*This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.*

# MARCH: A Distributed Incentive Scheme for Peer-to-Peer Networks

Zhan Zhang Shigang Chen Myungkeun Yoon
Department of Computer & Information Science & Engineering, University of Florida
{zzhan, sgchen, myoon}@cise.ufl.edu

***Abstract*—As peer-to-peer networks grow larger and include more diverse users, the lack of incentive to encourage cooperative behavior becomes one of the key problems. This challenge cannot be fully met by traditional incentive schemes, which suffer from various attacks based on false reports. Especially, due to the lack of central authorities in typical P2P systems, it is difficult to detect colluding groups. Members in the same colluding group can cooperate to manipulate their history information, and the damaging power increases dramatically with the group size. In this paper, we propose a new distributed incentive scheme, in which the benefit that a node can obtain from the system is proportional to its contribution to the system, and a colluding group cannot gain advantage by cooperation regardless of its size. Consequently, the damaging power of colluding groups is strictly limited. The proposed scheme includes three major components: a distributed authority infrastructure, a key sharing protocol, and a contract verification protocol.**

## I. Introduction

Today's P2P networks are designed based on the altruism of participating nodes: users are assumed to share their resources in exchange for the right to consume the resources of others. As networks grow larger and include more diverse users, it has been observed that many people consume resources without contributing to the community, resulting in the "free riding" problem. Consequently, as altruism breaks down, individual players' self interest may cause the systems to collapse.

To reduce free-riders, the systems have to incorporate incentive schemes to encourage cooperative behavior. Some recent works [1], [2], [3], [4], [5], [6] propose *reputation* based trust systems, in which each node is associated with a reputation established based on the feedbacks from others that it has made transactions with. The reputation information helps users to identify and avoid malicious nodes. An alternative is *virtual currency* schemes [7], [8], [9], in which each node is associated with a certain amount of money. Money is deducted from the consumers of a service, and transferred to the providers of the service after each transaction.

Both types of schemes rely on authentic measurement of service quality and unforgeable reputation/money information. Otherwise, selfish/malicious nodes may gain advantage based on false reports. For example, a consumer may falsely claim to have not received service in order to pay less or defame others. More seriously, malicious nodes may collude in cheating in order to manipulate their information. Several algorithms are proposed to address these problems. They either analyze statistical characteristics of the nodes' behavior patterns and other nodes' feedbacks [2], [10], or remove the underlying incentive for cheating [11]. However, in order to apply these algorithms, the nodes' history information must be managed by a central authority, which is not available in typical peer-to-peer networks.

Other works [12], [13] propose to find chains of trust based on the history information shared among trusted nodes. However, the communication overhead for discovering chains of trust is very high, making these schemes not scalable.

In this paper, we focus on how to design an effective incentive scheme suitable for P2P systems, which have no central authority to maintain individual nodes' history information. The major contributions are listed below.

(1) We propose a new distributed incentive scheme, which combines reputation and virtual money. It is able to strictly limit the damage caused by malicious nodes and their colluding groups. The following features distinguish our scheme from others.

- *The benefit that a node can get from the system is limited by its contribution to the system.*
- *The members in a colluding group cannot increase their total money or aggregate reputation by cooperation, regardless of the group size.*
- *Malicious nodes can only attack others at the cost of their own interest.*

(2) We design a distributed authority infrastructure to manage the nodes' history information with low overhead and high security.

(3) We design a key sharing protocol and a contract verification protocol based on the threshold cryptography to implement the proposed distributed incentive scheme.

The rest of the paper is organized as follows. Section II provides the motivation for our work. Section III defines the system model. Section IV proposes a distributed authority infrastructure. Section V presents our distributed incentive scheme. Section VI studies the properties of the proposed scheme. Section VII discusses several important issues. Section VIII evaluates the scheme by simulations. Section IX draws the conclusion.

## II. Motivation

### A. Limitation of Prior Work

Any node in a peer-to-peer network is both a service provider and a service consumer. It contributes to the system by working as a provider, and benefits from the system as a consumer. A transaction is the process of a provider offering a service to a consumer, such as supplying a video file.

**0743-166X/07/$25.00 ©2007 IEEE**

The purpose of an incentive scheme is to encourage the nodes to take the role of providers. However, without central authorities involved, the existing schemes cannot effectively prevent malicious nodes, especially those in collusion, from manipulating their history information by using false service reports. Specifically, they have the following problems.

*Reputation inflation:* In the reputation schemes, malicious nodes can work together to inflate each other's reputation or to defame innocent nodes, by which colluding nodes protect themselves from the complaints by innocent nodes as these complaints may be treated as noise by the systems.

*Money depletion:* In the virtual currency schemes, malicious nodes may launch attacks to deplete other nodes' money and paralyze the whole system. Without authentic reputation information, innocent nodes are not able to proactively select benign partners and avoid malicious ones.

*Frequent complainer:* In many incentive schemes, nodes will be punished if they complain frequently, which prevents malicious nodes from constantly defaming others at no cost. However, it also discourages innocent nodes from reporting frequent malicious acts because otherwise they would become frequent complainers.

*Punishment scale:* In most existing schemes, the scale of punishment is related to the service history of the transaction participants. Consequently, an innocent node may be subject to negative discrimination attacks [1] launched by nodes with excellent history.

### B. Motivation

There are two major kinds of bad behavior. First, a provider may *deceive* a consumer by providing less-than-promised service. Second, a consumer may *defame* a provider by falsely claiming the service is poor.

Consider how these problems are dealt with in real life. Before a transaction happens, the provider would want to know if the consumer has enough money to pay for the service, and the consumer would want to know the reputation of the provider. With such information, they can control the risk and decide whether to carry out the transaction or not. After the transaction, if the provider deceives, it will be sued by the consumer. Consequently, the malicious provider will build up bad reputation, which prevents it from deceiving more consumers. Now consider a consumer intentionally defames a provider. It does so only after it can show the evidence of a transaction, which requires it to pay money first. Consequently, defaming comes with a cost. The ability of the malicious consumer to defame others is limited by the amount of money it has.

Inspired by the observation above, we propose a new incentive scheme: ***MARCH***, which is a combination of Money And Reputation sCHemes.

The basic idea behind the scheme is simple: each node is associated with two parameters: money and reputation. The providers earn money (and also reputation) by serving others. The consumers pay money for the service. If a consumer does not think the received service worth the money it has paid, it reports to an authority, specifying the amount of money it believes it has overpaid. If the authority can determine who is lying, the liar is punished. Otherwise, the authority freezes the money claimed to have been overpaid. The money will not be available to the provider and will not be returned to the consumer either, which eliminates any reason for the consumer to lie. If the provider is guilty, the consumer has the revenge and the provider's reputation suffers. If the provider is innocent, the consumer does it at a cost because after all it has paid the price of the transaction. In addition, the falsely-penalized provider will not serve it any more. The technical challenges are (1) how to establish a distributed authority for managing the money and reputation, (2) how to design the protocol of transaction that ensures authentic exchange of money/reputation information and allows the unsatisfied consumers to sue the providers, (3) how to analyze the properties of such a system, and (4) how to evaluate the system.

## III. System Model

The nodes in a P2P network fall in three categories: honest, selfish, and malicious. *Honest* nodes follow the protocol exactly, and they both provide and receive services. *Selfish* nodes will break the protocol only if they can benefit. *Malicious* nodes are willing to compromise the system by breaking the protocol even when they benefit nothing and may be punished. Selfish/malicious nodes may form colluding groups. There may exist a significant number of selfish nodes, but the malicious nodes are likely to account for a relatively small percentage of the whole network. At a certain time, all self/malcious nodes that break the protocol are called *dishonest* nodes. A node is said to be *rejected* from the system if it has too little money and too poor reputation such that no honest providers/consumers will perform transaction with it. We study the incentive scheme in the context of DHT-based P2P networks, e.g., [14], [15], [16]. We assume the routing protocol is robust, ensuring the reliable delivery of messages in the network [17]. We also assume the networks have the following properties.

• *Random, non-selectable identifier:* A node can not select its identifier, which should be arbitrarily assigned by the system. This requirement is essential to defending the Sybil attack [18]. One common approach is to hash a node's IP address to derive a random identifier for the node [14].

• *Public/private key pair:* Each node $A$ in the network has a public/private key pair, denoted as $P_A$ and $S_A$ respectively. A trusted third party is needed to issue public-key certificates. The trusted third party is used off-line once per node for certificate issuance, and it is not involved in any transaction.

## IV. Authority Infrastructure

### A. Delegation

Who will keep track of the money/reputation information in a P2P network? In the absence of a central authority for this task, we design a distributed authority infrastructure. Each node $A$ is assigned a **delegation**, denoted as $D_A$, which consists of $k$ nodes picked pseudo-randomly. For example, we can apply $k$ hash functions, i.e., $\{h_1, h_2, ..., h_k\}$, on the

identifier of node $A$ to derive the identifiers of nodes in $D_A$. If a derived identifier does not belong to any node currently in the network, the "closest" node is selected. For example, in [14], it will be the node clockwise after the derived identifier on the ring. The $j$-th element in $D_A$ is denoted as $D_A(j)$.

$D_A$ keeps track of $A$'s money/reputation. Any anomaly in the information stored at the delegation members may indicate an attempt to forge data. The information is legitimate only if the majority of the delegation members agree on it. Therefore, as long as the majority of the delegation members are honest, the information about node $A$ cannot be forged. Such a delegation is said to be *trustworthy*. On the other hand, if at least half of the members are dishonest, then the delegation is *untrustworthy*.

The delegation members are appointed pseudo-randomly by the system. A node cannot select its delegation members, but can easily determine who are the members in its or any other node's delegation. To compromise a delegation, the malicious/selfish nodes from a colluding group must constitute the majority of the delegation. Unless the colluding group is very large, the probability for this to happen is small because the identifiers of the colluding nodes are randomly assigned by the system and the identifiers of the delegation are also randomly assigned. Let $m$ be the size of a colluding group and $n$ be the total number of nodes in the system. The probability for $t$ out of $k$ nodes to be in the colluding group is

$$P(t,k,\frac{m}{n}) = \left( \begin{array}{c} k \\ t \end{array} \right) (\frac{m}{n})^t (1-\frac{m}{n})^{k-t}$$

where $P(t,k,\frac{m}{n})$ denotes the probability of $t$ successes in $k$ trials in a Binomial distribution with the probability of success in any trial being $\frac{m}{n}$. Let $m^*$ be the total number of distinct nodes in all colluding groups, also including all malicious nodes. The probability of a delegation being trustworthy is at least $\sum_{t=0}^{\lfloor \frac{k}{2} \rfloor} P(t,k,\frac{m^*}{n})$. In order to control the overhead, we shall keep the value of $k$ small.

### B. $k$-pair Trustworthy Set

A transaction involves two delegations, one for the provider and the other for the consumer. They have to cooperate in maintaining the money and reputation information, and avoiding any fraud. To facilitate the cooperation, we introduce a new structure, called **$k$-pair delegation set**, consisting of $k$ pairs of delegation members. Suppose node $A$ is the provider and node $B$ is the consumer. The $i$th pair is $(D_A(i), D_B(i))$, $\forall i \in [1..k]$, and the whole set is

$$\{(D_A(1), D_B(1)), (D_A(2), D_B(2)), ..., (D_A(k), D_B(k))\}$$

If both $D_A(i)$ and $D_B(i)$ are honest, the pair $(D_A(i), D_B(i))$ is trustworthy. If the majority of the $k$ pairs are trustworthy, the whole set is trustworthy. It can be easily verified that the probability for the whole set to be trustworthy is

$$\sum_{t=0}^{\lfloor \frac{k}{2} \rfloor} P(t,k,2\frac{m^*}{n} - (\frac{m^*}{n})^2)$$

As an example, when $m^* = 3,000$, the trustworthy probabilities for a delegation and a 5-pair delegation set are 99.975% and 99.815%, respectively. Even if a delegation (or $k$-pair delegation set) is not trustworthy, we say it is *compromised* only when the majority of the members happens to belong to the same colluding group, which is very unlikely due to the random selection of delegation members.

## V. MARCH: A Distributed Incentive Scheme

### A. Money and Reputation

With the distributed authority designed in the previous section, the following information about a node $A$ is maintained by a delegation of $k$ nodes.

**Total money** ($TM_A$): It is the total amount of money paid *by* others to node $A$ minus the total amount of money paid *to* others by $A$ in all previous transactions. The universal refilled money (Section VII-B) will also be added to this variable.

**Overpaid money** ($OM_A$): It is the total amount of money overpaid by consumers. A consumer pays money to node $A$ before a service. If the service contract is not fulfilled by the transaction, the consumer may file a complaint, specifying the amount of money that it has overpaid. This amount cannot be greater than what the consumer has paid.

When a new node joins the network, its total money and overpaid money are initialized to zero. From $TM_A$ and $OM_A$, we define the following two quantities.

**Available money** ($m_A$): It is the amount of money that node $A$ can use to buy services from others.

$$m_A = TM_A - OM_A \tag{1}$$

**Reputation** ($r_A$): It evaluates the quality of service (with respect to the service contracts) that node $A$ has provided.

$$r_A = \begin{cases} \frac{TM_A - OM_A}{TM_A} & \text{if } TM_A \neq 0 \\ 1 & \text{if } TM_A = 0 \end{cases} \tag{2}$$

To track every node's available money and reputation, we propose a set of protocols. Consider a transaction, in which Alice ($A$) is the provider and Bob ($B$) is the consumer. The transaction consists of five sequential phases.

- *Phase one: Contract Negotiation.* Alice and Bob negotiate a service contract.
- *Phase two: Contract Verification.* Through the help of their delegations, Alice and Bob verify the authenticity of the information claimed in the contract.
- *Phase three: Money Transfer.* The amount of money specified in the contract is transferred from Bob's account in $D_B$ to Alice's account in $D_A$.
- *Phase four: Contract Execution.* Alice offers the service to Bob based on the contract specification.
- *Phase five: Prosecution.* After the service, Bob provides feedback reflecting the quality of service offered by Alice.

### B. Phase one: Contract Negotiation

Suppose Bob has received a list of providers through the lookup routine of the P2P network. Each provider specifies its reputation and its price for the service. Bob wants to minimize his risk when deciding which service provider he is going to use.

Let $L$ be the price specified by Alice and $G$ be the fair price estimated by Bob himself. According to the definition, $r_A$ can roughly be used as a lower bound on the probability of Alice being honest. Intuitively, the probability for Bob to receive the service $G_B$ is at least $r_A$, and the probability for Bob to waste its money $L_A$ is at most $(1-r_A)$. We define the benefit for Bob to have a transaction with Alice as $G_B \times r_A - L_A \times (1-r_A)$. We further normalize it as

$$R = r_A - \frac{L}{G}(1 - r_A) \tag{3}$$

To avoid excessive risk, Bob takes Alice as a potential provider if $R$ is greater than a threshold value $T$. The use of threshold helps the system reject dishonest providers with poor reputation. Among all potential providers, Bob picks the one with the highest normalized benefit.

Both the value of $L$ and the value of $r_A$ are given by the provider $A$. If $L$ is set too high, $R$ will be small and the provider runs the risk of not being picked by Bob. Providers with poor reputation can improve their $R$ values by setting their prices low. In this way, they can recover their reputation by selling services at lower prices. If Alice lies about its $r_A$, she will be caught in the next phase and be punished.

Now suppose Bob chooses Alice as the best service provider. They have to negotiate a service contract, denoted as $c$, in the following format.

$$< A, B, S, Q, L, Seq_A, Seq_B, r_A, m_B >$$

where $A$, $B$, $S$, $Q$, and $L$ specify the provider, the consumer, the service type, the service quality, and the service price respectively. $Seq_A$ and $Seq_B$ are the contract sequence numbers of Alice and Bob, respectively. After the transaction, Alice and Bob each increase their sequence numbers by 1. The values of $r_A$ and $m_B$ in the contract will be verified by the delegations in the next phase.

### C. Phase two: Contract Verification

After negotiating a contract, Alice and Bob should exchange an authenticatable contract proof, so that Alice is able to activate the money transfer procedure and Bob is granted the prosecution rights. In addition, the information in the contract, e.g., $r_A$ and $m_B$, should be verified by the delegations of Alice and Bob.

We use the notation $[x]_y$ for the digital signature signed on message $x$ with key $y$ and $\{x\}_y$ for the cipher text of message $x$ encrypted with key $y$. After Phase two, if the contract is verified by the delegations, Alice should have the following contract proof

$$c_A = [c]_{S_B}$$

$c_A$ should not be produced by Bob, who may lie about $m_B$. Instead, Alice must receive $c_A$ from Bob's delegation after the members confirm the value of $m_B$. Bob has $k$ delegation members. Each of them will produce a "piece" of $c_A$ and send it to Alice, who will combine the "pieces" into a valid contract proof. Similarly, Bob must receive the following contract proof from Alice's delegation

$$c_B = [c]_{S_A}$$

The contract proofs will be used by Alice for money transfer and by Bob for prosecution.

It is important to ensure that either both Alice and Bob, or none of them, receive the contract proofs. Otherwise, dishonest nodes may take advantage of it. It can be shown that ensuring both or neither one receives her/his contract proof is impossible without using a third party (the delegation of Alice or Bob in this case).

#### *Key Sharing Protocol*

A $k$-member delegation is not a *centralized* third party. One possible approach for producing a contract proof by a delegation is to use threshold cryptography [19]. A $(k, t)$ threshold cryptography scheme allows $k$ members to share the responsibility of performing a cryptographic operation, so that any subgroup of $t$ members can perform this operation successfully, whereas any subgroup of less than $t$ members can not. For digital signature, $k$ shares of the private key are distributed to the $k$ members. Each member generates a partial signature by using its share of the key. After a combiner receives at least $t$ partial signatures, it is able to compute the signature, which is verifiable by the public key.

In our case, the problem is to produce $c_B$ (or $c_A$) by the $k$-member delegation of Alice (or Bob). We employ a $(k, \lfloor \frac{k}{2} \rfloor + 1)$ threshold cryptography scheme to produce the contract proof. Alice distributes shares $S_A(i)$ of her private key $S_A$ to her delegation members $D_A(i)$, which will produce partial signatures $[c]_{S_A(i)}$ and forward them to Bob for combination.

When applying threshold cryptography, we have to defend against dishonest nodes, which may intentionally distribute incorrect secret shares. The incorrect partial signatures cannot yield a valid signatures. We propose a protocol for distributing the key shares. Take Alice as an example. The protocol guarantees that either all delegation members receive the correct shares, or they all detect that Alice is dishonest.

**Step 1:** Alice sends a key share $S_A(i)$ to each delegation member $D_A(i)$, encrypted by the member's public key $P_{D_A(i)}$. The messages are shown below.

$$\text{MSG1} \quad Alice \rightarrow D_A(i): \quad [\{S_A(i)\}_{P_{D_A(i)}}]_{S_A}, \forall D_A(i) \in D_A$$

**Step 2:** After all members receive their key shares, they negotiate a common random number $s$ (possibly by multi-party Diffie-Hellman exchange with authentication). Each member sends the number $s$ as a challenge to Alice, signed by the member's private key and then encrypted by Alice's public key.

$$\text{MSG2} \quad D_A(i) \rightarrow Alice: \quad \{[s]_{S_{D_A(i)}}\}_{P_A}, \quad \forall D_A(i) \in D_A$$

**Step 3:** Alice signs $s$ with $S_A(i)$ and then with $S_A$ before sending it back to $D_A(i)$.

$$\text{MSG3} \quad Alice \rightarrow D_A(i): \quad [[s]_{S_A(i)}]_{S_A}, \quad \forall D_A(i) \in D_A$$

**Step 4:** After authentication, if the received $[s]_{S_A(i)}$ value matches the locally computed one, $D_A(i)$ forwards the message to all other members in $D_A$. [1]

[1]Note that $D_A(i)$ knows $s$ and learns $S_A(i)$ from $MSG1$.

MSG4 $D_A(i) \rightarrow D_A(j)$: $[[s]_{S_A(i)}]_{S_A}$, $\forall D_A(j) \in D_A$

Otherwise, $D_A(i)$ files a certified complaint to other members.

MSG5 $D_A(i) \rightarrow D_A(j)$: ["$S_A(i)$ is invalid"]$_{S_{D_A(i)}}$, $\forall D_A(j) \in D_A$

**Step 5:** $D_A(i)$ needs to collect $[s]_{S_A(j)}$, $\forall D_A(j) \in D_A$, which are the partial signatures on $s$. If it receives $MSG4$ $[[s]_{S_A(j)}]_{S_A}$ from $D_A(j)$, the value of $[s]_{S_A(j)}$ is in the message. If it receives $MSG5$ from $D_A(j)$, there are two possibilities: either Alice or $D_A(j)$ is dishonest. To resolve this situation, $D_A(i)$ forwards the certified complaint to Alice. If Alice challenges the complaint, she must disclose the correct value of $S_A(j)$ to $D_A(i)$ in the following message (then $D_A(j)$ can learn $S_A(j)$ from $D_A(i)$).

MSG6 $Alice \rightarrow D_A(i)$: $[\{S_A(j)\}_{P_{D_A(i)}}]_{S_A}$

Learning $S_A(j)$ from this message, $D_A(i)$ can compute $[s]_{S_A(j)}$. After $D_A(i)$ has all $k$ partial signatures on $s$, it can determine that Alice is honest if any $(\lfloor \frac{k}{2} \rfloor + 1)$ partial signatures produce the same signature $[s]_{S_A}$, which can be verified by Alice's public key. Otherwise, Alice must be dishonest.

Since the value of $k$ is typically set small (e.g. 5) and the key distribution is performed once per node, the overhead of the above protocol is not significant. Due to space limitation, we omit the proofs of all theorems in the paper.

*Theorem 1:* The key sharing protocol ensures that all delegation members will either obtain the correct shares of Alice's private key or detect Alice's fraud.

*Contract Verification Protocol*

Both Alice and Bob must register the contract with their delegations so that the money transfer and the optional prosecution can be performed through the delegations at later times. The delegations must verify the information claimed by Alice and Bob in the contract and generate the contract proofs that Alice and Bob need in order to continue their transaction. We design a contract verification protocol to implement the above requirements. The protocol consists of four steps, illuminated in Figure 1 (the left portion), and the number of messages is $O(k)$ for normal cases.

A procedure call is denoted as $x.\mathbf{y}(z)$, which means to invoke procedure $y$ at node $x$ with parameter(s) $z$. If $x$ is a remote node, a signed message carrying $z$ must be sent to $x$ first.

**Step 1:** Alice sends the contract $c$ and a digital signature $c'$ to the delegation $D_A$ for validation. $c'$ may be a signature of the contract concatenating the identifier of the receiver, i.e., $c' = [c|D_A(i)]_{S_A}$. Bob does the same thing.

Alice.**SendContract**(Contract $c$, Signature $c'$)
1. **for** $i = 1$ to $k$ **do**
2. $D_A(i)$.**ComputePartialProof**($c$, $c'$)

**Step 2:** Then the delegation member $D_A(i)$ verifies the reputation claimed by Alice in the contract (denoted as $c.r_A$) and computes a partial signature (denoted as $ps_i$) on the contract with its key share established by the previous protocol.

$D_A(i)$.**ComputePartialProof**(Contract $c$, Signature $c'$)
1. **if** $r_A \geq c.r_A$ **then**
2. $ContractList.add(c, c')$
3. $ps_i = [c]_{S_A(i)}$
4. $D_B(i)$.**DeliverPartialProof**($c, ps_i$)
5. **else** punish(A)

Line 1 verifies whether $c.r_A$ is over-claimed or not. Line 2 saves the contract for later use in Step 3. The signature $c'$ will be used in a procedure called detect(). Line 3 produces a partial signature on the contract by using $S_A(i)$. Line 4 sends the partial signature to the $i$th member of $D_B$. If $c.r_A$ is over-claimed, Alice will be punished at Line 5.

The delegation members in $D_B$ execute a similar procedure except that the condition in Line 1 should be $m_B \geq c.m_B$.

**Step 3:** When $D_B(i)$ receives the contract $c$ and the partial signature $sp_i$ from $D_A(i)$, it executes the following procedure.

$D_B(i)$.**DeliverPartialProof**(Contract $c$, PartialSignature $ps_i$)
1. wait for a timeout period
2. **if** $c$ is found in $ContractList$ **then**
3. $Bob$.**ProcessPartialProof**($ps_i$)
4. **else**
5. detect()

Line 1 waits for a timeout period to ensure that the contract from Bob has arrived. Line 2 checks if the received contract $c$ is also in the local $ContractList$. If that is true, Bob has announced the exact same contract as Alice does, and $D_B(i)$ forwards the partial signature $ps_i$ to Bob. Otherwise, $D_B(i)$ believes that Alice and Bob do not have the same contract, and detect() procedure is executed to detect the malicious participant. Due to the space limitation, we omit the details of the detect procedure.

**Step 4:** After Alice (Bob) receives $t$ or more correct partial signatures, she (he) can compute the contract proof $c_A$ ($c_B$), which can be verified by using Bob's (Alice's) public key.

*Theorem 2:* If both Alice and Bob are honest and their $k$-pair delegation set is trustworthy, the contract verification protocol ensures that they will both receive the correct contract proofs. Otherwise, with high probability, neither one will receive a valid contract proof and the transaction is aborted.

### *D. Phases three and four: Money Transfer and Contract Execution*

Before providing the service, Alice requests its delegation to transfer money, which is illuminated in the middle portion of Figure 1. Upon receiving a money transfer request from Alice, the delegation member $D_A(i)$ invokes the following procedure.

$D_A(i)$.**TransferMoneyProvider**(Contracts $c$, ContractProof $c_A$)
1. **if** valid($c$, $c_A$) and $D_B(i)$.**TransferMoneyConsumer**($c, c_A$)
2. $TM_A = TM_A + c.L$
3. **else** verify()

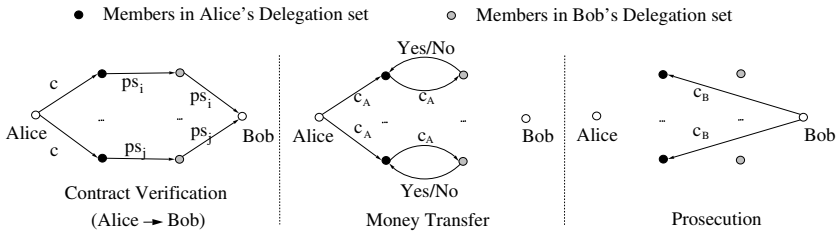


Fig. 1. Protocols for contract verification and exchange (left), money transfer (middle), and prosecution (right).

In Line 1, both $D_A(i)$ and $D_B(i)$ need to validate the contract by using Bob's public key, which can be queried from Bob if it is not locally available. After validation, $D_A(i)$ increases Alice's earned money in Line 2.

Note that $D_B(i)$ may be malicious. If $D_A(i)$ cannot get a positive answer from $D_B(i)$, it must verify the validity of the contract further (Line 3), which can be designed as follows. $D_A(i)$ asks other members in $D_A$. If the majority of $D_A$ have received a positive answer from $D_B$, the contract is considered to be valid ($D_B(i)$ is malicious). Otherwise, the contract is considered to be invalid and Alice is punished.

When $D_B(i)$ receives a money transfer request from $D_A(i)$, it performs the following operations.

$D_B(i)$.**TransferMoneyConsumer**(Contract $c$, ContractProof $c_A$)
1. **if** $valid(c, c_A)$ **then**
2. **if** $m_B > c.L$ **then**
3. $TM_B = TM_B - c.L$
4. **return** true;
5. **else**
6. punish(B)
7. **return** false;
8. **else return** false;

First, if the contract is valid (Line 1) and Bob has enough money to pay the service (Line 2), then Bob's spent money is increased and a positive answer is returned to $D_A(i)$ (Line 3 and 4). Second, it is possible that the contract is valid but Bob does not have enough money. This happens when Alice and Bob are colluding nodes and Alice gets the contract proof $c_A$ directly from Bob instead through her delegation. In such a case, Bob is punished and a negative answer is returned (Line 6 and 7). Third, if the contract is invalid, a negative answer is returned (Line 8).

$D_A(i)$ and $D_B(i)$, $\forall i \in [1..k]$, perform money transfer at most once for each contract. They keep track of the sequence numbers ($Seq_A$ and $Seq_B$) of the last contract for which the money has been transferred. All new contracts have larger sequence numbers.

### *E. Phase five: Prosecution*

After Bob receives the service from Alice, if the quality of service specified in the contract is not met, Bob may issue a prosecution request to Alice's delegation, as illustrated in the right portion of Figure 1. The request specifies the amount of money $f$ that Bob thinks he has overpaid.

Upon receiving a prosecution request from Bob, if $D_A$ cannot evaluate the service quality, it punishes both Alice and Bob by freezing the money overpaid by Bob. The procedure is given as follows.

$D_A(i)$.**Prosecution**(Contract $c$, ContractProof $c_B$, Overpaid $f$)
1. **if** $valid(c, c_B)$ and $f \leq c.L$ **then**
2. $OM_A = OM_A + f$
3. notify($A$)

First $D_A(i)$ validates the prosecution request by checking if the contract proof is authentic (Line 1). If the contract is valid, it increases Alice's overpaid money by $f$ (Line 2). Finally, it notifies Alice so that Alice is able to determine whether to sell service to Bob in the future.

## VI. System Properties and Defense against Various Attacks

### *A. System Properties*

We study the properties of MARCH, which solves or alleviates the problems in the previous approaches.

First, according to the money transfer procedures in Section V-D, transactions among members in the same colluding group cannot increase the total amount of available money of the group. We have the following property, which indicates that the malicious nodes cannot benefit by cooperation.

*Property 1:* Regardless of its size, a colluding group cannot increase its members' money or reputation by cooperation without decreasing other members' money and/or reputation.

Second, unlike some other schemes [1], [2], [4], [5], MARCH does not maintain the history of any consumer's complaints, and does not punish frequent complainers. Thus, we have the following property.

*Property 2:* If a consumer is deceived, it is not restricted by the system in any way from seeking prosecution against the malicious providers.

Third, the overpaid money is not returned to the complaining consumer, which eliminates any reason for the consumer to lie if the consumer is not malicious. If the consumer is malicious and intends to defame the providers, it has to pay the price for the transactions before committing any harm, which serves as an automatic punishment. Consequently, its ability of defaming is limited by the money it has, which cannot be increased artificially by collusion, according to Property 1.

In addition, by Property 2, a deceived consumer can seek revenge with no restriction, which means a malicious provider cannot benefit from its action. We have the following properties.

*Property 3:* A malicious provider cannot benefit by deceiving the consumers, and a malicious consumer will be automatically punished for defaming the providers.

*Property 4:* The maximum amount of loss for an innocent provider or consumer in a transaction with a malicious node is limited by the price specified in the contract.

Property 3 removes financial incentives to cheat. A provider can make money only by serving others; a consumer will not be refunded for cheating. Property 4 makes sure that an innocent node will not be subject to negative discrimination attacks [1], in which nodes with excellent reputation can severely damage other nodes.

In summary, the malicious nodes cannot increase their power (in terms of available money) by cooperation, and they can only attack others at the cost of their own interests, i.e., money and/or reputation. Consequently, the total damage caused by the malicious nodes is strictly limited. They will eventually be rejected from the system due to poor reputation or be enforced to serve others for better reputation in order to stay in the system.

### B. Defending Against Various Attacks

In the following, we consider four different types of attacks launched by a colluding group [1].

*Unfairly high ratings:* The members of a colluding group cooperate to artificially inflate each other's reputation by false reports, so that they can attack innocent nodes more effectively. In MARCH, a colluding group can inflate the reputation of some members only by moving the available money from other members to them. According to Property 1, the total money in the group cannot be inflated through cooperation. Therefore, although some members' reputation can be made better, other members' reputation will become worse, making them ineffective in attacks.

*Unfairly low ratings:* Providers collude with consumers to "bad-mouth" other providers that they want to drive out of the market. Because MARCH requires all consumers to pay money for their transactions before they can defame the providers, the malicious consumers lose their money (and reputation) for "bad-mouthing", which in turn makes it harder for them to stay in the system.

*Negative discrimination:* A provider only discriminates a few specific consumers by offering services with much lowered quality than what the contract specifies. It hopes to earn some "extra" money without damaging its reputation since it serves most consumer honestly. In MARCH, a provider cannot make such "extra" money because of the prosecution mechanism and Properties 2-3.

*Positive discrimination*: A provider gives an exceptionally good service to a few consumers with high reputation and an average service to the rest consumers. The strategy will work in an incentive scheme where a consumer's ability of affecting a provider's reputation is highly related to the consumer's own reputation, and vice versa. MARCH does not have this problem. The provider's reputation changes after a transaction is determined by how much money it receives for the service, not by the reputation of the consumer.

## VII. Discussions

In this section, we discuss other important issues on implementing MARCH. Due to space limitation, we will not discuss them in details.

### A. Rewarding Delegation Members

The system should offer incentive for the delegation members to perform their tasks. A simple approach is for the provider and the consumer of a transaction to reward their delegation members with a certain amount of money.

### B. Money Refilling

Because the overpaid money will be frozen forever, the total amount of available money in the whole system may decrease over the time. As a result, the system may enter into deflation and lack sufficient money for the providers and the consumers to engage in transactions. This problem can be addressed by *money refilling*. The delegation members of a node $A$ will replenish the total money $TM_A$ of the node at a slow, steady rate. In this way, a minimal amount of service is provided to all consumers, even the free-riders, at all time, which we believe is reasonable. For additional service, a consumer has to contribute to the P2P network by also serving as a provider.

### C. System Dynamics and Overhead

In a P2P network, nodes may join/leave the network at any time. When a node $X$ leaves the network, its DHT table will be taken over by the closest neighbor $X'$. In MARCH, suppose $X$ is a delegation member of $A$. After $X$ leaves the network, $X'$ will become a new member in $A$'s delegation. In order to deal with abrupt departure, $X'$ should cache the information kept at $X$, or it can learn the information from other delegation members after $X$ leaves. Moreover, the maintenance of the delegation may be free for a specific DHT network.

The communication overhead of a transaction (excluding the actual service) consists of $O(k)$ control messages. This overhead is quite small when comparing to the services themselves, such as downloading video files of many gigabytes. More importantly, the overhead does not increase with the network size, which makes MARCH a scalable solution, comparing with other schemes [12], [13].

## VIII. Simulation

In our simulations, the dishonest nodes fall into three categories with equal probability.

*Category one:* These nodes never offer services to others after receiving money, and always defame the providers after receiving services.

*Category two:* When these nodes find that they may be rejected from the system, they behave honestly. Otherwise, they behave in the same way as the nodes in category one.

*Category three:* When these nodes find that they may be rejected from the system, they behave honestly. Otherwise, they cheat their transaction partners with a probability taken from $[0.5, 1]$ uniformly at random.

If not explicitly specified otherwise, the system parameters are set as follows. The number of nodes is 100,000 and $k$ is

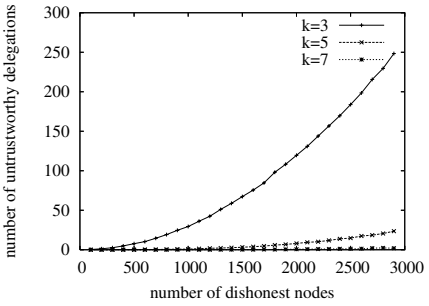


Fig. 2. Trustworthiness of delegation

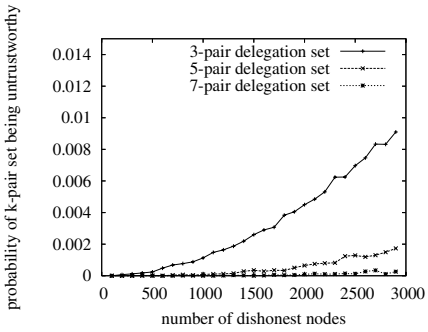


Fig. 3. Trustworthiness of $k$-pair delegation set

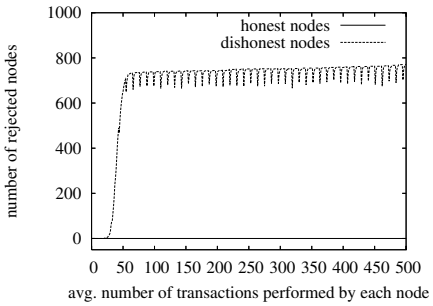


Fig. 4. Most of malicious nodes are rejected within the first 50 transactions.

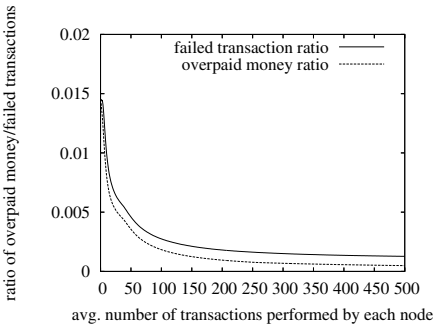


Fig. 5. The failed transaction ratio and the overpaid money ratio drop quickly to small percentages within the first 100 transactions.

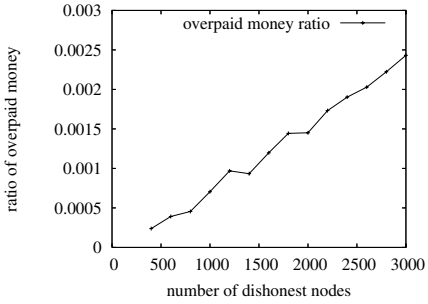


Fig. 6. The overpaid money ratio (measured after 250 transactions) increases linearly with the number of dishonest nodes.

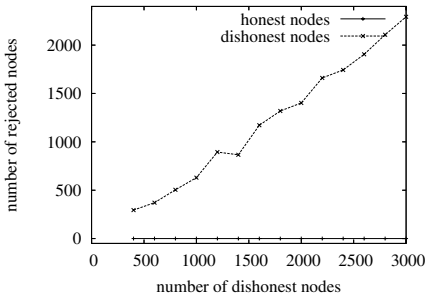


Fig. 7. The number of rejected dishonest nodes (measured after 250 transactions) increases linearly to the number of dishonest nodes.

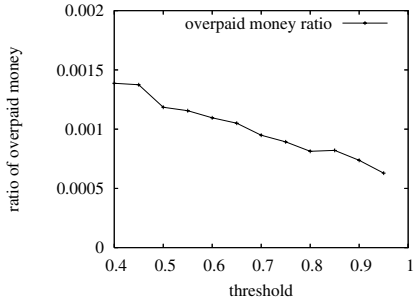


Fig. 8. The overpaid money ratio with respect to the threshold

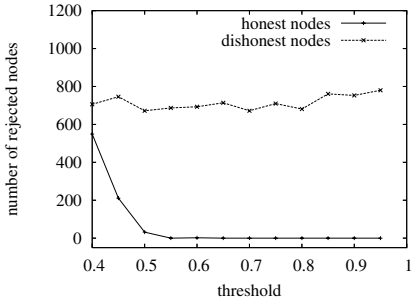


Fig. 9. The number of rejected nodes with respect to the threshold

5. The average number of dishonest nodes is 1,000. Initially, the total money for a node is 500, and the overpaid money is 0. The service price $G$ estimated by the consumers is 10. The threshold $T$ is 0.9. To satisfy the threshold requirement, the maximum selling price for a provider is denoted as $max$ ($max$ is the maximum value of $L$ that keeps $R$ above the threshold, calculated based on Eq. (3). If $max$ is negative, then the node can no longer be a provider. If a dishonest node in Category two or three finds that its $max$ value may become negative after additional malicious acts, it will behave honestly). The actual selling price is a random number taken uniformly from $(0, max]$. If a node can neither be a provider (due to poor reputation), nor be a consumer (due to little money), it is said to be rejected from the system.

If one participant in a transaction tries to deceive the other one, the transaction is called a *failed transaction*. We define "failed transaction ratio" as the number of failed transactions divided by the total number of transactions, and "overpaid money ratio" as the total amount of overpaid money divided by the total amount of money paid in the transactions. These metrics are used to assess the overall damage caused by dishonest nodes.

### A. Effectiveness of Authority

The first set of simulations study the trustworthiness of the delegations and the $k$-pair delegation sets. Figure 2 shows the number of untrustworthy delegations with respect to the number of dishonest nodes, and Figure 3 shows the probability for an arbitrary $k$-pair delegation set to be untrustworthy. By the figures, out of 100,000 delegations/delegation sets, only a few of them are untrustworthy. Even when there are 3,000 dishonest nodes, the number of nodes with untrustworthy delegations for $k = 5$ is just 23, and the 5-pair delegation

set is trustworthy with a probability larger than 99.8%. Note that when a delegation is untrustworthy, the dishonest members may not belong to the same colluding group. Without cooperation, the damage they can cause will be smaller.

### B. Effectiveness of MARCH

The second set of simulations study the effectiveness of our incentive scheme. Figure 4 presents how the number of rejected nodes changes with the average number of transactions performed per node, which can be used as the *logical time* as the simulation progresses. Recall that the default number of dishonest nodes is 1,000. The figure shows that most dishonest nodes are rejected from the system within 50 transactions per node. Because of money refilling, some rejected nodes will recover after enough money is refilled, but they will be rejected again after performing malicious transactions. No honest nodes are rejected from the system during the simulation.

Figure 5 shows that the failed transaction ratio drops quickly from 1.4% to 0.3% within the first 100 transactions per node, and the overpaid money ratio drops from 1.4% to 0.2% in the same period. As the time progresses, these ratios become even more insignificant. Note that the overpaid money ratio is smaller than the failed transaction ratio. This is because the dishonest providers have to lower their prices in order to compete with honest providers, which in turn lowers their ability to cause significant damage. Ironically, if a dishonest node with poor reputation wants to stay in the system, not only does it have to behave honestly to gain reputation, but also it has to do so with lower price in order to get consumers, which "repairs" the damage it does to the system previously.

Next, we study how the number of dishonest nodes affects the system performance. Figure 6 shows the overpaid money ratio after 250 transactions per node. We find that the ratio increases linearly with the number of dishonest nodes. However, even when there are 3,000 dishonest nodes, the overpaid money ratio remains very small, just 0.15%. Figure 7 shows that the more the number of dishonest nodes, the more they are rejected.

Last, we study the impact of the threshold on the system performance. The threshold is used by a consumer to select the potential providers (Section V-B). Figure 8 shows that the overpaid money ratio decreases linearly with the threshold value, which means the system performs better with a larger threshold. Figure 9 shows that the number of rejected dishonest nodes is largely insensitive to the threshold value. However, when the threshold is too low, some honest nodes may be rejected by the system because a smaller threshold allows the dishonest nodes to do more damage on the honest nodes, which may even cause some honest nodes to be rejected from the system due to defamed reputation. The numbers in the above two figures are measured after 250 transactions per node.

## IX. Conclusion

We propose a distributed incentive scheme (called MARCH) for P2P networks. The scheme uses a distributed authority infrastructure with delegations, instead of a centralized server, to maintain the money/reputation information of the nodes. We use a five-phase transaction framework to incorporate both virtual money and reputation into our scheme, which solves a number of problems that the previous schemes have. We also present a key sharing protocol and a contract verification protocol to produce the contract proofs that are authorized by the delegations of the provider and the consumer of a transaction. We analyze the system properties and use simulations to evaluate the system performance. The results demonstrate that MARCH has the potential to solve the free-riders problem in today's P2P networks.

## References


[1] C. Dellarocas, "Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior," *Proc. of EC'00: the 2nd ACM conference on Electronic commerce*, 2000.

[2] M. Srivatsa, L. Xiong, and L. Liu, "Trustguard: Countering vulnerabilities in reputation management for decentralized networks," *Proc. of WWW'05*, May 2005.

[3] P. Dewan and P. Dasgupta, "Securing reputation data in peer-to-peer networks," *Proc. of Parallel and Distributed Computing and Systems*, 2004.

[4] Y. Wang and J. Vassileva, "Bayesian network trust model in peer-to-peer networks," *Proc. of AP2PC'03*, 2003.

[5] M. Venkatraman, B. Yu, and M. P. Singh, "Trust and reputation management in a small-world network," *Proc. of ICMAS'00*, 2000.

[6] S. Marti and H. Garcia-Molina, "Identity crisis: Anonymity vs. reputation in p2p systems," *in Third IEEE International Conference on Peer-to-Peer Computing*, 2003.

[7] M. Jakobsson, J. Hubaux, and L. Buttyan, "A micropayment scheme encouraging collaboration in multi-hop cellular networks," *Proc. of Financial Crypto'03.*, 2003.

[8] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer, "Karma: A secure economic framework for p2p resource sharing," *Proc. of the Workshop on the Economics of Peer-to-Peer Systems*, June 2003.

[9] Y. Zhang, W. Lou, and Y. Fang, "Sip: a secure incentive protocol against selfishness in mobile ad hoc networks," *Proc of WCNC'04*, March 2004.

[10] J. C. L. T.B. Ma, Sam C.M. Lee and D. K. Yau, "A game theoretic approach to provide incentive and service differentiation in p2p networks," *Proc. of ACM SIGMETRICS/PERFORMANCE*, June 2004.

[11] L. Buttyn, "Removing the financial incentive to cheat in micropayment schemes," *IEE Electronics Letters, Vol. 36 No. 2, pp.132-133*, January 2002.

[12] S. Lee, R. Sherwood, and B. Bhattacharjee, "Cooperative peer groups in nice," *Proc. of INFOCOM'03*, Apr 2003.

[13] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust incentive techniques for peer-to-peer networks," *in ACM Electronic Commerce.*, 2004.

[14] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *Proc. of ACM SIGCOMM'01*, pp. 149–160, August 2001.

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," *Proc. of ACM SIGCOMM'01*, August 2001.

[16] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Proc. of Middleware'01*, November 2001.

[17] M. Castro, P. Drushel, A. Ganesh, A. Rowstron, and D. Wallach, "Secure routing for structured peer-to-peer overlay networks," *Proc. of OSDI'02*, 2002.

[18] J. R. Douceur, "The sybil attack," *Proc. of IPTPS'01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pp. 251–260, 2002.

[19] Y. G. Desmedt and Y. Frankel, "Threshold cryptosystems," *Proc. of CRYPTO'89*, pp. 307–315, 1989.

[20] E. Friedman and P. Resnick, "The social cost of cheap pseudonyms," *Journal of Economics and Management Strategy*, 2001.