore, the handoff operation budget is $B = s^*(u,a) + s^*(a,b) = 60 + 60 = 120$. The node that minimizes the VC cost under the budget constraint is node c and thus the final VC cost is 150 (Figure 11.9-d).

Theorem 10: *Let* $\alpha = 1 + 2 \cdot \frac{c_1}{c_2}$. *Then, Algorithm* $\mathcal{B}$ *is* $(2 + \frac{c_2}{c_1})$-*competitive.*

Corollary 4: *For general graphs with* $c_1 = c_2$, *Algorithm* $\mathcal{B}$ *is* 3-*competitive for* $\alpha = 3$.

The competitive analysis of Algorithm $\mathcal{B}$ can be found in [10]. Moreover, our simulations show that Algorithm $\mathcal{B}$ achieves better results on the average than Algorithm $\mathcal{A}$.

11.6 SIMULATION RESULTS

We compare by simulations the performance of the proposed competitive handoff rerouting algorithms with respect to the other schemes in average sense. The evaluated algorithms are the connection reestablishment algorithm [18], the path extension algorithm [2],[3] and two connection modification algorithms: the minimal path update algorithm [29] and the anchor rerouting algorithm [5]. For the latter, the initial location of the mobile user is selected as an anchor (a permanent COS) and only the path to the anchor is modified. We also evaluated the performance of algorithms $\mathcal{A}$ and $\mathcal{B}$, that are described in Section 11.5. Our simulations considered different networks, different roaming distances and various initial distances between the users. For algorithms $\mathcal{A}$ and $\mathcal{B}$ we also evaluated the affect of different credit parameter values, α, on their performance.

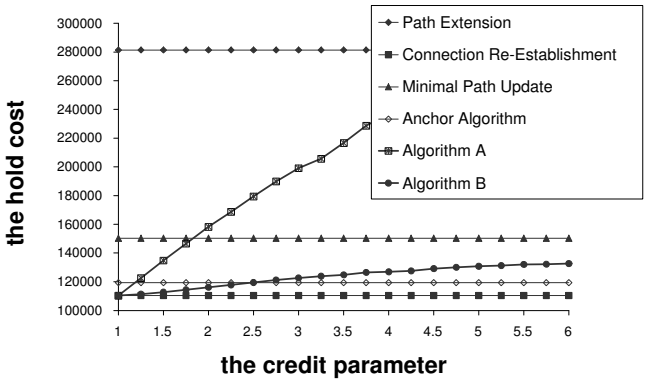

Figure 11.10: The effect of the credit parameter, α, on the hold cost.

Selected typical results from our experiments are described in in Figures 11.10-11.13. In this example, the tested communication network is a grid graph, where both the setup cost s_e, and the hold cost, h_e, of each edge $e \in E$ are 1. We evaluated the average setup and hold costs of sessions with the following characteristics. The initial distance between the users is 200 edges, where one of them is static and the other is mobile. The mobile user moves along a random path with 500 nodes. Its handoff rate is one handoff operation per time unit and the movement range is limited to a square of 100×100 nodes for achieving local movement

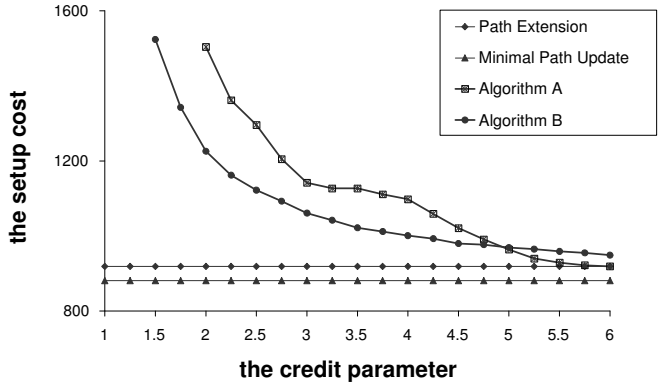

Figure 11.11: The effect of the credit parameter, α, on the setup cost.

effect. The session overall cost is calculated by the following equation,

$$Total\_Cost = \beta \cdot Setup\_Cost + (1 - \beta) \cdot Hold\_Cost$$

where $0 \leq \beta \leq 1$ determines the effect of the setup and the hold costs on the session overall cost.

Our experimental results show that the connection reestablishment algorithm yields the lowest hold cost with the expense of high setup cost, while the minimal path update algorithm achieved the lowest setup cost with high hold cost. The anchor rerouting algorithm and our proposed algorithms balance between the setup and the hold cost of the session, where Algorithm $\mathcal{B}$ yields a relatively low setup cost and hold cost. Figure 11.10 describes the effect of α on the hold cost and Figure 11.11 shows its effect on the setup cost. In the latter figure the results of the connection reestablishment and the anchor rerouting algorithms were omitted due to their high setup cost relative to the other algorithms (110365 and 19331 respectively). Figures 11.12 and 11.13 demonstrate the effect of α over the session overall cost for the cases where β equal 0.5 and 0.99 respectively. In the first case the hold cost is the dominant component of the session total cost. Therefore, $\alpha = 1.25$ yields the lowest total cost for both algorithms $\mathcal{A}$ and $\mathcal{B}$. Moreover, Algorithm $\mathcal{B}$ produces the minimal session cost for every α[4]. In the second case the setup and the hold cost have a similar effect over the session total cost. Here, for every $\alpha \geq 2.25$,

[4]The evaluated α is in the range $[1, 6]$.

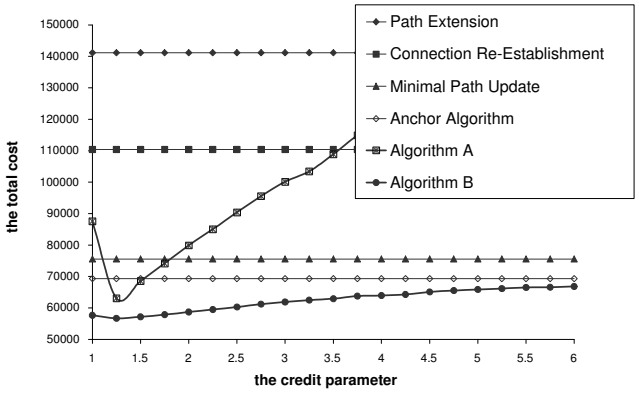

Figure 11.12: The effect of the credit parameter, α, on the session total cost with $\beta = 0.5$.

Algorithm $\mathcal{B}$ produces the minimal cost. We see that the credit parameter, α, can be used for balancing between the setup cost and the hold costs. If the hold cost is high relative to the setup cost, then selecting a low value to α guarantees low overall session cost. If the setup cost is the dominant component, then a high value to α is preferable.

11.7 CONCLUSIONS AND FUTURE WORK

Efficient handoff rerouting algorithms are essential for maintaining continuous connections between mobile users. These algorithms are required to meet two main goals, the first is reducing the network resources used by the connections (both signaling and allocated bandwidth) and the second is minimizing the handoff operation time. For capturing the different goals of the connection management problem, we introduce a network model where each link is associated with setup and hold costs. In this model the different needs can be consolidated into a single objective function and the goal is to reduce the connection overall cost. By reducing this cost, the system efficiently uses the network resources with moderate handoff time. Since, the connection times as well as the movements of the mobile users are not known in advance, we use competitive analysis for evaluating the performance of the various handoff algorithms. Initially, we have shown that most of the proposed handoff rerouting

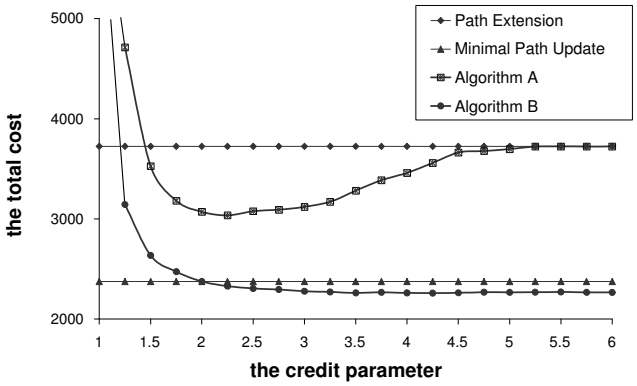

Figure 11.13: The effect of the credit parameter, α, on the session total cost with $\beta = 0.99$.

algorithms perform very poorly in the worst case with respect to the optimal session cost. We have also proved a lower bound of $\Omega(\log n)$ on the competitive ratio of any handoff algorithm for general graphs, where n is the number of nodes in the graph. Then we turned our attention to describe novel handoff algorithms that ensure constant competitive rations in two models. In the case of handoff-anticipating model, we presented a 2-competitive algorithm for a tree topology, and a 4-competitive for a ring topology. For the non-anticipating model we considered the case of correlated costs, where for every edge e of the network graph the ratio of its setup cost, s_e, and hold cost, h_e, bounded by two positive constants c_1 and c_2, i.e., $c_1 \leq h_e/s_e \leq c_2$. In this case, we described two new handoff rerouting algorithms with a competitive ratio of $(2 + \frac{c_2}{c_1})$. These algorithms balance between setup and hold costs by using a credit parameter α. Our simulation results demonstrated that the proposed algorithms yield low overall cost also in the average sense relative to the best algorithms described in the literature. These experiments show that selecting a proper value of α is essential for minimizing the session cost. This value depends on the network parameters as well as on the users movement characteristics. Finding the best α that optimizes the algorithms performance is still an open question.

Bibliography

[1] A. Acampora and M. Naghshineh. An Architecture and Methodology for Mobile Executed Handoff in Cellular ATM networks. IEEE J. on selected area in communication (JSAC). Vol. 12 No. 8. October 1994.

[2] A. Acharya, S. Biswas, L. French and D. Raychaudhuri. Handoff and Location Management in Mobile ATM Networks. Proc. of the 3rd Int. Conf. on Mobile Multimedia Communication (MoMu-C3), September 1996.

[3] P. Agrawal, E. Hyden, P. Krzyzanowski, P. Mishra, M. B. Srivastava and J. A. Trotter. "SWAN: A Mobile Multimedia Wireless Network", IEEE personal communication, April 1996.

[4] M. Ajmone Marson, C. F. Chiasserini, A. Fumagalli, R. Lo Cigno and M. Munafo. "Local and Global Handovers Based on In-Band Signaling in Wireless ATM networks", Wireless Networks, Vol. 7, No. 4, pp-425-436, 2001.

[5] 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 96, 1996.

[6] I. F. Akyildiz, J. Y. McNair, J. S. M. Ho, H. Uzunalioglu, and W. Wang, "Mobility Management in Next Generation Wireless Systems", IEEE Proceedings Journal, Vol. 87, No. 8, pp. 1347-1385, August 1999.

[7] B. Akyol and D. Cox. "Re-routing for Handover in a Wireless ATM Network", IEEE Personal Communication, October 1996.

[8] B. A. J. Banh, G. J. Anido and E. Dutkiewicz. "Handover Rerouting Schemes for Connection Oriented Services in Mobile ATM Networks", Proc. IEEE INFOCOM 98, 1998.

[9] Y. Bejerano, I. Cidon, and J. Naor. "Dynamic Session Management: A Competitive On-Line Algorithmic Approach", Proc. of the Dial-M for Mobility workshop. August 2000.

[10] Y. Bejerano, I. Cidon, and J. Naor. "Efficient Handoff Rerouting Algorithms: A Competitive On-Line Algorithmic Approach", Proc. IEEE INFOCOM 98, 1998. To appears in the IEEE Transaction on Networking.

[11] A. Borodin and R. El-Yaniv. "Online Computation And Competitive Analysis", *Cambridge University Press*, 1998.

[12] J. Davie and Y. Rekhter, "MPLS: Technology and Applications", San Francisco: Morgan Kaufman Publishers, 2000.

[13] J. Eberspacher, H. J. Vogel and C. Bettstetter. GSM Switching, Services and Protocols", 2nd edition. John Wiley and Sons, 1999.

[14] A. Festag, T. Assimakaopoulos, L. Westerhoff and A. Wolisz, "Rerouting for Handover in mobile Networks with Connection-Oriented Backbones: An Experimental Testbad", Proc. of ICATM 2000, June 2000.

[15] R. Ghai and S. Singh. "An Architecture and Communication Protocol for Pico-cellular Networks", IEEE Personal Communication Magazine, Vol. 1 No. 3. 1994.

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

[17] 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.

[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] J. Li, R. Yates and D. Raychaudhuri. "Performance Analysis on Path Rerouting Algorithms for Handoff Control in Mobile ATM Networks", Proc. IEEE INFOCOM 99, 1999.

[20] J. Li, R. Yates and D. Raychaudhuri. "Mobile ATM: A Generic and Flexible Network Infrastructure for 3G Mobile Services", . Journal of Communications and Networks, March 2000.

[21] D. E. McDysan and D. L. Spohn. "ATM Theory and Application". McGraw-Hill, 1994.

[22] J. Mikkonen, C. Corrado, C. Evci and M. Progler. "Emerging Wireless Broadband Networks", IEEE Communication Magazine, February 1998.

[23] M. Mouly and M. B. Pautet. "The GSM System For Mobile Communication", M. Mouly, 49 rue Louise Bruneau, Palaiseau, France 1992.

[24] J. Naylon, D. Gimurray, J. Porter and A. Hopper. "Low-Latency Handover in Wireless ATM LAN", IEEE J. on selected area in communication (JSAC). Vol. 16 No. 6. August 1998.

[25] T. Ojanpera and R. Prasad. "An Overview of Third-Generation Wireless Personal Communication: A European Perspective", IEEE Personal Communication, December 1998.

[26] R. Ramjee, T. F. La Porta, J. Kurose and D. Towsley. "Performance Evaluation of Connection Rerouting Schemes for ATM-based Wireless Networks", , IEEE/ACM Transaction on Networking, Vol. 6. No. 3. June 1998.

[27] D. Raychaudhuri. "Wireless ATM: An Enabling Technology for Multimedia Personal Communication", Wireless Networks, Vol. 2, pp 163-171, 1996.

[28] W. Stallings. "ISDN and Broadband ISDN", 2nd edition, Macmillan Publishing Company, 1992.

[29] C. K. Toh. "Performance Evaluation of Crossover Switch Discovery Algorithms for Wireless ATM LAN". Proc. IEEE INFOCOM 96, 1996.

[30] N. D. Tripathi, J. H. Reed and H. F. VanLandingham "Handoff in Cellular Systems" Performance Evaluation of Crossover Switch Discovery Algorithms for Wireless ATM LAN. IEEE Personal Communication. December 1998.

[31] V. W. S. Wong and V. C. M. Leung, "A path optimization signaling protocol for inter-switch handoff in wireless ATM networks", Computer Networks, Vol. 31, No. 9-10, pp. 975-984, May 1999.

[32] O. Wu and V. C. M. Leung, "Connection Architecture and Protocols to support efficient handoff over ATM/BISDN personal communication networks", Wireless Networks, Vol. 1, No. 2, 1996.

[33] V. W. S.Wong and V. C. M. Leung, "Location Management for Next-Generation Personal Communications Networks", IEEE Networks, Vol. 14, No. 5. pp. 18-24, Sep/Oct 2000.

11.A THE PROOF OF THEOREM 3

Theorem 11: *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.*

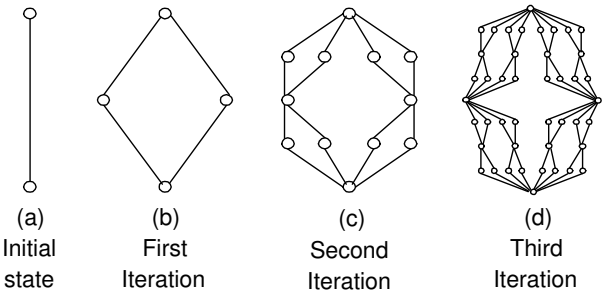

Figure 11.14: The layered graph for proving the lower bound.

Proof: The proof of this theorem is similar to the one presented in [16]. We show a scenario where the cost of every on-line algorithm is at least

$\Omega(\log n)$ times the cost of an optimal off-line algorithm OPT. Let Algorithm $\mathcal{X}$ be an on-line algorithm. We use a layered graph which is defined recursively. We start with a graph that contains two nodes and a single edge connecting them, as depicted in figure 11.14-(a). At each iteration, each edge is replaced by a diamond shape subgraph that contains two new nodes and four edges. Figure 11.14 shows the layered graph obtained after three iterations. Consider a layered graph which is generated after K iterations. This graph contains $n = \frac{2\cdot(4^K-1)}{3}+2$ nodes in 2^K+1 layers, hence, $n \leq 4^K$. The layers are numbered in increasing order from 0 till 2^K, from top to bottom. Each node is associated with two indices: the *layer index* and the *iteration index*. The latter index specifies the iteration in which it was generated. Each edge is associated with a cost function. For simplicity we assume that the cost function of all the edges have a setup component which is equal to one unit. We say that node u is the *k-upper-neighbor* of node v if it is the nearest node to v with a higher layer index and its iteration index is at most k. In a similar way, a node u is the *k-lower-neighbor* of node v if it is the nearest node to v with a lower layer index and its iteration index is at most k.

Now, consider a session σ between two users. One is static and is located at the single node at layer 2^K, while the second is mobile. The movements of the mobile user are determined by a cruel adversary, whose goal is to increase the session cost of Algorithm $\mathcal{X}$ without increasing the cost of OPT. The adversary starts a session as described in Figure 11.15. The session contains $K+1$ phases numbered from 0 to K. At phase 0 the mobile user is located at the single node at layer 0 (see figure 11.15-(a)). As soon as the session VC is established, phase 1 starts and the mobile user moves to a new node. At phase j, the mobile user visits a single node from each layer whose iteration index is j, from top to bottom. A node is selected only if it is not included in an established VC, but its $(j-1)$-th upper and lower neighbors were visited by the mobile user. As soon as a VC between the selected node and the static user is established, the mobile user moves to a new node. This process continues until a node from each layer is visited. Such a session for $K = 3$ is described in Figure 11.15. At the given example, we assume without loss of generality that algorithm $\mathcal{X}$ prefers to route the VC over the right most possible path. Note that the session duration is negligible and therefore the cost of the session is attributed to the VC setup operations. Furthermore, all the visited nodes form a path with 2^K edges that goes through all the graph layers.

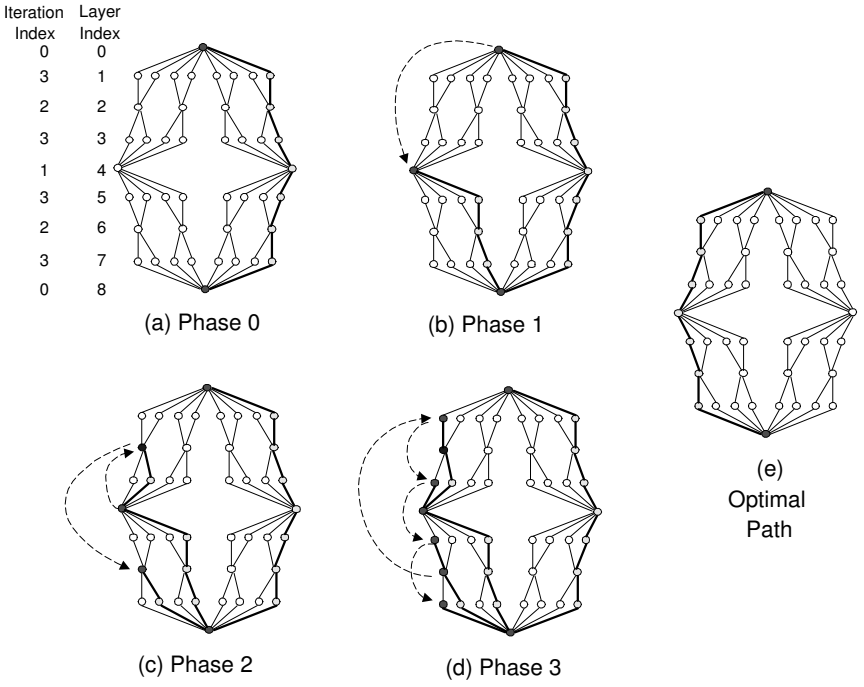

Figure 11.15: The movement pattern for proving the lower bound.

Next, we compare the session cost according to $\mathcal{X}$ and OPT. During the session initialization both algorithms are required to establish a VC that contains 2^K edges. However, while this was the only VC established by OPT, algorithm $\mathcal{X}$ is required to establish more VCs so as to respond to the movements of the mobile user. At each phase j, algorithm $\mathcal{X}$ establishes 2^{j-1} VCs, where each one contains 2^{K-j} edges. Hence, the setup cost of $\mathcal{X}$ at each phase $j \geq 1$ is 2^{K-1}. As a result, the following lower bound is obtained.

$$\frac{Cost_{\mathcal{X}}(\sigma)}{Cost_{OPT}(\sigma)} = \frac{2^K + K \cdot 2^{K-1}}{2^K} =$$

$$= \frac{K+2}{2} \geq \frac{\log_4 n + 2}{2} \geq \frac{\log_4 n}{2} = \Omega(\log n)$$

□

11.B THE COMPETITIVE RATIO OF ALGORITHM $\mathcal{R}$

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 11.2. 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 11.2.

Lemma 4: *Let T_1 and T_2 be two sequences of time intervals such that $T_1 \subseteq T_2$, then for every edge $e \in E$, $Cost^*_{T_1}(e) \leq Cost^*_{T_2}(e)$*

Proof: The correctness of this Lemma is derived from the fact that both the activation cost and the hold cost functions of edge e are continuous and monotonic non-decreasing functions. □

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 11.2.

Lemma 5: *For every session σ and a given time $t_{end} \geq 0$, $Cost^*_{\mathcal{R}}(\sigma, t_{end}) \leq 2 \cdot \tilde{c}(t_{end})$.*

Proof: First, let us divide the session duration into phases, and suppose without loss of generality that the session contains K phases from its initialization time until time t_{end}. Each phase $k \in \{1 \ldots K\}$ starts at time t_{k-1} and ends at time t_k, when $t_0 = 0$ and $t_K = t_{end}$. In phase k the session VC is the same as the VC path of a single routing option, denoted by z_k, which is optimal at time t_k. Hence, the VC path of algorithm $\mathcal{R}$ at time t, $p(\mathcal{R}, t) = p(z_k, t)$ for each time $t \in (t_{k-1}, t_k]$. Such a phase partition is obtained by defining the phases in reverse order. The routing option z_K is the one that is optimal at time $t_K = t_{end}$, and phase K starts just after the latest time t' such that $p(\mathcal{R}, t') \neq p(z_K, t')$. Time t' defines also the time t_{K-1} when phase $K-1$ terminates. Let z_{K-1} be the routing option that is optimal at that time, and continue

the phase partition process as described above until all the phase are defined. From the phase definition we get that $p(z_k, t_k) \neq p(z_{k+1}, t_k)$. Hence, $p(z_k, t_k) \cup p(z_{k+1}, t_k) = E$.

Let $\Delta c(z_k, t, t')$ be the cost of the routing option z_k during the time interval (t, t'), given that at time t the session resources are allocated over all the ring edges. Note that $\Delta c(z_k, t, t') \leq c(z_k, t') - c(z_k, t)$.

The cost of each phase k is the sum of two components. The first is the accumulated cost of the routing option z_k during the phase period. This cost is $c(z_1, t_1)$ for the first phase, and $\Delta c(z_k, t_{k-1}, t_k)$ for each phase $k > 1$. The second component is the transition cost from routing option z_k to z_{k+1} at time t_k. This transition is performed by allocating all resources that routing option z_{k+1} has at that time. According to Lemma 4 the transition cost is bounded by the cost of the routing option z_{k+1} during the period of phase k, which is $c(z_2, t_1)$ for the first phase, and $\Delta c(z_{k+1}, t_{k-1}, t_k)$ for $k > 1$. Therefore,

$$Cost^*_{\mathcal{R}}(\sigma, t_{end}) \leq c(z_1, t_1) + c(z_2, t_1) + \sum_{k=2}^{K-1} \{\Delta c(z_k, t_{k-1}, t_k) +$$

$$+\Delta c(z_{k+1}, t_{k-1}, t_k)\} + \Delta c(z_K, t_{K-1}, t_K)$$

Now, let us separate the contribution of the odd and even routing options.

$$\begin{aligned} Cost^*_{\mathcal{R}}(\sigma, t_{end}) \quad \leq \quad & c(z_1, t_1) + \sum_{k=3,odd}^{K} \{\Delta c(z_k, t_{k-2}, t_{k-1}) + \\ & +\Delta c(z_k, t_{k-1}, t_k)\} + c(z_2, t_1) + \Delta c(z_2, t_1, t_2) + \\ & + \sum_{k=4,even}^{K} \{\Delta c(z_k, t_{k-2}, t_{k-1}) + \Delta c(z_k, t_{k-1}, t_k)\} \\ \leq \quad & c(z_1, t_1) + \{ \sum_{k=3,odd}^{K} \Delta c(z_k, t_{k-2}, t_k)\} + \\ & +c(z_2, t_1) + \Delta c(z_2, t_1, t_2) + \sum_{k=4,even}^{K} \Delta c(z_k, t_{k-2}, t_k) \end{aligned}$$

Since $\Delta c(z_k, t_{k-2}, t_k) \leq c(z_k, t_k) - c(z_k, t_{k-2})$ follows that

$$Cost^*_{\mathcal{R}}(\sigma, t_{end}) \quad \leq \quad c(z_1, t_1) + \sum_{k=3,odd}^{K} \{c(z_k, t_k) - c(z_k, t_{k-2})\} +$$

$$+c(z_2,t_2)+\sum_{k=4,even}^{K}\{c(z_k,t_k)-c(z_k,t_{k-2})\}$$

By using $c(z_k,t_k)=\tilde{c}(t_k)$ and $c(z_k,t_{k-2})\geq\tilde{c}(t_k)$ we receive,

$$\begin{aligned} Cost^*_{\mathcal{R}}(\sigma,t_{end}) &\leq \tilde{c}(t_1)+\sum_{k=3,odd}^{K}\{\tilde{c}(t_k)-\tilde{c}(t_{k-2})\}+ \\ &\quad +\tilde{c}(t_2)+\sum_{k=4,even}^{K}\{\tilde{c}(t_k)-\tilde{c}(t_{k-2})\} \end{aligned}$$

These are two telescopic sums, therefore, $Cost^*_{\mathcal{R}}(\sigma,t_{end})\leq 2\cdot\tilde{c}(t_{end})$. □

Lemma 6: *For every session σ and a given time $t_{end}\geq 0$, $Cost_{\mathcal{R}}(\sigma,t_{end})\leq 4\cdot\tilde{c}(t_{end})$.*

Proof: According to Lemma 5, $Cost^*_{\mathcal{R}}(\sigma,t_{end})\leq 2\cdot\tilde{c}(t_{end})$, when we ignore the cost of redundant postponed periods. This cost contains both edge setup cost and edge hold cost. To include also the cost of redundant postponed periods in the total cost, we should count twice the cost of each edge setup operation, as defined by Equation 11.1. Hence, $Cost_{\mathcal{R}}(\sigma,t_{end})\leq 2\cdot Cost^*_{\mathcal{R}}(\sigma,t_{end})$ As a result, $Cost_{\mathcal{R}}(\sigma,t_{end})\leq 4\cdot\tilde{c}(t_{end})$. □

Theorem 12: *For every session σ, $Cost_{\mathcal{R}}(\sigma)\leq 4\cdot Cost_{OPT}(\sigma)$.*

Proof: On one hand, from Lemma 6, we get that $Cost_{\mathcal{R}}(\sigma,t_{end})\leq 4\cdot\tilde{c}(t_{end})$. On the other hand, the routing option of OPT is also included in the decision tree of the session. Therefore, for any given time $t\geq 0$, $Cost_{OPT}(\sigma,t)\geq\tilde{c}(t)$. As a result, $Cost_{\mathcal{R}}(\sigma)\leq 4\cdot Cost_{OPT}(\sigma)$. □