Detecting Reputation Variations in *P2P* Networks

THEODORA DARIOTAKI
The University of Athens, 15771, Athens, Greece

ALEX DELIS
The University of Athens, 15771, Athens, Greece

Abstract

Peer-to-Peer overlay networks have gained in popularity as they present an effective alternative to resource-sharing. Users' anonymity though, has allowed peers to also share malicious artifacts that may be wrongly perceived as popular resources. A number of proposed reputation schemes attempt to address this problem by recording the requestor's satisfaction at the end of each operation. In such schemes, a node that has no evidence on quality of a sought resource may consult with recommending peers before a download operation commences. Existing reputation mechanisms are prone to attacks that exploit the formation of groups of colluding peers. The latter collectively attempt to either raise or lower one's reputation value. We propose a *Reputation Monitoring Mechanism (RMM)* that restricts the extend of those attacks by continually monitoring the reputation level of each peer over a number of consecutive time periods or *epochs*. Should *RMM* observe rapid changes in the reputation value of a node, it smooths out these variations; thus, it provides resistance to attacks by colluding peers. *RMM* actions depend on both peer reputation value and variations during several *epochs* in recent past.

Keywords

Reputation Schemes in *P2P* Networks, Distributed Evaluation of Reputation, Node and Resource Reputation Reporting.

1 Introduction

Peer-to-Peer (P2P) networks have become a very attractive choice for distributed resource sharing as they shield users from the inherent heterogeneity of participant computing systems. Due to lack of inspection and/or assessment of shared resources, a peer might be deceived and be led to download a malicious or simply an irrelevant object. To prevent fraud, *P2P* networks incorporate *reputation* *schemes* [2, 13, 3, 11, 9, 15, 18] that offer advice to peers regarding reliability of unknown resources and/or resource-holders.

Upon a number of possible offers, the requesting site has to consult with recommending peers in the overlay network, termed recommenders, in order to determine the most reputable offerers and resources to download. This process is vulnerable to *pseudospoofing* and *shilling attacks* [12]. *Pseudospoofing* attackers establish and control multiple virtual peers. Whenever an attacker's reputation level is low, it may turn to a new pseudonym and/or create virtual yet fake identities. These identities will with certainty spread false evidence in favor of the attacker. In *Shilling*, multiple peers with real IP addresses are actually created and/or controlled, and are not just simulated, by the attacker. *Pseudospoofing* is usually addressed by initiating a challenge-response handshake between the requestor and the host of a resource to be downloaded so that the originality of the offerer is established [5]. *Shilling* is certainly more difficult to deal with, as the attackers control legitimate peers with real IP addresses. The only known detection mechanism clusters votes originated from similar IP addresses and considers them as a single vote having an average value [6]. However, an attacker that creates and controls multiple diverse peer identities may exploit these multiple identities to either increase its reputation or reduce the reputation of others.

Since the number of peers deceived by the colluders continuously increase, eventually the number of complaints may counterbalance the colluders' attempt to tamper with the reputation level of a site and/or resource. In this way, a reputation scheme may take a long time before it detects *pseudospoofing/shilling* activity. A malicious peer with artificially increased reputation may exploit this delay to distribute malicious artifacts. In this context, we propose a distributed *Reputation Monitoring Mechanism (RMM)* that monitors the reputation levels of peers. *RMM* does not only limit the extent of attacks but more importantly attempts to significantly reduce the time in which malicious activity goes undetected. As colluding peers may cause rapid changes at the reputation levels of sites/resources, the *RMM* attempts to detect significant variations of reputation values and limit the extent of the sought change. Once candidate nodes for unlawful activity are determined, the *RMM* reprimands these sites following a policy of proportional penalties to their current level of trustworthiness. Section 2 presents the fundamental structures used and section 3 outlines a baseline reputation scheme. Section 4 introduces the *RMM* and section 5 discusses the core features of our proposed mechanism. Related work and conclusions are found in sections 6 and 7 respectively.

2 Basic Data Structures

The required data structures that both the *Baseline Reputation Scheme (BRS)* and our *Reputation Monitoring Mechanism (RMM)* use are depicted in Table 1. Both *Direct Peer Experience (DPE)* and *Direct Resource Experience (DRE)* tables are the fundamental structures maintained at every peer p in the overlay network. They help store the "experience" for all download-requests initiated by a peer q in reference to nodes that are willing to act as servers for resource s. Both nodes and distributed resources can be affiliated with such experience ratings. Along with identifiers of q, p, and s, these ratings form tuples whose constituent elements quantify the experience obtained after the completion of sessions between resource holders and query initiators. The experience concerning a peer may be computed based on a combination of features including the honesty of a site (i.e. if it really holds the resource requested), the average response time (determined by its computation power and its load), the available bandwidth through which it connects to the network, and finally, the network latency between the requestor and the resource holder. The resource experience can be also computed based on quantitative characteristics that include authenticity, level of quality, and relevance to the query criteria. Every entry in the two experience tables regarding a peer p or a resource s contains d_p or d_s values respectively.

Data Structure	*Definition*
Direct Peer Experience table (DPE)	Stores experience $E(q,p,t)$ of q after interacting with site p at time t regarding p's behavior represented by d_p distinct features that are parameters to peer reputation.
Direct Resource Experience table (DRE)	Stores experience $E(q,p,s,t)$ that q estimates on the resource s located in p at time t expressed by d_s distinct features that are parameters to resource reputation.
Reputation Table (RT) applicable to *RMM* only	For all resource-holders under the responsibility of a *Reputation Variation Monitor (RVM)*, the entries to this table consist of the reputation levels (comprised of the values of the d_p contributing features) observed during the last λ epochs. For each resource-holder, *RT* also stores the identities of all other *RVM*s also responsible for specific peers.

Table 1: *BRS* and *RMM* pertinent data structures

In our *RMM* mechanism, each node that is part of the overlay network, is monitored by a set of *Reputation Variation Monitors (RVM)* peers. Every *RVM* site maintains a *Reputation Table (RT)* with reputation levels of managed peers during the last λ observed time periods, termed *epochs*. *RVM* sites are a subset of the network peers and may undertake the management of reputation levels for multiple nodes. Since all above information is distributed, single point of failure is avoided. This helps advance the scalability of the reputation system's overall design. In the rest of the discussion, we will refer to a group of offerers as $\mathbf{p}$ and a single offering node as $\mathbf{p_i}$. It is worth pointing out that as the experiences of q are maintained in tables stored at q, there is no need of repeating the identity of the site in question in the corresponding records. However, we use this extended format for readability.

3 Baseline Reputation Scheme (*BRS*)

The Baseline Reputation Scheme (*BRS*) that we discuss here is derived from the description of the *XRep* protocol [6]. We adopt the distinct five phases of *XRep*. In addition, we offer estimation formulae for the evaluation of all factors involved. Below, we briefly present the five phases of *BRS*.

• **Phase I – Resource request:** when an initiator q searches for a resource s, it sends an `AskResource` query message to its *Direct Neighbor Set (DNS)*. The `AskResource` query requests s and states the criteria that either the resource or the resource-holder must satisfy. *DNS* is defined as the group of nodes to which q is connected through a direct link and the criteria correspond to features outlined in Section 2. Each peer in the *DNS* examines whether it possesses the requested resource s that may also satisfy the imposed query criteria. In light of a positive outcome, a node responds to the initiator with a `HoldResource` message that includes the name of the resource offered as well as the properties of both resource and peer. Otherwise, the query is forwarded to q's *Indirect Neighbor Set (INS)* in a recursive fashion until a number of predefined number of hops h is reached. To prevent cycles in the forwarding motion and reduce network traffic, if a query reaches the same peer more than once with a request coming from the same point of origin, it is neglected.

• **Phase II – Recommendation Request:** after receiving `HoldResource` messages from multiple offerers in set $\mathbf{p}$ and checking their properties[1], the initiator q solicits recommendations for all members of $\mathbf{p}$ and s even if it has already recorded prior experiences with the same peers/resource. If q would reject a specific peer $\mathbf{p_i}$ offer with whom q had prior insufficient satisfaction, then q would blacklist $\mathbf{p_i}$. From this point on, $\mathbf{p_i}$ may depart from the blacklist only if q takes into account indirect evidence. Consequently, q dispatches an `AskRecom` message to *DNS* neighbors in a similar way used in the handling of `AskResource` messages. Upon receipt of an `AskRecom`, a peer examines both its experience tables (*DPE* and *DRE*) for the resource s and the offerers in set $\mathbf{p}$ and ships the results back to the initiator q using a `PostRecom` message. Public key infrastructure (PKI) [1] may be used to ensure integrity and authenticity of messages.

• **Phase III – Evaluation of Offerer Reputation:** the initiator q may receive several recommendations from trustworthy, un-trustworthy, and unknown[2] peers. This set is denoted as $\mathbf{r}$ and we term them as first-line recommenders. q searches in its *DPE* for any prior experience with the set of peers $\mathbf{r}$. In addition, q solicits from both its *DNS* and *INS* neighbors their own direct experiences in reference to

[1] For example, that the network bandwidth is greater than the requested limit.

[2] nodes that have not yet interacted with q.

a particular recommender $\mathbf{r_i}$; the aggregation of these values makes up the indirect experience of q about a recommender. If the combined expression of direct and indirect experiences exceeds a user-defined threshold θ, then q accepts the recommendation; otherwise, the recommendation is discarded. If first-line recommending sites are unknown, then q has to evaluate their trustworthiness before accepting their recommendation. To achieve this, q requests direct experiences that a set of second-line recommenders $\mathbf{r}'$ has recorded for each of the first-line recommending sites $\mathbf{r_i}$. Initiator q accepts advice from second-line recommenders $\mathbf{r}'$ if q's direct experience with the individual sites in question exceeds θ.

Let $DPE_{q,\mathbf{p_i},\tau}$ be the *Direct Peer Experience* the initiator q has on the offerer $\mathbf{p_i}$ at this present time τ. This can be derived from the experience entries $E(q,\mathbf{p_i},t)$ of q's *DPE* table weighted by a *time decay function* $f(\tau,t)$. As defined earlier, $E(q,\mathbf{p_i},t)$ states the satisfaction of q regarding node $\mathbf{p_i}$ at distinct times t where $t \leq \tau$. The time decay function is used to express the fact that a recent peer experience deserves more attention than an old one since peers may have behaved differently in the elapsed time. Assuming that q has g entries in its *DPE*, then q's direct peer experience is defined as the weighted average of those entries:

$$DPE_{q,\mathbf{p_i},\tau} = \frac{1}{g}\sum_{j=1}^{g} E(q,\mathbf{p_i},t_j) * f(\tau,t_j) \tag{1}$$

q proceeds with the computation of indirect peer experience $IPEq,\mathbf{p_i},\tau$ that it gathers at time τ from n reputable recommending peers concerning the individual site $\mathbf{p_i}$:

$$IPE_{q,\mathbf{p_i},\tau} = \frac{1}{n}\sum_{k=1}^{n} DPE_{\mathbf{r}_k,\mathbf{p_i},\tau} \tag{2}$$

The selection of the n first-line reputable peers is accomplished by computing $DPEq,\mathbf{r_i},\tau$ (Eqn. 1) for each $\mathbf{r_i}$ in the set. Only sites whose value exceeds the threshold θ qualify for the set. If $\mathbf{r_i}$ is unknown to q, then q may dispatch a new query to its vicinity asking for recommendation on $\mathbf{r_i}$. The advice of second-line recommending site $\mathbf{r}'_\mathbf{i}$ with which q has sufficient direct experience (i.e., $DPE_{q,\mathbf{r}'_\mathbf{i},\tau} > \theta$) are accepted.

After having computed both $DPEq,\mathbf{p_i},\tau$ and $IPEq,\mathbf{p_i},\tau$, the initiator q may compute the reputation level of $\mathbf{p_i}$ by assigning these factors different weights. The coefficient a, defined by user application level, ranges in $[0,1]$ and indicates how much the initiator wishes to take into account recommenders' experiences [2]. When $a < 0.5$ the initiator relies more on the recommenders' experience than on its own. The *Reputation Level* of a peer $\mathbf{p_i}$ is then defined as:

$$RL_{q,\mathbf{p_i},\tau} = a * DPE_{q,\mathbf{p_i},\tau} + (1-a) * IPE_{q,\mathbf{p_i},\tau} \tag{3}$$

Similarly, we compute the *Reputation Level* of resource s at site $\mathbf{p_i}$ based on the recommendations received. The *Direct Peer Experience* that the initiator q has on a resource s originating from peer $\mathbf{p_i}$ at this present time τ is termed $DRE_{q,\mathbf{p_i},s,\tau}$. We can derive $DRE_{q,\mathbf{p_i},s,\tau}$ using the v entries $E(q,\mathbf{p_i},s,t)$ found in the *DRE* table of q for v instances of time where $t \leq \tau$. $IRE_{q,\mathbf{p_i},s,\tau}$ is the *Indirect Resource Experience* the initiator q obtains at time τ from m recommending peers. This can be computed by the *DRE* records of the recommenders.

$$DRE_{q,\mathbf{p_i},s,\tau} = \frac{1}{v}\sum_{u=1}^{v} E(q,\mathbf{p_i},s,t_u) * f(\tau,t_u) \tag{4}$$

$$IRE_{q,\mathbf{p_i},s,\tau} = \frac{1}{m}\sum_{l=1}^{m} DRE_{\mathbf{r}_l,\mathbf{p_i},s,\tau} \tag{5}$$

In summary, the *Reputation Level* of a resource s located at $\mathbf{p_i}$ is defined as:

$$RL_{q,\mathbf{p_i},s,\tau} = a * DRE_{q,\mathbf{p_i},s,\tau} + (1-a) * IRE_{q,\mathbf{p_i},s,\tau} \tag{6}$$

We note that the behavior of a node may be independent of the reputation of its hosted resources and vice versa [3, 6, 18].

• **Phase IV – Offerer Selection:** an offerer becomes a candidate for downloading the sought resource s only if the expression $(RL_{q,\mathbf{p_i},\tau} > \theta)$ AND $(RL_{q,\mathbf{p_i},s,\tau} > \theta)$, qualifies. Frequently, reputation schemes select the most reputable peer among the candidates [6, 14]. As the latter may produce high load for a few nodes, a random choice among the top candidates can be used instead [15]. Before proceeding with downloading from the selected trustworthy offerer $\mathbf{p_w}$, the initiator q begins a challenge-response handshake with $\mathbf{p_w}$ to prevent pseudospoofing [12]. If $\mathbf{p_w}$ fails the test, another offerer of the top-list is selected until the identity of the new offerer $\mathbf{p'_w}$ is verified.

• **Phase V – Resource Download and Experience Updates:** node q sends a *DownloadReq* message to $\mathbf{p_w}$ asking for s. Once the download of s completes, the initiator q records its experience regarding the offering peer $\mathbf{p_w}$ and the downloaded resource s at transaction time t_n. This is materialized by adding a record $E(q,\mathbf{p_w},t_n)$ in q's *DPE* table and a record $E(q,\mathbf{p_w},s,t_n)$ in q's *DRE* table. Similarly, q stores to its *DPE* structure its own opinion on the first- and second-line recommender peers in sets $\mathbf{r}$ and $\mathbf{r'}$ that expressed their opinion on $\mathbf{p_w}$.

Over time, a node $\mathbf{p_i}$ may occasionally provide service as either offerer or recommender to an initiating site q. The level of satisfaction is stored at q structures (i.e., *DPE*) no matter what is the type of service rendered by p in the past. Algorithms 1-4 show the functions of our baseline reputation scheme (*BRS*) and Figure 1 depicts a sample session on how a peer is selected.

Algorithm 1 (*BRS*) - Initiator's Algorithm

1: broadcast `AskResource(`s,h`)` /* h number of hops */
2: receive `HoldResource(`s`)` from $\mathbf{p}$ /* the set of offering peers */
3: broadcast `AskRecom(`$\mathbf{p},s,h$`)`
4: receive `PostRecom(`$\mathbf{p},s$`)` from $\mathbf{r}$ /* the set of first-line recommenders */
5: **for all** $\mathbf{r_i}$ **do**
6:    compute $DPE(q,\mathbf{r_i},\tau)$
7:    broadcast `AskRecom(`$\mathbf{r_i},h$`)`
8:    receive `PostRecom(`$\mathbf{r_i}$`)` from $\mathbf{r}'$ /* set of second-line recommenders */
9:    **for all** $\mathbf{r}'_\mathbf{k}$ **do**
10:       **if** ($DPE(q,\mathbf{r}'_\mathbf{k},\tau) > \theta$) **then**
11:          accept recommendation of $\mathbf{r}'_\mathbf{k}$ for recommender $\mathbf{r_i}$
12:       **end if**
13:    **end for**
14:    compute $IPE(q,\mathbf{r_i},\tau)$
15:    **if** ($DPE(q,\mathbf{r_i},\tau) > \theta$) AND ($IPE(q,\mathbf{r_i},\tau) > \theta$) **then**
16:       accept recommendation of $\mathbf{r_i}$ on offerer $\mathbf{p_j}$
17:    **end if**
18: **end for**
19: compute s reputation from $\mathbf{r}$ set /* in a similar way to that of Steps 5–18 */
20: **for all** $\mathbf{p_j}$ **do**
21:    compute $RL_{q,\mathbf{p_j},\tau}$ (Eqn. 3) and $RL_{q,\mathbf{p_j},s,\tau}$ (Eqn. 6)
22:    **if** ($RL_{q,\mathbf{p_j},\tau} > \theta$) AND ($RL_{q,\mathbf{p_j},s,\tau} > \theta$) **then**
23:       $\mathbf{p_j}$ is candidate for downloading
24:    **end if**
25: **end for**
26: sort list of candidates
27: **repeat**
28:    select one of the top, named $\mathbf{p_w}$
29: **until** challenge-response handshake between q and $\mathbf{p_w}$ is successful
30: send $\mathbf{p_w}$ `DownloadReq(s)` to start download
31: append in *DPE* the associated experience /* for all $\mathbf{r_x},\mathbf{r}'_\mathbf{y}$ recommending $\mathbf{p_w}$ */
32: append in *DRE* experience for resource s

Algorithm 2 (*BRS*) - *Daemon*$_1$ run by the set of offering peers $\mathbf{p}$

1: **loop**
2:    receive `AskResource(`s,h`)` from an initiator q
3:    **if** s is included in resource repository **then**
4:       reply to q with `HoldResource(`s`)`
5:    **else if** within range of h **then**
6:       forward the `AskResource(`$s,h-1$`)` to neighbors
7:    **end if**
8: **end loop**

Algorithm 3 (*BRS*) - *Daemon*$_2$ run by set of recommending peers $\mathbf{r}$, $\mathbf{r}'$, and sites $\mathbf{p}$

loop
  receive `AskRecom(`$\mathbf{p_i}$`,`s`,`h`)`
  if records exist in *DPE* / *DRE* for $\mathbf{p_i}$ or s **then**
    reply to q with `PostRecom(`$\mathbf{p_i}$`,`s`)`
  else if within range of h **then**
    forward the `AskRecom(`$\mathbf{p_i}$`,`s`,`h`-1)` to neighbors
  end if
end loop

Algorithm 4 (*BRS*) - *Daemon*$_3$ run by $\mathbf{p_w}$

loop
  receive `DownloadReq(`s`)`
  send a copy of s to q
end loop

1) q broadcasts an `AskRequest` seeking s
2) $\mathbf{p_1}$ and $\mathbf{p_2}$ respond with `HoldResource`
3) q broadcasts `AskRecom` regarding offerers $\mathbf{p_1}$, $\mathbf{p_2}$ and s
4) recommenders reply with `PostRecom`; $\mathbf{r_1}$, $\mathbf{r_2}$ recommend $\mathbf{p_1}$, s; $\mathbf{r_3}$, $\mathbf{r_4}$ recommend $\mathbf{p_2}$, s
5) as $\mathbf{r_1}$, $\mathbf{r_2}$, $\mathbf{r_3}$ and $\mathbf{r_4}$ are unknown to q, the latter broadcasts `AskRecoms` on their behalf
6) second-line recommenders reply with `PostRecom`; $\mathbf{r'_1}$ for $\mathbf{r_1}$, $\mathbf{r'_2}$ for $\mathbf{r_2}$, $\mathbf{r'_3}$ for $\mathbf{r_3}$ and $\mathbf{r'_4}$ for $\mathbf{r_4}$ ($\mathbf{r'_i}$ and $\mathbf{r_i}$ for i=1..4 are all trustworthy here).
7) after evaluating $\mathbf{p_1}$, $\mathbf{p_2}$ and s, q challenge-response-handshakes with $\mathbf{p_2}$ and decides to download s from $\mathbf{p_2}$.
Finally, q appends its own new experience for $\mathbf{p_2}$, s, $\mathbf{r_3}$, $\mathbf{r_4}$, $\mathbf{r'_3}$ and $\mathbf{r'_4}$. Broadcasting is represented only with arrows to help figure clarity.

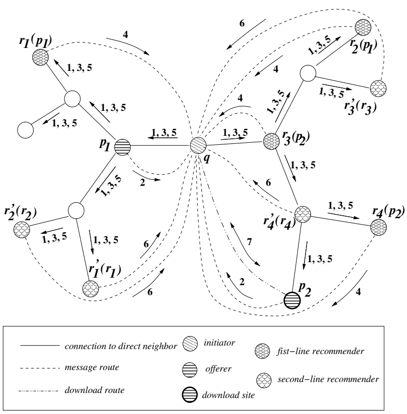

Figure 1: Sample session using *BRS*

4 The Reputation Monitoring Mechanism (*RMM*)

The operation of our proposed *RMM* scheme is based on *Reputation Variation Monitor (RVM)* sites that not only facilitate regular data querying but also undertake an augmented responsibility. The additional obligation of an *RVM* is to continually monitor the reputation levels of all peers that it supervises. When a site joins an overlay network, it contacts a trusted/known *bootstrap node* [15] which randomly selects a number of anonymous *RVM*s; these help monitor the site in question. The bootstrap node guarantees that designated *RVM*s in conjunction with the site entering the network do not produce any collusions [15]. It is expected that collaborating *RVM*s that oversee the same site are known among themselves. The identity of a *RVM* is never revealed to supervised peers. *RVM*s use multiple windows in the past, called *epochs*, to better gauge "prior" recorded site behavior in the network. We simplify *RMM*'s design by recording only the behavior of sites in previous epochs but not that of their individual resources. The ultimate selection of a downloading node is based on both the *BRS* recommendation (that yields "only-current" reputation levels for both sites and their resources) in conjunction with the evaluation of *RVM*s (that produce aggregations of prior behavior of sites by examining the set of λ most recent observed epochs). In doing so, the *RMM* algorithm attempts to limit abrupt changes of reputation levels that are often result of collusion [12, 4]. In light of suddenly popular resources, *RMM* will initially slow down steep changes in the reputation levels of peers; however, over time and in step-wise fashion, *RMM* will ultimately accommodate such "hot" resources. Last but not least, our *RMM* follows a third-party recommendation approach [16, 15] in contrast to previous proposals [6, 5, 18].

We enhance the *BRS* algorithm by substituting **Phase IV** as follows: the initiator q consults with the *RVM*s (via a broadcast) that were supervising the offering peers, concerning their reputation levels in the recent past. Once pertinent answers are received, q re-evaluates all offerers $\mathbf{p}$ which have qualified in the *BRS* test. The filtered list of peers $\mathbf{p}'$ is computed by considering both *BRS* recommendations as well as those of the overseeing *RVM*s. Last, q changes the line-up of its reputable peers in order to select a top one. Below, we discuss the required five steps that now constitute the **Phase IV** in the *RMM* algorithm.

- **Step IV(a) – Early Offerer Selection:** once all offerers $\mathbf{p}$ have been evaluated in terms of both peer and resource quality at the present time, the initiator q discards those offers that do not meet the expected reputation threshold θ for both sought resource s and furnishing peer (i.e., requires $(RL_{q,\mathbf{p_i},\tau} > \theta)$ AND $(RL_{q,\mathbf{p_i},s,\tau} > \theta)$). Qualifying offerers $\mathbf{p}'$ (a subset of $\mathbf{p}$) proceed to the next step.
- **Step IV(b) – Reputation Variation Request:** for each offerer i in the $\mathbf{p}'$ set, q dispatches an `AskRVM` message towards its associated *RVMs*. We employ *RVM* identity-hiding [7, 14] to prevent peers from starting *Denial-of-Service attacks* towards the associated *RVMs*. As q cannot be aware of the *RVM* identities[3] communication is established through a *mutual anonymity* communication protocol [7, 14, 19]. We employ anonymous communication between *RVM*s and initiators to prevent global eavesdroppers from identifying the nodes assigned to specific *RVM*s. As mentioned in Section 2, every *RVM* manages a *Reputation Table (RT)*

[3] Overseeing the i candidate peer in the $\mathbf{p}'$ set.

that stores the reputation levels for a subset of the potential offerers during the previous λ observed epochs. We assume that the duration of an epoch is application and/or user defined. At the end of each such period, a *RVM* determines the reputation of its own offerers using Eqn. (3) of the *BRS*. As soon as a *RVM* computes a new reputation value, it logically shifts all previous values one epoch back. The oldest of the λ values is discarded. Upon receipt of an `AskRVM` message, a *RVM* searches its *RT* structure to locate the λ appropriate reputation values. These are anonymously transported to q via a `RVMReply` message.

•**Step IV(c) – Evaluation of Offerer Reputation Variation:** as soon as q receives all pertinent `RVMReply` messages, it re-evaluates the reputation of all offerers in the set $\mathbf{p}'$. For each such offerer i, the initiator obtains a maximum of $\lambda * N_{RVM_i}$ reputation values; here, λ represents the number of epochs observed and N_{RVM_i} is the number of *RVMs* that coordinate the i-th element of $\mathbf{p}'$. Note that N_{RVM_i} may be different for each element in the set $\mathbf{p}'$. Since q is not aware of the real identity of *RVM* peers, it cannot evaluate their trustworthiness. On the other hand, the *RVMs* know their counterparts; thus, when they dispatch `RVM-Reply` messages during **Step IV(b)**, they may also include reputation-related experiences (i.e., Eqn. 3) for each other.

Let $R_{RVM_{x,y}}$ represent the reputation value that RVM_x perceives for RVM_y. AVR_{RVM_x} corresponds to the average reputation value that q computes for RVM_x after receiving reputation values that all other *RVM*s perceive for RVM_x at this point:

$$AVR_{RVM_x} = \frac{1}{N_{RVM_x} - 1} \sum_{y=1, y \neq x}^{N_{RVM_x}-1} R_{RVM_{x,y}} \tag{7}$$

The initiator q is now able to discard responses of untrustworthy *RVMs* (whose level is less than θ) and selects to trust n' qualifying peers, termed *RVM'*$[1..n']$. q takes into account only the λ reputation levels that each of the n' sites has dispatched. The initiator then computes the average *Reputation Level* concerning offerer $\mathbf{p'_i}$ for each epoch denoted $\overline{RL_{\mathbf{p'_i},t}}$ as follows:

$$\overline{RL_{\mathbf{p'_i},t}} = \frac{1}{n'} \sum_{k=1}^{n'} RL_{RVM'_k,\mathbf{p'_i},t} \qquad 1 \leq t \leq \lambda \tag{8}$$

For every offerer $\mathbf{p'_i}$ the $\lambda - 1$ *Relative Variation Values* at this time may be computed as:

$$V_{\mathbf{p'_i},c,t} = \frac{\overline{RL_{\mathbf{p'_i},t}} - \overline{RL_{\mathbf{p'_i},c}}}{1 - min\left\{\overline{RL_{\mathbf{p'_i},t}}, \overline{RL_{\mathbf{p'_i},c}}\right\}}, \qquad with\ 2 \leq t \leq \lambda\ \ and\ \ -1 \leq V_{\mathbf{p'_i},c,t} \leq 1 \tag{9}$$

Here, c represents the last epoch and t ranges over the $\lambda - 1$ prior time intervals. When the average *Reputation Level* of the offerer $\mathbf{p'_i}$ increases, $V_{\mathbf{p'_i},c,t}$ becomes negative triggering a reputation decrease. Correspondingly, when the average *Reputation Level* of $\mathbf{p'_i}$ signals an increase in reputation level value. When the current and one of the $\lambda-1$ *Reputation Level* values are one, the resulting zero variation points into an entirely reputable peer $\mathbf{p_i'}$ for the specific combination of epochs in consideration.

Note that the higher the difference is between the old and the new average *Reputation Level* of an offerer $\mathbf{p_i'}$ the higher the absolute value $|V_{\mathbf{p_i'},c,t}|$ is. *RMM* reprimands peers commensurately to their "stature" in the network. We deem a reputation that ranges in high values more significant than one that varies in low values. For example in Figure 2, *A*'s absolute $V_{\mathbf{p_i'},c,t}$ is higher than that of *B*. This is despite the fact that the change in the reputation of *B* is higher than that of *A*. As *A* enjoys higher reputation value, it is more likely to be designated as the selected resource provider, and thus, it must be reprimanded more strictly. Moreover, *B* is given a chance to behave well and improve its reputation. Due to the fact that *RMM* is not memoryless, it impacts the reputation levels of peers in a way proportional to λ as Figure 3 depicts. *RMM* smooths out changes and deliberately follows either ascending or descending trend in a slow fashion; this is useful to prevent pseudospoofing or shilling attacks but impedes nodes that suddenly become "hot". As *RVMs* cannot be strictly synchronized, their responses to the initiator q may reflect different time periods. This can be reduced by encouraging *RVMs* to gossip with their counterparts whenever idle.

•**Step IV(d) – Reputation Update:** the initiator q re-evaluates the *Reputation Level* of each trustworthy offerer $\mathbf{p_i'}$ as follows:

$$RL'_{q,\mathbf{p_i'},\tau} = RL_{q,\mathbf{p_i'},\tau} + \sum_{t=2}^{\lambda}(z_t * |\overline{RL_{\mathbf{p_i'},t}} - \overline{RL_{\mathbf{p_i'},c}}| * V_{\mathbf{p_i'},c,t}), \qquad \sum_{t=2}^{\lambda} z_t = 1 \tag{10}$$

where $RL_{q,\mathbf{p_i'},\tau}$ is the *Reputation Level* of $\mathbf{p_i'}$ at this time τ (Eqn. 3), $\overline{RL_{\mathbf{p_i'},t}} - \overline{RL_{\mathbf{p_i'},c}}$ represents the difference of $\mathbf{p_i'}$'s reputation level between the t^{th} epoch and the most recent recorded one c, $V_{\mathbf{p_i'},c,t}$ is the *Relative Variation Value* derived with Eqn. 9, and z_t are user-defined coefficients that are identical to all qualified offerers in $\mathbf{p'}$. The z coefficient values correspond to the weight that one elects to have when dealing with $\lambda-1$ epochs.

• **Step IV(e) – Final Offerer Selection:** the candidate peers in the $\mathbf{p_i'}$ are sorted according to their recomputed reputation level and the initiator q selects one of the top to reduce the probability for always overloading the top-performer. Before downloading, q carries out a challenge-response handshake to ensure no impersonation. Algorithms 5-7 show how *BRS* should be enhanced to produce our *Reputation Monitoring Mechanism*. Figure 4 depicts a sample session with *RMM*.

The final **Phase V** of *RMM* is identical to that of *BRS*. Although the initiator's opinion of the *RVM*s behavior would in principle be desirable, it is impossible due to their anonymity.

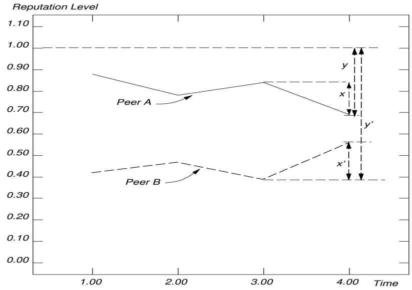

Figure 2: The fraction x/y represents intuitively the *Relative Variation Value* during the last *epoch*. The absolute *Relative Variation Value* of *A* (0.48) is higher than that of *B* (0.39), even though the actual change is 0.15 and 0.17 respectively, because *A* ranges in higher reputation levels.

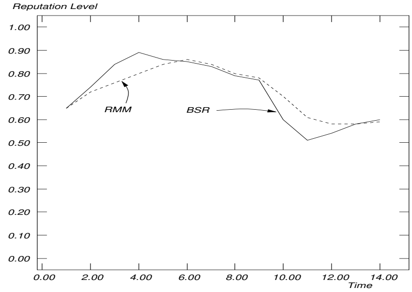

Figure 3: The *Reputation Level* computed by a peer, with no previous experience on the resource holder, in both reputation schemes. It is shown that *RMM* smooths out any attempt to artificially increase or decrease the *Reputation Level* of the resource holder.

1) *q* broadcasts an `AskRequest` seeking *s*
2) $\mathbf{p_1}$ and $\mathbf{p_2}$ respond with `HoldResource`
3) *q* broadcasts `AskRecom` regarding offerers $\mathbf{p_1}$, $\mathbf{p_2}$ and *s*
4) recommenders reply with `PostRecom`; $\mathbf{r_1}$, $\mathbf{r_2}$ recommend $\mathbf{p_1}, s$; $\mathbf{r_3}$, $\mathbf{r_4}$ recommend $\mathbf{p_2}, s$
5) as $\mathbf{r_1}$, $\mathbf{r_2}$, $\mathbf{r_3}$ and $\mathbf{r_4}$ are unknown to *q*, the latter broadcasts `AskRecoms` on their behalf
6) second-line recommenders reply with `PostRecom`; $\mathbf{r'_1}$ for $\mathbf{r_1}$, $\mathbf{r'_2}$ for $\mathbf{r_2}$, $\mathbf{r'_3}$ for $\mathbf{r_3}$ and $\mathbf{r'_4}$ for $\mathbf{r_4}$ ($\mathbf{r'_i}$ and $\mathbf{r_i}$ for i=1..4 are all trustworthy here).
7) *q* sends an anonymous `AskRVM` asking for reputation variations on $\mathbf{p_1}$ and $\mathbf{p_2}$
8) RVM_1, RVM_2 and RVM_3, RVM_4 report to *q* (via a `RVMReply`) regarding $\mathbf{p_1}$ and $\mathbf{p_2}$
9) after re-evaluating $\mathbf{p_1}$, $\mathbf{p_2}$ and *s*, *q* challenge-response-handshakes with $\mathbf{p_1}$ and decides to download *s* from $\mathbf{p_1}$.
Finally, *q* appends to *DPE/DRE* its experience on $\mathbf{p_1}$, *s*, $\mathbf{r_1}$, $\mathbf{r_2}$, $\mathbf{r'_1}$ and $\mathbf{r'_2}$.

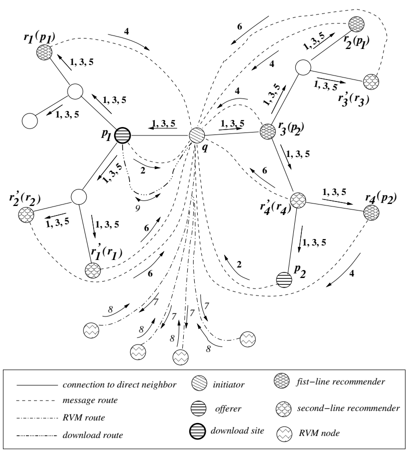

Figure 4: Our reputation scheme after adding *RMM* functionality

Algorithm 5 *RMM* Scheme - Initiator's Algorithm

1: /* The following segment is inserted after line 25 of Algorithm 1 */
2: **for all** $\mathbf{p_i'}$ **do**
3:    send anonymous `AskRVM(`$\mathbf{p_i'}$`)` to supervising *RVM*s
4:    receive `RVMReply(`$\mathbf{p_i'}$`)`
5: **end for**
6: **for all** RVM_k of $\mathbf{p_i'}$ **do**
7:    compute the reputation of RVM_k (Eqn. 7)
8:    discard responses from untrustworthy *RVM*s
9: **end for**
10: select the n' most reputable *RVM*s called RVM'
11: **for all** $\mathbf{p_i'}$ **do**
12:    compute the $\lambda - 1$ relative variation values (Eqn. 9)
13:    compute the new reputation value of $\mathbf{p_i'}$ (Eqn. 10)
14: **end for**

Algorithm 6 *RMM* Scheme - $Daemon_4$ run by all *RVM*s

1: **loop**
2:    receive `AskRVM(`$\mathbf{p_i'}$`)`
3:    **if** *RT* contains records for the reputation values of $\mathbf{p_i'}$ **then**
4:       send anonymously to q a `RVMReply(`$\mathbf{p_i'}$`)` returning λ variation values along with the reputation values of the other *RVM*s as perceived at this site.
5:    **end if**
6: **end loop**

Algorithm 7 *RMM* Scheme - Update *RT* at *RVM*s

1: /* q_{RVM} is the *RVM* that acts as query initiator */
2: **loop**
3:    wait until epoch finishes
4:    **for all** $\mathbf{p_i}$ supervised by the site **do**
5:       /* with values appearing in *RT* */
6:       compute $RL_{q_{RVM},\mathbf{p_i},\tau}$
7:       shift λ reputation values 1 epoch back /* make space for the result of line 6 */
8:       append $RL_{q_{RVM},\mathbf{p_i},\tau}$ into *RT* structure /* just vacated position in line 7 */
9:    **end for**
10: **end loop**

5 Discussion of *RMM* Main Features

Attack Resistance: Our approach is resistant to attacks targeting reputation-based systems. In particular, it prevents *pseudospoofing* since the query initiator has a challenge-response handshake with the selected offerer. If the offerer is a virtual node it will not be able to verify its existence and consequently, the fraud is revealed. The handshake measure could be also employed in the recommendation process of both *BRS* and *RMM*; however it is not deemed necessary as the recommenders are evaluated by third parties.

Concerning *shilling*, no technique may prevent the formation of groups that jointly try to alter reputation values of peers. The introduction of *Reputation Monitoring Mechanism* ensures that even if peers collaborate, the reputation scheme can resist abrupt changes in the reputation values of nodes. *RMM* smooths out the reputation variations and prevents peers from easily altering their own reputation level. In order to attain better overall behavior, we anticipate that peers have to bind with permanent identities that cannot be easily modified; this manner, we avoid having sites that unilaterally reset their own reputation level to a default value.

A general-purpose security issue that affects any reputation system is *message transmission* [4]. A peer might reply honestly when asked about its experience on another peer, but a *man-in-the-middle* may mediate the communication between initiator and recommender and tamper with the recommender's reply message. Public Key Infrastructure (PKI) [1] can be used to ensure integrity and verify message origination. There is no reason to impose confidentiality in message exchange as a peer's reputation is not secret. Moreover, in order to protect peers that act as *RVMs*, we impose *anonymity* constraints. If a malicious peer were allowed to identify its overseeing *RVMs* it could potentially initiate a *Denial-of-Service* attack in order to prevent distribution of prior reputation values.

Another possible attack is the impersonation of a *RVM*. Since *RVMs* are anonymous, the initiator cannot directly communicate with them to verify identity. To overcome this issue, the bootstrap peer creates a private-public key pair $\langle C, D \rangle$ that is distributed only to the *RVMs* [15]. The assigned offerer $\mathbf{p_i}$ becomes aware only of the public key D. This private-public key pair is different from the pairs that the *RVMs* and $\mathbf{p_i}$ use as individual peers. When $\mathbf{p_i}$ replies to the initiator with a `HoldResource` message, it includes the public key D. Also, when *RVMs* are requested the reputation variation values of $\mathbf{p_i}$, they encrypt their response with their private key C and the initiator decrypt it with the public key D. In this way, no peer is capable of impersonating a *RVM* unless it is aware of C.

RVM Failure: As *RVM* peers remain part of the *P2P* network, they always reserve the right to disconnect whenever they wish. If such a peer disconnects, it may designate a new site at random (as if the *RVM* site were the *bootstrap* node) in order to take over the *RVM* responsibilities. A hardware failure, a networking problem, or even an attack may force a *RVM* node to go offline without notice. Its absence will be detected by its *RVM* counterparts and subsequently, following an assignment policy (such as random), another *RVM* node may take over.

Depending on the case, either the disconnecting peer or any of its counterparts is responsible to inform the new *RVM* site about the previous reputation values of their supervised peer. In addition, the new *RVM* gets informed of the identities of the rest *RVMs* that are responsible for the same peer. Subsequently, the new *RVM* notices the rest *RVMs* that it has replaced the departed peer.

Number of Epochs and Duration: The designation of the λ number of *epochs* and the *epoch* duration require calibration in a specific application environment. In networks with high traffic, we should avoid using many and small period epochs, as this would impose a significant load on the *RVMs*.

There is a dependence between the epoch duration and the average frequency f_x at which the most popular peers receive download requests. We may examine requests on popular peers as they are expected to have the most rapid reputation variations. An epoch should be sufficiently short to prevent loosing peaks in the reputation curve. The network population N may also determine the number of epochs observed (i.e., λ) and their duration. We anticipate that small networks are less prone to rapid reputation variations and therefore we may employ few *epochs* with long duration to effectively capture the active history of nodes. In contrast, in larger networks where the likelihood of conspiracy is greater, we expect to observe more frequent reputation variations; thus, the number of observed *epochs* should be increased while their duration should be decreased.

Space Overhead and Communication Cost: In terms of space overhead, each peer q running *BRS* maintains two repositories *DPE* and *DRE*. Site q may adopt any replacement policy (i.e. *LRU*) in order to limit the recorded experiences to a manageable number of entries. The *RMM* algorithm imposes some additional space requirements for *RVM*. Each *RVM* site stores $\lambda * N_p$ reputation records in its *RT* structure where N_p is the average number of peers assigned to an individual *RVM*. Last, every *RVM* node has to be aware of the aggregate reputation levels of all its counterparts at a time.

BRS utilizes broadcasting to seek resources as well as recommendations for peers and resources. As the broadcasting travels maximum h legs, we assume $h/2$ average hop propagation for a message. Should each peer talk to average k peers, then the average number of messages handled is $k^{h/2}$. If each peer forwards queries to its most reputable neighbors k_r, it is equiprobable that it will reach also reputable resource holders. In this case, the number of messages decreases as $k_r << k$. The fact that a site may occasionally receive the same requesting message twice (i.e., the message has to be discarded), helps to further reduce the above average estimation. If $k^{h/2}$ becomes too large, a *P2P DHT* protocol (such as *Chord* [17]) may reduce the number of messages needed to locate a resource to approximately $O(logN)$ with N being the number of nodes. The use of such a routing protocol would lack the resource holder selection through the most reputable neighbors.

The *RMM algorithm* requires an extra cost for the anonymous communication between an initiator and the corresponding *RVMs*. If *Tarzan*'s mechanism [7] is deployed, a cost proportional to the network population $O(N)$ incurs.

Initial reputation selection: According to [10], the best initial reputation value is 0.3 for reputation schemes where reputation values range in $[0,1]$ and where peers maintain their identities. They select a value that is lower than the middle (0.5)

because in most schemes it is not very difficult to raise someone's reputation. In our approach though, we have no reason to set the initial reputation to anything but 0.5, as it is more difficult for an already reputable peer to raise its reputation.

6 Related Work

In the context of *BRS*, [2] proposed a trust management scheme based on complaints instead of the actual experience of the requestors. A possible disadvantage of this approach is that a peer with no history may be considered as trustworthy as it lacks complaints. In *TrustMe* [15], *RCert* [11], and *P2Prep* [5], there is no differentiation between peer behavior and resource quality. *XRep* [6], a follow-up of the *P2Prep* system, maintains two types of structures that store behavior for both peers and resources. [3] offers a classification of trust factors and introduces the *context-specific* voting. In [18], multidimensional trust is exploited and *Bayesian networks* are used to express peer/resource satisfaction. *SHARP* proposes the binding of resources with *leases* that expire within finite periods of time to attain secure peering [8]. Our work builds upon these earlier efforts and proposes the *RMM* mechanism that continually monitors the reputation of nodes in addition to that of resources. The notion of *epoch* is used to quantify the quality perceived by nodes regarding the trustworthiness of resource contributing peers. In our approach, an initiator selects the most reputable offerer by utilizing the encountered reputation values over the last λ observed epochs.

7 Conclusions

The deployment of reputation schemes in overlay networks is necessitated by the fact that requesting peers may not be certain of the intention of resource offerers. In this paper, we propose the *Reputation Monitoring Mechanism* that protects peers from being deceived by malicious nodes. This is also the case even if faulty nodes attempt to control other legitimate peers and conceal their misbehavior. We suggest that each offerer peer is assigned to a set of *Reputation Variation Monitor (RVM)* sites. *RVM*'s main goal is to monitor reputation changes of resource offering peers during a number of epochs. Hence, any resource requestor should consult with the supervising *RVMs* of any offerer to ascertain level of confidence. *RMM* detects abnormal peer behavior by monitoring abrupt reputation changes and so prevents the establishment of false reputation. *RMM* operates as an advisory mechanism that helps a user make the best possible decision in reference to the reputation of a resource-holder. *RMM* can be also employed to enhance security in *P2P* networks as it is able to cooperatively function with any underline protocol and reputation scheme.

References

[1] Public key infrastructure. http://www.pki-page.org/.

[2] K. Aberer and Z. Despotovic. Managing Trust in a Peer-to-Peer Information System. In *Proc. of 10th International Conference on Information and Knowledge Management*, Atlanta, GA, November 2001.

[3] F. Azzedin and M. Maheswaran. Trust Modeling for Peer-to-Peer Based Computing Systems. In *Proc. of 3rd IEEE International Conference on Peer-to-Peer Computing*, Linkoping, Sweeden, September 2003.

[4] M. Bishop. *Computer Security: Art and Science*. Addison Wesley Professional, Boston, MA, 2003.

[5] F. Cornelli, E. Damiani, S. Vimercati, S. Paraboschi, and P. Samarati. Implementing a Reputation-Aware Gnutella Servent. In *Proc. of 1st International Workshop on Peer-to-Peer Computing*, Pisa, Italy, May 2002.

[6] E. Damiani, D. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. In *Proc. of 9th ACM Conference on Computer and Communications Security*, Washington, DC, USA, November 2002.

[7] M. Freedman and R. Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *Proc. of 9th ACM Conference on Computer and Communications Security*, Washington, DC, USA, November 2002.

[8] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. SHARP: An Architecture for Secure Resource Peering. In *19th ACM Symposium on Operating Systems Principles (SOSP'03)*, Lake George, NY, October 2003.

[9] P. Maniatis, M. Roussopoulos, TJ Giuli, D.S.H. Rosenthal, M. Baker, and Y. Muliadi. Preserving Peer Replicas By Rate-Limited Sampled Voting. In *19th ACM Symposium on Operating Systems Principles (SOSP'03)*, Lake George, NY, October 2003.

[10] S. Marti and H. Garcia-Molina. Identity Crisis: Anonymity vs. Reputation in P2P Systems. In *Proc. of 3rd International Conference on Peer-to-Peer Computing*, Linkoping, Sweeden, September 2003.

[11] B.C. Ooi, C.Y. Liau, and K.-L. Tau. Managing Trust in Peer to Peer Systems Using Reputation-based Techniques. In *Proc. of 4th Int. Conf. on Web-Age Information Management*, Chengdu, China, August 2003.

[12] A. Oram. *Peer-to-Peer Harnessing the power of Disruptive Technologies*. O'Reilly & Associates, New York, NY, USA, 2001.

[13] M.K. Ramanathan, V. Kalogeraki, and J. Pruyne. Finding Good Peers in Peer-to-Peer Networks. In *IEEE International Parallel and Distributed Computing Symposium (IPDPS'02)*, Fort Lauderdale, FL, April 2002.

[14] V. Scarlata, B. Levine, and C. Shields. Responder Anonymity and Anonymous Peer-to-Peer File Sharing. In *Proc. of 9th IEEE International Conference on Network Protocols*, Riverside, CA, November 2001.

[15] A. Singh and L. Liu. TrustME: Anonymous Management of Trust Relationships in Decentralized P2P Systems. In *Proc. of 3rd IEEE Int. Conf. on Peer-to-Peer Computing*, Linkoping, Sweeden, September 2003.

[16] J.G. Steiner, C. Neuman, and J.I. Schiller. Kerberos: An Authentication Service for Open Network Systems. In *Proc. of the 1988 USENIX Conference*, Dallas, TX, February 1988.

[17] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proc. of ACM SIGCOMM Conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, UC San Diego, CA, USA, August 2001.

[18] Y. Wang and J. Vassileva. Trust and Reputation Model in Peer-to-Peer Networks. In *Proc. of 3rd International Conference on Peer-to-Peer Computing*, Linkoping, Sweeden, September 2003.

[19] L. Xiao, Z. Xu, and X. Zhang. Mutual Anonymity Protocols for Hybrid Peer-to-Peer Systems. In *Proc. of 23rd IEEE Inter. Conf. on Distributed Computing Systems, (ICDCS'03)*, Providence, RI, May 2003.

Theodora Dariotaki received both her BS and MS degrees in Computer Science from The University of Athens in 2001 and 2003 respectively and is now a research associate. Her inetrests are in peer-to-peer networks and systems. She can be reached at th.dariotaki@di.uoa.gr

Alex Delis is with the faculty of Informatics & Telecommunications at The University of Athens. His research interests are in Distributed Systems, Networked Databases, and System Evaluation. He can be reached at ad@di.uoa.gr