Optimal Itinerary Planning for Mobile Agents-Based Management

Dr. Damianos Gavalas
Department of Cultural Technology and Communication
University of the Aegean, Mytilini, Lesvos Island, Greece
E-mail: dgavalas@aegean.gr

***Abstract* - Proposed management architectures, incorporating mobile agent (MA) technology, fail to address scalability problems, when monitoring tasks employing multi-hop agents are considered. This is because they lack mechanisms which can guarantee that the agents itinerary is optimised so as to minimise migration traffic and round-trip latency. Herein, we propose an algorithm that adapts methods usually applied for addressing network design problems in the problematic of MA itinerary planning. The algorithm not only suggests the optimal number of MAs that minimise the overall cost but also constructs optimal itineraries for each of them. The algorithm implementation has been integrated into our MA framework research prototype while its validity and competence has been verified through simulation results over large enterprise networks.**

I. Introduction

Despite its wide deployment base, the IETF Simple Network Management Protocol SNMP is known to exhibit serious scalability problems, result of client/server paradigm it follows. These problems have motivated a trend towards distributed management intelligence that represents a rational approach to overcome the limitations of centralised NSM. A new trend in NSM involves using mobile agents (MAs) [2] to manage distributed network systems [4][8]. An MA can be used to locally retrieve and filter management data, monitor systems health and networking conditions in distributed environments. In particular, management tasks are assigned to an agent, which delegates and executes management logic in a distributed and autonomous fashion. After completing these tasks, the results are either communicated through a messaging mechanism or carried back to the manager by the MA.

Delegation of management logic may be realized with agents bound to *single-hop* mobility; the agents move from the managing node to remote managed nodes, where they statically execute their tasks. What is not commonly exploited in management is the MA *multiple-hop* capability, where agents may move several times as they adapt to changing circumstances. While single-hop mobility can improve flexibility and scalability in the context of relatively static networked systems, it is the multiple-hop capability offered by MAs that needs to be exploited to meet the requirements of future networked systems, i.e. large scale and dynamics [4]. In addition, network monitoring based on multi-hop (itinerant) MAs is advantageous for short-term monitoring tasks and also in cases that a global (domain)-level rather than a local (device)-level view of managed resources is required (for a complete discussion of these issues, the interested reader may refer to [2] and [4]).

However, while in single-hop mobility, agent itinerary control is straightforward (the itinerary is restricted to the single destination host), this is not the case in multi-hop mobility, where slight variations on the set of visited hosts or even on the order that a specific set of nodes is visited may result in dramatic changes of the overall trip latency and migration traffic. In this article, we focus on multi-hop mobility, aiming at devising methods to optimize MA itineraries.

In network monitoring applications, using a single MA object launched from the manager platform that sequentially visits all the managed elements, regardless of the underlying topology may actually lead to performance worse than the conventional SNMP-based approach. The performance further declines, when the monitoring MA collects monitoring data from multiple subnets, often interconnected by low-bandwidth links. In such cases, the traffic associated with management tasks typically traverses several network segments and, when summed up, results in increased bandwidth waste [8].

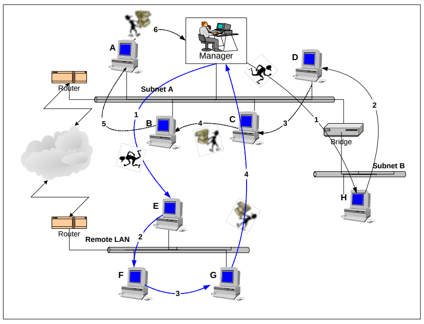


Figure 1. Optimized partitioning of the network into two management domains

This inefficient approach, known as 'flat' MA-based monitoring [2], presents serious scalability problems: in large networks the *round-trip delay* of the MA greatly increases as the overall travel time depends on the number of hops realized by the MA and also the *network overhead* imposed by the MA transfers grows exponentially with the network size [2][8].

A rational approach to overcome such scalability problems is to partition the managed network into several logical/physical domains. For instance, in Figure 1, an MA object polls the devices of the remote LAN, whereas a second MA is assigned to the subnet local to the manager host as well as to another subnet, which is part of the same LAN. Through

The research work presented herein has been co-funded by 75% from EU and 25% from the Greek government under the framework of the Education and Initial Vocational Training II, Programme Archimedes.

management traffic localization, unnecessary usage of expensive networking resources is restricted, thereby improving management scalability. However, the scenario illustrated in Figure 1 represents an ideal case in terms of the WAN link utilisation. That is because the link is traversed only twice per polling interval. A slightly different partitioning scheme or alteration on MAs itinerary would dramatically increase total migration cost. Apparently, even when specific partitioning criteria are followed, the design of MA itineraries lacks a mechanism that would guarantee minimal use of network resources; hence an algorithm for itinerary planning is required. In section III, we describe a Heuristic algorithm for Itinerary Planning (HIP).

The remainder of the paper is organised as follows: Section II reviews works related to the research presented herein. Section III discusses the design and functionality of an algorithm for optimal itinerary planning. Simulation results are presented in Section IV, while Section 0 concludes the paper.

II. Related Work

The problem of optimizing the itineraries of multi-hop MAs has not been sufficiently addressed in the literature. An attempt to address this issue has been reported in [7], which describes a performance model that allows agents to decide whether they should migrate to a site and communicate locally or the communication should be performed remotely. The decision is taken according to an ‘optimal design graph’; in most cases, it has been indicated that the optimal performance of an agent is achieved by a critical sequence of mixed remote procedure calls and agent migrations.

Rubinstein et al. [8] evaluated the scalability of MA-based management on large enterprise networks and compared the performance of this approach against that of centralized management paradigm. Recognizing the fact that “MA size increases with the number of visited nodes and, as a consequence, migration becomes difficult”, they proposed a strategy in which the MA returns to the management station to deliver its collected data, thereby reducing its size before visiting the remaining hosts. However, the possibility of using multiple MAs to perform management tasks is not investigated, nor is the issue of designing efficient agent itineraries.

A work relevant to the research presented herein has been presented in [6], where Qi and Wang propose the employment of MA paradigm in wireless sensor networks. To optimize agents itinerary, they derived a Local Closest First (LCF) algorithm according to which each MA searches for the next destination with the shortest distance to its current location. However, their cost function formulation does not take into account potential partitioning of visited hosts in multiple clusters, which would affect the calculation of the distance matrix. The output of LCF-like algorithms though highly depends on the MAs original location, while the nodes left to be visited last are associated with high migration cost [3]; the reason for this is that they search for the next destination among the nodes adjacent to the MA’s current location, instead of looking at the ‘global’ network distance matrix.

Most importantly though, both the works presented in [6] and [7], deal with the problem of constructing near-optimal MA itineraries for given sets of network nodes, where a *single* MA visits the whole set. Also, they do not address the fundamental problem of partitioning network nodes in optimal clusters. On the other hand, the algorithm presented in the following section deals with the optimal clustering problem and subsequently uses the algorithm’s output to construct near-optimal agent itineraries.

III. A Heuristic Algorithm for Itinerary Planning (HIP)

Interestingly, the problem of designing optimised itineraries exhibits many similarities with the Multi-point Line Topologies or Constrained Minimum Spanning Trees (CMST) problems. A CMST is a Minimum Spanning Tree (i.e. a connected graph without cycles, with the least total cost) with the additional constraint on the size of the subtrees rooted on the ‘center’ (there is an upper limit on the number of nodes included on each of the subtrees originated at the tree’s root). CMST algorithms are used in graph theory, with the main application field being network design problems [3]. In such problems, the objective is the optimal selection of the links connecting terminals to concentrators or directly to the network center, resulting in the minimum possible total cost. The output of CMST algorithms typically comprises topologies partitioned on several multi-point lines (or tree branches), where groups of terminals share a subtree to a specific node (center).

Substituting the terms ‘network center’, ‘link’ and ‘multi-point-line’ with the terms ‘manager station’, ‘migration’ and ‘itinerary’ respectively, and following the observation that the output of CMST algorithms (group of multi-point lines rooted at the center) very much resembles a group of itineraries all originated at the manager station, the similarity of CMST and MA itinerary planning problems becomes evident. As a result, the idea of using algorithms originally devised for CMST problems in the application area of MA itinerary planning, naturally shapes up. CMST problems are NP-hard and as a result several heuristics have been proposed to efficiently deal with them. Our HIP (Heuristic for Itinerary Planning) algorithm adapts some basic principles of *Esau-Williams* (E-W) algorithm [3] in the specific requirements of itinerary planning problems.

The cost function used be E-W algorithm considers selected links cost as the only contributing factor to the total itinerary cost. This is certainly not adequate metric to evaluate the cost of agents itineraries c_{total}. A key factor also affecting c_{total} is the agent size; more importantly, the agent size increment rate [8], which depends on the amount of data collected by the MA on every host. Let us assume that a set of itineraries $I = \{I_1, I_2, \ldots, I_n\}$ is constructed, each assigned to an individual MA object. Each itinerary I_i includes a set of hosts to be sequentially visited by its respective MA: $I_i = \{N_0, N_1, .. , N_n, N_0\}$. Note that all itineraries originate and terminate at the manager station host N_0. The total cost per polling interval over all itineraries $|I|$ becomes:

$$C_{total} = \sum_{i=1}^{|I|} \sum_{j=0}^{|I_i|-1} (d_j + s) * c_j \qquad (1)$$

where d_j is the amount of data collected by the MA on the first j visited hosts, s the MA initial size and c_j the cost of utilizing the link traversed by the MA on its j^{th} hop, i.e. the link connecting hosts N_j and N_{j+1} (c_j is given by the network cost matrix). In principle, HIP algorithm aims at constructing a set of itineraries I minimizing the cost function of equation (1).

In order to provide our HIP algorithm a fair performance metric we have also implemented the LCF algorithm, as described in [6]. For instance, for the network graph of Figure 2a (which also presents link selection costs), the itinerary constructed by LCF is shown in Figure 2b. The sequence numbers enclosed within circles indicate the order in which individual links (or migrations) become accepted in the corresponding algorithm steps.

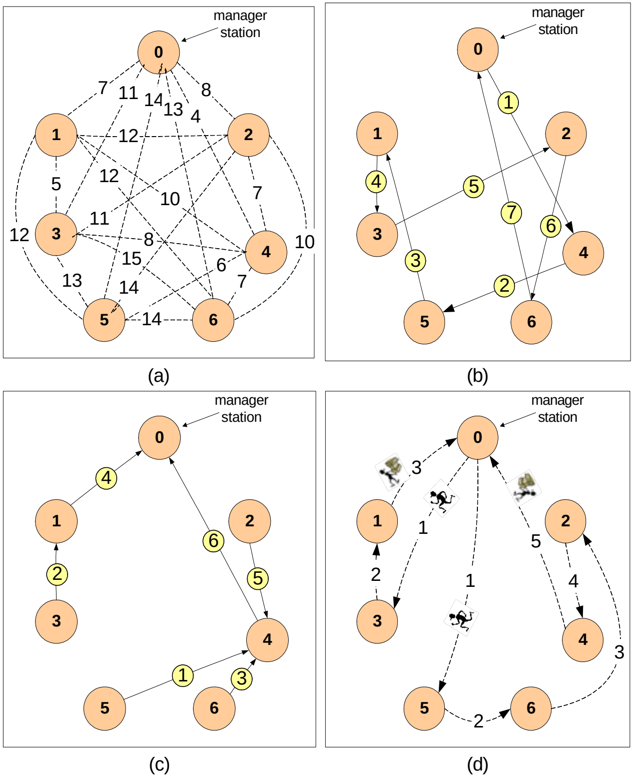


Figure 2. The MA itinerary planning problem: (a) The original network graph; (b) The output of LCF algorithm; (c) The output of HIP algorithm; (d) Two MA itineraries derived from the output of HIP algorithm

Although in Figure 2 example, link selection costs are random, when testing the efficiency of HIP and LCF algorithms in real or simulated environments cost matrices are constructed so as to reflect the cost of using the underlying networking infrastructure, e.g. we accept the fact that it is 'cheaper' for an MA to migrate within a high-speed LAN than over a low-bandwidth WAN link or a wireless connection.

HIP algorithm takes into account the amount of data accumulated by MAs at each visited host (without loss of generality, we assume this is a constant d), a parameter ignored by LCF algorithm. Namely, it recognizes that traveling MAs become 'heavier' while visiting managed devices without returning back to the manager site to 'unload' their collected data [8]. Therefore, HIP restricts the number of migrations performed by individual MAs, thereby promoting the parallel employment of multiple cooperating MAs, each visiting a subset of managed devices.

Specifically, the aim of HIP algorithm is, given a set of hosts $N = \{N_0, N_1, \ldots, N_{n-1}\}$, the manager station N_0 and the cost matrix C, to return a set of near-optimal itineraries $I = \{I_0, .., I_k\}$, all originated and terminated at the manager station. Initially, we assume $|N|$ (= n) itineraries, as many as the network nodes: $I_0, .., I_{n-1}$, each containing a single host ($N_0, N_1, \ldots, N_{n-1}$, respectively). On each algorithm step, two hosts i and j are 'connected' and, as a result, the itineraries including these hosts ($I(i)$ and $I(j)$ respectively) are merged into a single itinerary.

As mentioned in Section 0, LCF-like algorithms usually fail as they tend to leave hosts located far from the center stranded since they prioritize the inclusion of hosts closed to last selected host. As a result, relatively expensive links are left last to be included in the solution, significantly increasing the overall cost. A way of dealing with this problem is to pay more attention to nodes far from the center, giving preference to links incident upon them. HIP algorithm accomplishes this by using the concept of 'tradeoff function' $t_{i,j}$ associated with each link (i, j), defined by:

$$t_{i,j} = c_{i,j} + (d*(|I(i)|+|I(j)|)) - C_{i,N_0} \qquad (2)$$

where C_{i,N_0} is the cost of connecting $I(i)$ to the manager station N_0. Initially, this is simply the cost of connecting node i directly to the manager station. As i becomes part of an itinerary containing other nodes, however, this changes to:

$$C_{i,N_0} = \min_{k\in I(i)} c_{k,C} \qquad (3)$$

Equation (2) implies that the more hosts an itinerary already includes, the more difficult for a new host to become part of that itinerary, especially when d is large. Figure 3 lists a pseudo-code implementation of HIP algorithm.

```
// n: Total number of hosts, c: cost matrix, d: data collected per
host, N_0: origin station */
HIP (n, c, d, N_0)
  // I: the list of itineraries to be constructed
  initialize I
  current = N_0
  /* N_connected: the number of hosts already included into
  an itinerary */
  N_ connected = 0
  while (N_ connected < n)
    /* I(i): the itinerary where host i has already been
      included, $|I(i)|$ :number of hosts included within I(i) */
    compute $t_{i,j} = c_{i,j} + (d*(|I(i)|+|I(j)|)) - C_{i,N_0}$,
    where $I(i) \cap I(j) = \varnothing$ and $C_{i,N_0} = \min_{k\in I(i)} c_{k,N_0}$
    merge (I(i), I(j)), for (i, j) minimizing the tradeoff
    function ( $\min_{i,j} t_{i,j}$ )
    N_ connected ++
  return I
```

Figure 3. Pseudocode implementation of HIP algorithm

HIP algorithm execution steps for the test network graph of Figure 2a are demonstrated in Figure 4, where the links (agent migrations) selected are highlighted; we assume that the amount of data collected per host is d = 1. On every algorithm step, a pair (i, j) minimizing $t_{i,j}$ is selected and, following that, the itineraries containing hosts i and j are merged into a single itinerary. This process is repeated until a set if itineraries including *all* hosts is constructed. Figure 4 presents the values of $t_{i,j}$ for pairs (i, j) minimizing the tradeoff function for each host i ($\min_j t_{i,j}$); the pair (i, j) that minimizes $t_{i,j}$ over all hosts ($\min_{i,j} t_{i,j}$) is then selected.

Step 1	**Step 2**
$t_{13} = 5+(1+1)-7 = 0$ $t_{24} = 7+(1+1)-8 = 1$ $t_{31} = 5+(1+1)-11 = -4$ $t_{40} = 4+(1+1)-4 = 2$ $t_{54} = 6+(1+1)-14 = -6$ $t_{64} = 7+(1+1)-13= -4$	$t_{13} = 0$ $t_{24} = 7+(1+2)-8 = 2$ $t_{31 =} -4$ $t_{40} = 4+(2+1)-4 = 3$ $t_{51} = 12+(2+1)-4 = 11$ $t_{64} = 7+(1+2)-13= -3$
Step 3	**Step 4**
$t_{10} = 7+(2+1)-7 = 3$ $t_{24} = 2$ $t_{34} = 8+(2+2)-7 = 5$ $t_{40} = 3$ $t_{51} = 12+(2+2)-4 = 12$ $t_{64} = -3$	$t_{10} = 3$ $t_{24} = 7+(1+3)-8 = 3$ $t_{34} = 8+(2+3)-7 = 6$ $t_{40} = 4+(3+1)-4 = 4$ $t_{51} = 12+(3+2)-4 = 13$ $t_{62} = 10+(3+1)-4= 10$
Step 5	**Step 6**
$t_{14} = 10+(3+3)-0 = 16$ $t_{24} = 3$ $t_{34} = 8+(3+3)-0 = 14$ $t_{40} = 4+(3+3)-4 = 6$ $t_{51} = 12+(3+3)-4 = 14$ $t_{62} = 10$	$t_{14} = 10+(3+4)-0 = 17$ $t_{20} = 8+(4+3)-4 = 11$ $t_{34} = 8+(3+4)-0 = 15$ $t_{40} = 4+(4+3)-4 = 7$ $t_{51} = 12+(4+3)-4 = 15$ $t_{61} = 12+(4+3)-4 = 15$

Figure 4. HIP algorithm execution steps for the network of Figure 2a

For instance, on step one, the pair minimizing $t_{i,j}$ is (i, j) = (5, 4), hence itineraries including hosts 5 and 4 are merged forming: $I(5) \cup I(4) = \{5,4\}$. On next step, $t_{i,j}$ values are re-calculated, for instance, $t_{2,4} = c_{2,4} + (1*(|I(2)| + |I(4)|) - c_{2,N_0}$ = 7+(1+2)-8 = 2 (note that the set elements of the itinerary including host 4 have increased: I(4) = {5, 4} $\Rightarrow$ $|I(4)| = 2$). At the end of step 6, two sets are constructed, forming two subtrees rooted at the manager host: {6, 5, 4, 2} and {3, 1} (see Figure 2c). It is then a trivial task to form the itinerary plan of the two MAs: I_1 = {0, 5, 6, 2, 4, 0} and I_2 = {0, 3, 1, 0} (see Figure 2d).

IV. Experimental Evaluation

To prove HIP algorithm validity and effectiveness, we have implemented and incorporated the algorithm as an add-in module, termed the *Itinerary Scheduler Module* (ISM), into our research prototype presented in [2]. ISM has been implemented in Java programming language; ISM executes the HIP algorithm and informs the manager application on the number of MAs that need to be instantiated and their respective itineraries. Agent itineraries are reconstructed whenever a new managed device is 'discovered'. LCF algorithm has also been simulated for comparison purposes.

HIP has only been tested on realistic network monitoring application scenarios; yet, it suits any application field which benefits from distribution of intelligence and processing overhead offered by MA paradigm. The experimental testbed includes several Windows NT and Linux Red Hat v. 6.1 stations. The managed network comprises two 10Mbps LANs connected through a 1Mbps leased line (see Figure 5); the first LAN (where the manager application executes) hosts 5 managed elements while the second 10 elements. The network traffic generated by MA migrations has been measured using the WinDump network analyser [9]. Network monitoring data are collected through interacting with SNMP agents hosted by managed elements.

The cost matrix used by HIP algorithm reflects the cost of using network resources (the bandwidth of LANs medium is ten times larger than the bandwidth of the leased line):

$$c_{i,j} = \begin{cases} 1, \text{when i and j are located on the same LAN} \\ 10, \text{when i and j are located on different LANs} \end{cases} \qquad (4)$$

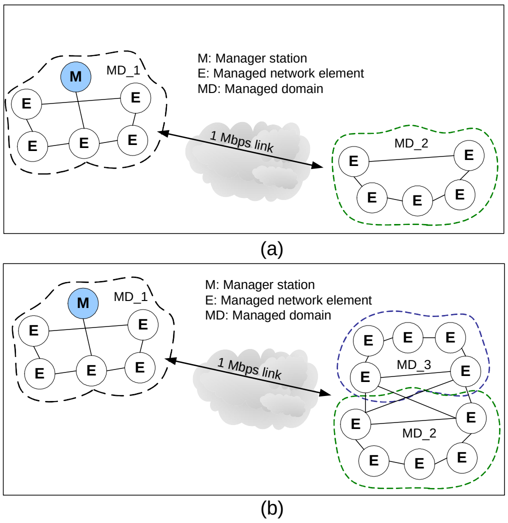


Figure 5. HIP execution on the experimental testbed. (a) The remote subnet hosts 5 managed devices all included in a single managed domain; (b) The remote subnet hosts 10 managed devices separated in two managed domains

The performance of HIP and LCF algorithms has been compared against an SNMP implementation [1]. The application scenario involves polling managed elements for the contents of tcpConnTable MIB-II table [5], which lists information about all TCP connections of a host (a management information base, MIB, is a formal description of the network objects that can be managed using SNMP; MIB-II is supported by virtually all SNMP-compliant network devices in the IP world). In the SNMP-based implementation, individual MIB tables are remotely retrieved through exchanging successive request/response messages, each retrieving a table row. Every tcpConnTable row contains *5* values (columns) [5], while managed elements stored information about *46* TCP connections on average, i.e. included *46* table rows (46 × 5 values in total).

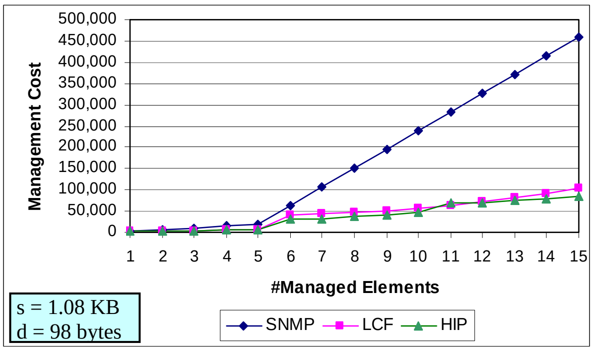
In MA-based approaches, travelling MAs obtain table values through local interaction with legacy systems; table contents are then compressed and encapsulated within MAs state. In contrast with LCF algorithm, HIP involves the parallel execution of multiple MA objects which depends on network size. For instance, when *5* managed elements are hosted in the remote subnet, two managed domains are created, each assigned to an individual MA (Figure 5a). However, when managed elements increase to *10*, HIP separates them into two distinct managed domains and engages an additional MA into polling operation (Figure 5b). The network overhead experiment parameters associated with our experiments have been measured by WinDump tool [9] and presented in Table I.

TABLE I. NETWORK OVERHEAD PARAMETERS

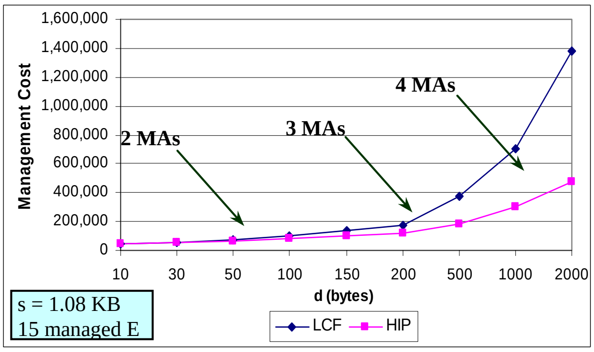
Parameter	Value
MA initial size	1,08 KB
Average SNMP request/response packet size (including UDP header)	90 bytes
Average SNMP packet increment for each additional requested value	17 bytes
MA state size increment per table sample (compressed data)	98 bytes

Experimental results are presented in Figure 6. The overall management cost has been calculated according to the cost function of equation (1), where cost matrix coefficients are given in equation (4).

Figure 6a compares the management cost of SNMP against MA-based implementations, as a function of the number of polled devices, for the network monitoring application scenario described above. SNMP-based polling does not scale well as it involves heavy usage of the relatively expensive interconnection link, when increasing the number of polled devices located in the remote subnet (managed elements *6-15*). In contrast LCF and HIP algorithms require usage of the interconnecting link only when an MA migrates/returns to/from the remote subnet. HIP presents superior performance since the MA(s) assigned to the remote subnet managed domain(s) migrate through the low-bandwidth link with empty state; in the LCF-based solution, the unique MA first visits all the elements local to the manager host subnet and then migrates to the remote subnet (at migration time though, the MA has already collected an amount of data, therefore increasing network traffic).



(a)



(b)

Figure 6. (a) Management cost of SNMP monitoring (per polling interval) against HIP and LCF implementations for varying managed network sizes; (b) Management cost of HIP vs. LCF as a function of the amount of collected data per managed element

In Figure 6b, experiment parameters are adjusted to examine the effect of collected data amounts in the overall cost. The experiments have been performed on the test network of Figure 5b (15 managed elements). Apparently, HIP performance gain over LCF increases as the amount of data collected from each managed element increases. HIP algorithm separates remote subnet hosts into 2, 3 and 4 'virtual' managed domains as data sample size increases to 100, 500 and 2000 bytes respectively.

V. Conclusions & Ongoing Research

Methodologies for designing efficient MA itineraries have received little attention in the literature. In addition to the number of hops realised by a multi-hop agents, the order in which MAs visit their assigned elements (i.e. MA itineraries) is also a crucial factor affecting the overall cost.

This article introduces an algorithm that borrows ideas and concepts from the area of network design to address the issue of MA optimal itinerary planning. The HIP algorithm not only suggests the appropriate number of MAs that should be employed in parallel to minimise the associated cost but also constructs near-optimal itineraries for each of them. Ongoing research involves simulation study of HIP algorithm in large enterprise environments.

References

[1] Adventnet, http://www.adventnet.com.

[2] D. Gavalas, "Mobile Software Agents for Network Monitoring and Performance Management", PhD Thesis, University of Essex, UK, 2001.

[3] A.Kershenbaum, "Telecommunications Network Design Algorithms", McGraw-Hill, 1993.

[4] A. Liotta, G. Pavlou, G. Knight, "Exploiting Agent Mobility For Large Scale Network Monitoring", IEEE Network, 16(3), pp. 7-15, 2002.

[5] K. McCloghrie, M. Rose, "Management Information Base for Network Management of TCP/IP-based Internets: MIB-II", RFC 1213, 1991.

[6] H. Qi, F. Wang, "Optimal itinerary analysis for mobile agents in ad hoc wireless sensor networks", Proc. of the 15th IEEE International Conference on Wireless Communications, 2001.

[7] E. Reuter, F. Baude, "System and Network Management Itineraries for Mobile Agents", Proc. of MATA'02, pp 227-238, 2002.

[8] M.G. Rubinstein, O. C. Duarte, G. Pujolle, "Scalability of a Mobile Agents Based Network Management Application", Journal of Communications and Networks, 5(3), 2003.

[9] WinDump: tcpdump for Windows, http://windump.polito.it/