Performance Analysis of Untraceability Protocols for Mobile Agents Using an Adaptable Framework

Rafał Leszczyna[1] and Janusz Górski[2]

[1] European Commission, Joint Research Centre,
Via Enrico Fermi, 21020 Ispra (VA), Italy
[2] Gdansk University of Technology,
Narutowicza 11/12, Gdańsk, Poland

18 October 2005

Abstract

Recently we had proposed two untraceability protocols for mobile agents and began investigating their quality. We believe that quality evaluation of security protocols should extend a sole validation of their security and cover other quality aspects, primarily their efficiency. Thus after conducting a security analysis, we wanted to complement it with a performance analysis. For this purpose we developed a performance evaluation framework, which, as we realised, with certain adjustments, can be applied to evaluate performance of a whole class of security protocols for mobile agents. This class contains the protocols in which each host on the agent's route is responsible for providing a security function, and, inter alia, includes a big variety of integrity checking schemas and accountability mechanisms for mobile agents such as Lee's Partial Results Authentication Codes or family of Karjoth's et al protocols for protection of computation results of free-roaming agents. Finally, we used the framework to evaluate efficiency of our Untraceability Protocol I. In this paper we present the framework and the results of the evaluation.

1 Introduction

We proposed two security protocols for untraceable migration of mobile agents [1, 2] which have this advantage over other existing untraceability protocols for agents, that they support agents' autonomy in choosing their route [1, 2]. We performed a basic security analysis of the protocols [1, 2] by checking their resistance to known generic and traffic analysis attacks, and currently we are working on extending this analysis. We also applied the protocols to an e-health service to verify their applicability to real IT services [3].

Although in [2] we had presented some implementation suggestions aiming at improving performance of the protocols, the general question of their performance was left open. In this paper we present the results of our recent work aiming at comprehensive coverage of the issue.

To evaluate performance of our protocols we developed a framework. The framework is composed of three modules: an analytic study, related to computation of time complexities; a performance benchmark based on standard performance testing; and a protocols' target environment overhead estimation. The objective of the first part was to obtain the indicator of the protocols' performance, independent of a concrete computer architecture, agent environment and implementation details [4]. The empirical part aims at complementing the analytical results with validation of the simplifying assumptions of the analytical model which was introduced to derive the final complexity equations; and providing a direct view on the protocols' efficiency in a real context. The third module is to obtain a picture of the overhead which will be brought in by the protocols to their target architecture.

In the next sections we will describe the framework and we will present the results of evaluation of our Untraceability Protocol I performed on its basis. Because the description of the framework often refers to our real case, we decided to precede it with a brief presentation of the protocols.

2 Untraceability protocols

Anonymity is the property that ensures that a user may use a resource or a service (the items of interest) without disclosing his/her identity [5]. *Untraceability* is a subclass of communication anonymity [6] assuring that the identities of the message sender and the receiver can not be inferred (and linked) by tracing the message. In case of mobile agents this means that the agent's owner and the target (a particular location or just a statement of the goal) can not be identified by tracing the agent.

Anonymity plays a crucial role for various activities conducted in the Internet. Gülcü et al [7] describe four categories of the activities[1]. These are: discussion of sensitive and personal issues, information searches, freedom of speech in intolerant environments and polling/surveying. For some of these uses, the anonymous access was an enabler and contributed to increasing the popularity of the Internet.

To support anonymity in agent environments we proposed an untraceability protocol for mobile agents – the *Protocol I* which has this advantage over the solutions of Westhoff et al and Enzmann et al [8, 9] that it does not constrain agents' autonomy of choosing a migration path.

In brief, the concept of the protocol is that an agent, while migrating, at each visited host encrypts the identifier of the last visited host (using the public key of the present host) and puts it in the LIFO queue stored in its state. Then, after achieving a goal, when the agent wishes to come back to its base host, it

[1] This is important to note that the authors don't claim this set to be exhaustive.

uses the queue to find its way back. Down the route back the identifiers are subsequently decrypted using each host's private key [2].

We investigated security [1] of the protocol and figured out that the protocol, while being effective, is prone to the costly, nonetheless possible to perform – *cordoning-off attack* – i.e. the attack in which the attacker compromises all hosts surrounding the source one [1]. Consequently we introduced the *Protocol II* which eliminates this vulnerability at the cost of addition of some computation workload to the source host and restriction of the agent's autonomy at the beginning of its route [3]. Hereby we gave choice to the users: they can use more effective but slightly less secure Protocol I, or choose the Protocol II, to receive stronger protection.

Because we do not refer to the second protocol in this article, we decided to skip its details. An interested reader may find them in the [3].

3 Framework

The framework consist of three modules: an analytic study, a performance benchmark and a target environment overhead estimation.

The first module, aiming at estimation of performance of protocols which abstracts of any underlying technological architecture [4], focuses on calculation of time complexity of a protocol as a function of the length of the agent's route.

The second, provides a practical view at the performance of the protocols, allowing to confront the analytic results with the figures obtained during the tests performed in a testing architecture.

The third module gives an image of the overhead brought in by the protocols to their target architecture.

We developed the framework to evaluate performance of our untraceability protocols but we believe it can be applied to the whole class of security protocols for mobile agents.

This class contains the protocols in which each host on agent's route is responsible for providing a security function. For example in case of our Protocol I, each host has to compute the hash value of a concatenation of the preceding container identifier $\mathsf{ID}_{\mathrm{m}-1}$, the own identifier and the successor identifier; then it has to encrypt the $\mathsf{ID}_{\mathrm{m}-1}$, the hash value and a nonce with its secret key; so the agent could add it to the end of its queue of encrypted platform identifiers [3].

Another example is the Simple Partial Results Authentication Codes (Simple PRAC) schema of Lee et al [10]. In the schema a digital signature (and optionally a hash function) has to be computed at each host to assure that the partial results collected at the host are not-repudiable and undeniable.

To the group also belong other variations of Yee's PRAC such as MAC-based PRAC with One-Way Functions [10], and techniques presented by Karjoth et al [11] as well as the Vigna's tracing schema [12]. Apparently most of the methods aiming at detection of tampering with data gathered by a mobile agent will fall into the category since they are based on the similar concepts.

Thus, our framework, although primarily targeting our untraceability protocols, eventually may become very useful to evaluate performance of various other security protocols for mobile agents.

4 Analytic study

The analytic study focuses on calculation and comparing time complexities of security protocol -enabled (protected) and -disabled (plain) agent's migrations. These complexities are calculated as sums of partial times spent on each involved host. Because specific protocols may differ from each other with relation to numbers of these hosts between protected and plain migration (e.g. in our case untraceable migration needs twice as many nodes as plain migration) or other than Java environments might be required to serve as a reference, the study should be performed individually for a particular protocol. For this reason, we do not provide any common template, yet instead, we present derivations corresponding to our untraceability protocol, which we encourage to use as a reference point.

Since Java is the most popular environment for implementation of agent platforms [13] the Java-founded migration mechanisms served as a reference for our analysis. Relying on our experience, we claim that the general code-transfer principles of Java are equally applicable to other environments, and where they are not, they differences are just slight.

In the next subsection we present the reasoning for the protected (untraceable) migration and then the results of adequate deduction applied to the plain migration. Comparing the results will allow us to derive the equation for the overhead induced by employing our security protocol which we present in section 4.3. Implications of this specific formula we will discuss in further part of the paper.

In this section, the term 'time' is used interchangeably with 'time complexity'.

4.1 Protected migration

During the initialisation of its migration, the Untraceable Agent (UA) is given a route to be traversed in order to achieve its goal. Let N be a length of the route. The total migration of UA is composed of $2N$ steps since, according to the Protocol, during the actual migration, the given route is covered twice – first when the agent performs its task, and second when the agent is coming back to the source location. If time of each step performance is denoted as $\ddot{t}[i]$ then the total migration time of UA $\ddot{T}[N]$ is specified as:

$$\ddot{T}[N] = \sum_{i=1}^{2N} \ddot{t}[i] \; . \tag{1}$$

A single migration step roughly involves – suspension of the execution, then serialization of the agent's state and code, and finally sending the resulting data

package to the new location where it is de-serialized and the execution resumes. This implies that the time $\ddot{t}[n]$ is the combination of the following times:

$\ddot{t}_T[n]$ – time of UA's transfer (T) to the container n (from the container $n-1$) through the communication channel,

$\ddot{t}_S[n]$ – time of UA's serialization and de-serialization (S) on the container n,

$\ddot{t}_{TR}[n]$ – time of UA's task realization (TR) on the container n,

$\ddot{t}_{UO}$ – time of UA's untraceability operations (UO) on one container.

The time $\ddot{t}_{UO}$ does not depend on N and is fixed since the untraceability operations always operate on fixed amount of data. The other times can vary: the agent can perform a different task on each location, the agent transfer time depends on characteristics of a transmission medium and the agent (de-)serialization may last longer or shorter proportionally to size of the agent's state. While the first two times are the N-independent, the third includes a component which depends on N – the information about agent's route is maintained in the agent's state.

Because the migration times while heading to the target location and returning to the source can differ significantly, the time $\ddot{t}[i]$ distinguishes between them as follows:

$$\ddot{t}[n] = \begin{cases} \overrightarrow{t}_{UA}[n], & \text{if } 1 \leq n < N; \\ \overleftarrow{t}_{UA}[n-N], & \text{if } N < n \leq 2N; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

where $\overrightarrow{t}_{UA}[n]$ denotes the step performance while going ahead and $\overleftarrow{t}_{UA}[n]$ represents step performance while going backwards.

The time $\ddot{t}_S[n]$ is a combination of the two components:

$$\ddot{t}_S[n] = \ddot{t}_{RS}[n] + \ddot{t}_{RESS}[n] \ , \tag{3}$$

where $\ddot{t}_{RS}[n]$ is the time of UA's route serialization (RS) and $\ddot{t}_{RESS}[n]$ – the time of UA's route excluded state serialization (RESS).

The time $\ddot{t}_{RS}[n]$ can be expressed by the times of atomic serialization operations:

t_{LS} – time of one location serialization (LS),

t_{ELS} – time of one encrypted location serialization (ELS),

– and distinguished between $\overrightarrow{t}_{UARS}[n]$ – the time of route serialization (RS) while UA is heading the target location and $\overleftarrow{t}_{UARS}[n]$ – the time of route serialization (RS) while the UA is coming back to the source:

$$\overrightarrow{t}_{UARS}[n] = [N-n]t_{LS} + nt_{ELS} \ , \tag{4}$$

$$\overleftarrow{t}_{UARS}[n] = [N-n]t_{ELS} \ . \tag{5}$$

Consequently:

$$\begin{aligned}\overrightarrow{t}_{UA}[n] = \ddot{t}_{TR}[n] + \ddot{t}_{RESS}[n] + (N-n)t_{LS} \\ + nt_{ELS} + \ddot{t}_{UO} + \ddot{t}_{T}[n] \ ,\end{aligned} \tag{6}$$

$$\overleftarrow{t}_{UA}[n] = \ddot{t}_{RESS}[n] + (N-n)t_{ELS} + \ddot{t}_{UO} + \ddot{t}_{T}[n] \ , \tag{7}$$

$$\ddot{T}[N] = \sum_{i=1}^{N} \overrightarrow{t}^{UA}[i] + \sum_{i=1}^{N} \overleftarrow{t}^{UA}[i] \ , \tag{8}$$

and the final expression for $\ddot{T}[N]$ is:

$$\begin{aligned}\ddot{T}[N] = N^2 t_{ELS} + 2N\ddot{t}_{UO} + \frac{N(N-1)}{2}t_{LS} \\ + \sum_{i=1}^{N} \ddot{t}_{TR}[i] + \sum_{i=1}^{2N} \ddot{t}_{RESS}[i] + \sum_{i=1}^{2N} \ddot{t}_{T}[i] \ .\end{aligned} \tag{9}$$

4.2 Plain migration

After applying the analogous reasoning and notation to the Traceable Agent (TA), and keeping in mind that TA needs only to cover a one-way route (N locations), after which it moves back directly to the source location (1 migration step), the following equations describe the relevant times for TA:

$$\dot{T}[N] = \sum_{i=1}^{N} \overrightarrow{t}_{TA}[i] + \overleftarrow{t}_{TA} \ , \tag{10}$$

$$\overrightarrow{t}_{TA}[n] = \dot{t}_{TR}[n] + \dot{t}_{RESS}[n] + (N-n+1)t_{LS} + \ddot{t}_{T}[n] \ , \tag{11}$$

$$\overleftarrow{t}_{TA}[n] = \dot{t}_{RESS}[n] + \ddot{t}_{T}[n] \ . \tag{12}$$

And from the above we can convert (10) into:

$$\begin{aligned}\dot{T}[N] = \frac{N(N+1)}{2}t_{LS} + \sum_{i=1}^{N} \dot{t}_{TR}[i] \\ + \sum_{i=1}^{N+1} \dot{t}_{RESS}[i] + \sum_{i=1}^{N+1} \dot{t}_{T}[i]\end{aligned} \tag{13}$$

4.3 Overhead

The overhead introduced by the untraceability protocol can be calculated as a ratio:

$$\frac{\ddot{T}[N] - \dot{T}[N]}{\dot{T}[N]} \cdot 100\% \; . \tag{14}$$

where the value of $\ddot{T}[N] - \dot{T}[N]$ is given as following:

$$\begin{aligned} \ddot{T}[N] - \dot{T}[N] = N^2 t_{ELS} + 2N\ddot{t}_{UO} \\ + (N-1)\ddot{t}_{RESS} + \sum_{i=1}^{N-1} \dot{t}_T[i] - N t_{LS} \end{aligned} \tag{15}$$

It is assumed that the parameters not related to untraceability are equal, i.e. n: $\dot{t}_{TR}[n] = \ddot{t}_{TR}[n]$ and $\dot{t}_T[n] = \ddot{t}_T[n]$ and $\dot{t}_{RESS}[n] = \ddot{t}_{RESS}[n] = t_{RESS} = const$.

5 Performance benchmark

The testing module is composed of a testing architecture and a series of tests to be performed upon it.

5.1 Testing architecture

To obtain meaningful test results, tests should be performed either directly in the target environment or in an environment that is sufficiently close to the target one. Since at security protocols' design and validation stages authors of protocols usually do not know the target environment, and yet because security protocols are usually designed to cover various areas of problems (both cases applied to us), only the second option remains available.

To reach the feature of sufficient proximity between testing and possible target environments, when designing our testing architecture, we paid particular attention to its flexibility and configurability. In other words, we aimed at providing an architecture, which with one setting of 'switches' would be similar to one predicted target environment, while with another setting it would become close to another target environment.

For this reason, firstly we had to identify the characteristics of computational environments which have the strongest impact on performance of the environments.

Each agent environment is composed of three layers: the agents layer, comprising agents of any type; the underlying containers layer – a middleware allowing agents' operation; and the hosts layer which aggregates hosts interconnected via a computer network.

layer	characteristics
Agents Layer	number of agents present in the environment, diversity of agents behaviours (including agent relocation), agents intercommunication, agents implementation, agents resources, agents control (autonomous agents, user-controlled agents etc)
Containers Layer	number of containers, containers deployment, services resources
Hosts Layer	number of hosts, hosts efficiency, inter-host communication (communication protocols, communication medium)

Table 1: Characteristics of agent environments of the strongest influence on performance of the environments.

When extracting the characteristics of interest we examined each of the three layers respectively. Results of the study are listed in the table 1[2].

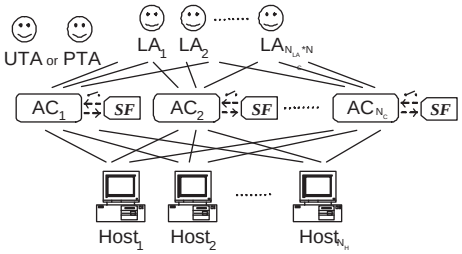

Figure 1: Benchmarks architecture.

After receiving the above characteristics we developed a three-layered testing architecture (Figure 1) in which we can control a subset of the characteristics. The architecture is realised in Java Agent DEvelopment Framework (JADE – `http://jade.tilab.com/`) since it appears to be most popular agent middleware [13].

In our architecture, the agents layer is composed of Load Agents (LAs), amount of which is flexibly adjustable. As the name indicates, the main aim of LAs is to load the environment and consume its resources. This must be done quite similarly to as original agents would do. Thus, each LA actively operates in the system – it migrates and 'realises tasks'. The task realisation is simulated by stopping LA on a container for a random period in time. Despite the fact that the container resources are not consumed directly in sense of being utilised by a working agent, the LA's presence on a container requires reservation of an execution space for LA. The stay-time is chosen with equal probability from the range between 10 milliseconds and 10 seconds – corresponding to various kinds of agent tasks. Another type of diversity of agents' behaviours, is simulated by introducing two types of LAs – Protected LAs (PLAs) and Unprotected LAs

[2]It is important to note that the authors do not claim this set to be exhaustive.

(ULAs), what allows to examine the behaviour of the environment depending on the level of use of protection (usually only a certain subset of agents requires protection).

Since our untraceability protocols do not involve any agent intercommunication and a user-control we decided not to include these characteristics of agents layer into our architecture. We also did not examine nuances of implementation since our protocols straightforwardly follow specifications and do not leave a space for performance-focused improvements.

In the containers layer we can configure a number and deployment of containers and to enable or disable protection. Originally we used untraceability but it may be conveniently replaced by any other measure by means of JADE add-ons. We can also enable another services, which could consume (or provide) additional resources of environment.

The hosts layer is very independent of the actual implementation of our architecture and can be flexibly configured according to the hardware resources available to testers.

In such simulated environment test agents are run. Each such agent can be either an Unprotected Test Agent (UTA) or a Protected Test Agent (PTA). They are identical except that the first one utilises and the second one does not utilise the untraceability service.

During a test each test agent has to follow the route of N_{IC} length N_R times. The UTAs and PTAs are following the same routes. The time of passing every round is measured and recorded as well as the time of passing all N_R rounds. The precision of time measurements is 10 milliseconds [3].

5.2 Tests

The objectives of the tests were to:

- examine if and to which extent enabling protection affects performance of agents which do not use it,
- compare performance of protected agents (PAs) and unprotected agents (UAs),
- analyse performance of PAs as a function of length of the agent route and the agent environment load.

While comparing performance of UAs and PAs the two most common agent migration scenarios were taken into consideration[4]:

Agent Migration Scenario I – the agent has its goal formulated in general terms so the it has to traverse multiple locations in order to satisfy the goal (e.g. web crawling)

[3] We have used java `System.currentTime()` – being the most precise of the program execution measuring methods which do not require hardware devices [14].

[4] In figures we use the notation: N_C – total number of containers, N_{IC} – number of Intermediate Containers (IC).

Agent Migration Scenario II – the agent goal is to visit its target location and to perform its task there (e.g. bandwidth conservation scenarios – in which an agent is sent to a remote host in order to reduce multi-host communication)

Figures 2 – 5 illustrate the scenarios applied to our untraceability protocol.

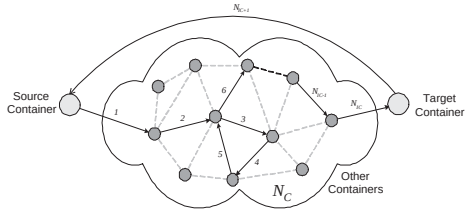

Figure 2: Agent Migration Scenario I realized by TA: TA traverses N_{IC} containers and in N_{IC+1} step returns to the source container.

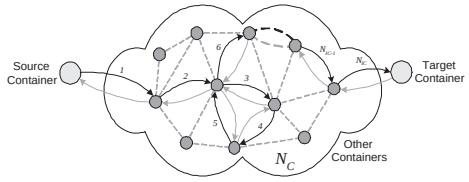

Figure 3: Agent Migration Scenario I realized by UA: UA traverses N_{IC} containers and then has to cover the traversed route again (in reversed order) to return to the source container.

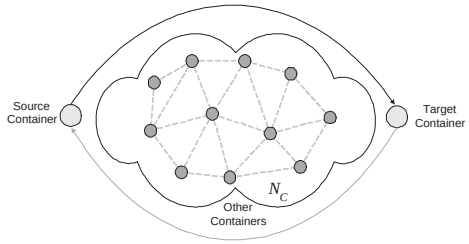

Figure 4: Agent Migration Scenario II realized by TA: TA simply 'jumps" directly to the target container and after realizing its task returns to the source container.

In each case, after achieving the goal the agent returns to the source container.

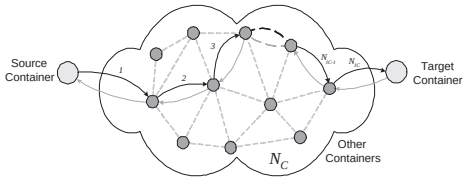

Figure 5: Agent Migration Scenario II realized by UA: Before reaching the target container the UA has to cover an intermediate route of a length N_{IC} in order to hide a migration trace. This route must be traversed again by the UA when returning to the source container.

6 Target environment overhead estimation

As a third module of the evaluation framework we propose estimation of overhead brought in by a security protocol to the foreseen target environment.

At this stage it is necessary to obtain the specification of the target environment allowing the most precise determination of functional types of agents required in the environment and their quantitative proportions. With these figures it is possible to assess the amount of the agents actually utilising the security protocol, and on its ground, to evaluate the real overhead which the security protocol will bring to the environment.

We will illustrate this on the example of our protocols.

One of the applications where users anonymity (so untraceability) plays crucial role is anonymous e-health counselling. Anonymous e-health counselling is a service allowing IT-based provision of health related advices on an anonymous basis. Another words, it provides all functionalities for performing the scenarios where a patient using an IT system asks for a health advice, yet wishes to do it anonymously (because, for example, they suffer from an embarrassing disease or from an addiction) [15].

This application, which for the first sight seems to be quite dedicated, in reality it represents a large area of (often used) applications in which the schema of agent-based accessing multiple knowledge sources in order to obtain a comprehensive information is implemented [3].

Thus we developed an experimental application providing the e-health counselling service [3] which takes advantage of untraceable agents – to provide ourselves the opportunity to observe how our protocols behave in a real settlement.

And the e-health environment we treated as the target environment when estimating the level of the total untraceability overhead brought in by enabling the untraceability service.

The target health environment involves many stakeholders participating in the process of medical service delivery: patients, health professionals, pharmacists, dietitians etc. In the technological e-health platform, a stakeholder is represented by an agent, – its *Virtual Ego*. Apart of Virtual Ego agents, various other agents are present in the platform [3]: Knowledge Discovery Agents (KDA), Information Agents (IA) and a compound group of Auxiliary Agents

active patients	patients VE	health physicians VE	KDA	IA	AA
10000	10000	34	5000	5000	30000

Table 2: Numbers of different mobile agents running (active) in the health system for 10000 active patients.

(AA) – encompassing different types of agents responsible for performing supportive tasks (Figure 6).

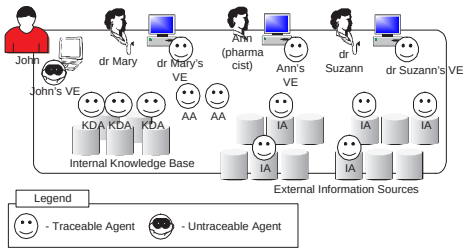

Figure 6: Mobile agent based health platform.

For such platform we estimated the proportion of active (running) agents in the system as a function of N_{PU} – the number of patients using the system at a particular moment ($N_{PU} \leq N_P$, where N_P is the number of all agents registered in the system). It is worth to note that besides these active agents also temporarily suspended agents exist in the system, which wait for dispositions of their owners. Our prognoses are based on the report of Royal College of Physicians of London [16] saying that average number of practising physicians per 1000 inhabitants in Europe equals 3.4 and the surveys conveyed under the International Network Health Policy & Reform project indicating average 6.19 health consultations per person taking into account ten European countries [17]; and our experience in design of IT systems.

The results (assuming $N_{PU} = 10000$) are presented in the table 2.

With the figures we estimated the overhead. The results of the estimation we present in the next section.

7 Evaluation Results

7.1 Analytic study

The equation (15) for the overhead introduced by the untraceability protocol, indicates polynomial (quadratic) dependence of $\ddot{T}[N] - \dot{T}[N]$ on N. The Untraceability-Related Component (URC) ($N^2 t_{ELS} + 2N\ddot{t}_{UO}$ – with a higher significance of the first component of the sum) grows faster than the Migration-Related Component (MRC) ($(N-1)\ddot{t}_{RESS} + \sum_{i=1}^{N-1} \dot{t}_T[i]$). However the real value of t_{ELS} is at least four orders of magnitude smaller than the value of MRC,

which means that not until N reaches very high values ($N \geq 10000$), the influence of URC becomes visible. In practical agent applications, agents scarcely (if ever) have to cover routes of such lengths. It must be also remembered that assigning a route of that large length to a traceable agent, leads to dominance (because of the quadratic growing with reference to location serialization) of the route-storage component as well.

7.2 Performance benchmarks

The following benchmarks were performed basing upon our testing architecture:

Test I – Comparing the non-protected agent (the traceable agent – TA) performance in the untraceability enabled and untraceability disabled environments, $N_{IC} = N_C = 100$, the random stay-time is turned off,

Test II – Comparing migration times of the traceable agent (TA) and the untraceable agent (UA) as the function of N_{IC}, $N_{IC} = NC \in 20, 40, 60, 80, 100, 120, 140$ (refers to Migration Scenario I),

Test III – UA performance dependent on N_{IC}, $1 \leq N_{IC} \leq 64$, $N_C = 150$ (refers to Migration Scenario II),

Test IV – Comparing UA performance in variably loaded environments. Due to identified problems with JADE this test was performed for $N_C = 60$ and only three $N_L A$ values – $N_L A \in 60, 120, 180$.

At the current stage of our research all benchmarks were performed on one host[5], but we consider extending them to a higher number of hosts in the future.

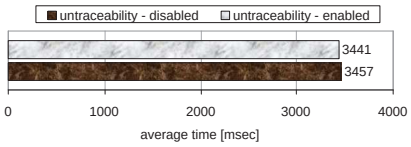

Figure 7: Test I – Comparing TA performance in untraceability enabled and untraceability disabled environment, $N_{IC} = N_C = 100$.

The results of Test I (see Fig. 7) empirically prove that enabling the untraceability service does not influence the performance of traceable agents[6]. This is not surprising since the mobile agents do not use any functions from the service. However the question was if the fact that the service is enabled on the containers (and takes some resources) might affect the performance.

[5]Intel®Pentium®4, 3 GHz, 1.5 GB RAM. Java^{TM}2 Runtime Environment, Standard Edition (build 1.4.2_01-b06), Java HotSpot^{TM}Client VM (build 1.4.2_01-b06, mixed mode) – run with default settings.

[6]The slight difference of values (this time, surprisingly –) in favour of the untraceability enabled environment is in the range of standard deviation which is equal to 110 [ms].

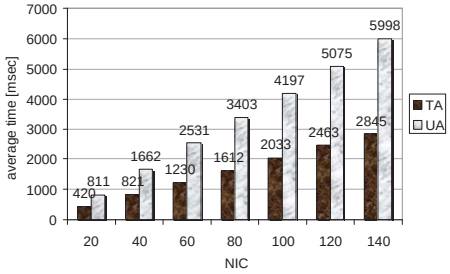

Figure 8: Test II – Comparing migration times of TA and UA dependent on N_{IC} (refers to Migration Scenario I).

The results of Test II (Figure 8) confirm the intuition, that for realistic values on N_{IC}, the dominant factor influencing the performance is agent migration – the approximately linear growth of time consumption in dependence on N – practically the same for TA and UA (the linearity coefficient is roughly equal to 1 – to cover two times longer route agent needs approximately two times more of time) – as it could be inferred from the estimation of complexity. Consequently, the dominant role of the migration factor affected the overhead, which oscillates on the level of around 100% – UA must again cover the already traversed route whereas TA just 'jumps' from the target location back to the source.

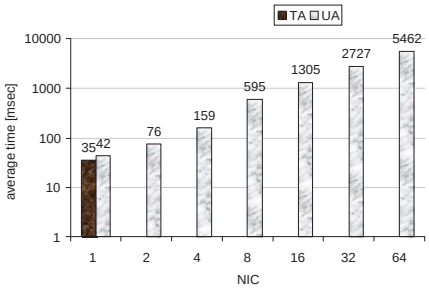

Figure 9: Test III – UA performance dependent on N_{IC} (refers to Migration Scenario II).

The meaningful result is obtained from Test III – when sending UA the user has to be conscious that its performance is directly proportional to the length of untraceability route (Figure 9).

Test IV confirms the intuition that if the number of the load agents approaches the limit resulting from the capacity of the agent environment, the performance falls significantly (as the result of the shortage of the operational resources) (Figure 9).

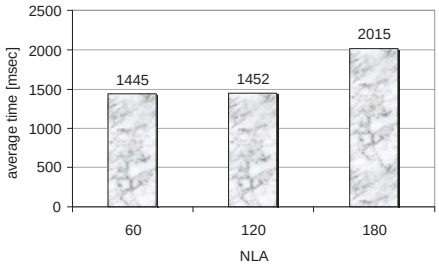

Figure 10: Test IV – Comparing UA performance in variably loaded environments.

7.3 Target Environment Estimated Overhead

Assuming that every tenth patient want's to take advantage of a medical service untraceably, only 1000 of total 50034 active agents will introduce the overhead related to untraceability to the platform. Having in mind that each individual UA introduces around 100% of the TA overhead, and representing a performance of each TA as 1 Unit of Performance (UP), for 50034 UP's – related to the presence of 50034 agents in the system – the 1000 UA's will induce additional 1000 UP's. Consequently the total overhead (we must underline that the overhead is related solely to a time of migration and not with any other agent activities) will be almost 2%.

8 Summary

The paper presented a performance evaluation framework which although originally designed for our untraceability protocols, may be applied to a much wider category of security protocols for mobile agents. This category include all the protocols where during migration of agent, a security function is computed at each host passed. The framework is based on three modules. The first, analytic module aims at deriving a mathematic formulae for the overhead imposed by the employment of the protection, and involves computation of time complexities of standard and protected agents. In the empirical part we provide a testing architecture and a series of tests to be performed upon it, in order to obtain 'real' performance characteristics of protected and unprotected agents. The third module requires forecasting the structure of the environment to which the protection will be added and it results in estimation of the overall overhead brought in by the protection.

With the architecture, we carried out the evaluation of our Untraceability Protocol I. We derived the universal (not tied to any implementation) symbolic equation for the overhead as a function of the length of the agent's route. Then we performed the tests, results of which confirm the analytical estimates and provide more concrete data related to the Java enabled agent platforms. The

general result is that untraceability cost (in terms of time overhead resulting from traversing a given route) is twice as much as the cost without untraceability. A good news is that this cost does not depend on the tasks performed by the agent. It should be also remembered that the real agent applications would likely be composed of both – traceable and untraceable – agents, so the resulting total overhead would be relatively small (see Section 7.3).

References

[1] Rafał Leszczyna and Janusz Górski. Untraceability of mobile agents. Technical report, European Commission, Joint Research Centre, Institute for the Protection and security of the Citizen, December 2004.

[2] Rafał Leszczyna and Janusz Górski. Untraceability of mobile agents. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05)*, volume 3, pages 1233–1234, Utrecht, the Netherlands, July 2005.

[3] Rafał Leszczyna. The solution for anonymous access of it services and its application to e-health counselling. In *Proceedings of the 1st 2005 IEEE International Conference on Technologies for Homeland Security and Safety (TEHOSS '05)*, volume 1, pages 161–170, Gdańsk, Poland, September 2005. Gdańsk University of Technology.

[4] Marek Kubale. *Introduction to computational complexity and algorithmic graph coloring.* Gdańsk Scientific Society, 1998.

[5] National Institute of Standards and Technology (NIST). *Common Criteria for Information Technology Security Evaluation - Part 2: Security Funtional Requirements.* U.S. Government Printing Office, 1998.

[6] Andreas Pfitzmann and Marit Hansen. Anonymity, unobservability, and pseudonymity - a proposal for terminology. draft v0.23. August 2005.

[7] Ceki Gülcü and Gene Tsudik. Mixing email with Babel. In *Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96)*, page 2. IEEE Computer Society, 1996.

[8] Dirk Westhoff, II Markus Schneider, Claus Unger, and Firoz Kaderali. Protecting a mobile agent's route against collusions. In *Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography*, pages 215–225. Springer-Verlag, 2000.

[9] Matthias Enzmann, Thomas Kunz, and Markus Schneider. Using mobile agents for privacy amplification in the trade with tangible goods. In *6th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI)*, volume IV, Orlando, Florida, USA, July 2002.

[10] Bennet S. Yee. A sanctuary for mobile agents. In *Secure Internet Programming*, pages 261–273, 1999. citeseer.ist.psu.edu/article/yee97sanctuary.html.

[11] Günter Karjoth, N. Asokan, and C. Gülcü;. Protecting the computation results of free-roaming agents. In *MA '98: Proceedings of the Second International Workshop on Mobile Agents*, pages 195–207, London, UK, 1999. Springer-Verlag.

[12] G. Vigna. Protecting mobile agents through tracing. In *Third Workshop on Mobile Object Systems*, 1997. Available at https://citeseer.ist.psu.edu/vigna97protecting.html.

[13] Rafał Leszczyna. Evaluation of agent platforms. Technical report, European Commission, Joint Research Centre, Institute for the Protection and security of the Citizen, Ispra, Italy, June 2004.

[14] David B. Stewart. Measuring execution time and real-time performance. In *Embedded Systems Conference*, San Francisco, April 2001. Embedded Research Solutions. http://www.embedded-zone.com/Publications/mezexec.pdf.

[15] EU IST-2002-507591 PRIME. Requirements version 0 part 3: Application requirements.

[16] Royal College of Physicians of London. European working time directive: European dimensions. Internet, November 2002. Available at http://www.rcplondon.ac.uk/college/statements/doc_ewtd_european.asp. Last accessed: October 2005.

[17] International Network Health Policy & Reform. Health policy monitor: Country facts. Internet, April 2005. Available at http://www.healthpolicymonitor.org/en/index.html. Last accessed: October 2005.