Off-line Karma: A Decentralized Currency for Static Peer-to-peer and Grid Networks*

Flavio D. Garcia
Jaap-Henk Hoepman
Nijmegen Institute for Computing and Information Sciences (NIII)
Radboud University Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, the Netherlands
flaviog,jhh@cs.ru.nl

23rd November 2004

Abstract

Peer-to-peer (P2P) and grid systems allow their users to exchange information and share resources in a uniform and reliable manner. In an ideal world, users make roughly as much resources available as they use. In reality, this is not always the case, and some kind of currency or barter (called *karma*) is needed that can be exchanged for resources to limit abuse. Previous proposals for such systems have not taken the decentralised and non-hierarchical nature of P2P and grid systems into account.

We present a completely decentralised, off-line karma implementation for P2P and grid systems, that detects double spending and other types of fraud under varying adversarial scenarios. The system is based on tracing the spending pattern of coins, and distributing the normally central role of a bank over a predetermined, but random, selection of nodes.

1 Introduction

Peer-to-peer (aka. *P2P*) networks like BitTorrent [Coh03] and Gnutella [Kir], and grid systems like XGrid are distributed systems without centralised control or hierarchical organisation. Given this flat structure, these systems scale very well when the number of nodes increases. Scalability is important, given the fact that the Internet is still growing exponentially and more people have permanent Internet connections. Current applications of these systems include but are not limited to: file sharing, redundant storage, distributed computations, data perpetuation, P2P e-mail, P2P web cache and anonymity.

Grid systems have been developed as a response to the fact that computer resources are usually very badly distributed in time and space, and almost all of them are wasted most of the time. CPU cycles are maybe the best example of this. Most of the computers in the world are idle most of the time, with only occasional periods of high load. Then, it seems natural to make

*Id: karma-inc.tex,v 1.4 2004/11/23 11:54:57 jhh Exp

resources available when idle, and to be able to use other users resources in return when needed. In an ideal grid system, the whole Internet constitutes a huge supercomputer with practically unlimited resources, that members can use as long as they contribute to it as well.

Projects like seti@home, folding@home and distributed.net have shown that a large set of common desktop computers can provide a tremendous amount of computing power. Even though they receive no direct benefit, users participate in such projects because they associate themselves with the goals of the project, and for the very odd chance of being the lucky guy that finds the solution. If such large scale computations are for an uncompelling cause, it is not easy to find people willing to donate their CPU time.

Also, many P2P networks suffer form the 'free-riders' problem where users only occasionally connect to the network to use the resources offered by it, but do not donate any resources themselves. To counter such problems, 'currencies' of some sort have been proposed to reward users contributing to the system and that can be used as payment when the resources of the network are used.

1.1 Related work

Several P2P systems like POPCORN [NLRC98] and MojoNation[1] use some kind of digital currency to enforce contribution and optimise resource distribution. All these systems use a central bank or broker to track each user's balance and transactions. Micropayment schemes like Rivest and Shamir's PayWorld and MicroMint [RS97], Glassman et al's Millicent [GMA+95] and Rivest's Pepercoin [Riv04] seem to be especially suitable for such a task. However, these schemes are centralised and the load of the central broker grows linearly with the number of transactions.

It is clear that when scalability is of primary concern, a central bank or broker constitutes both a bottleneck as well as a single point of failure. In the context of P2P networks, avoiding lawsuits is an extra motivation for having a decentralised system. The aforementioned approaches are therefore not suitable to fully solve our problem. Furthermore, there is no trivial way to decentralise them given that the trusted bank plays an important role in such a schemes.

At the moment, the only distributed currency we are aware of that is fully-decentralised is KARMA [VCS03]. KARMA splits the bank functionality in different bank sets, which are sets of users of size r. Each of these bank sets is responsible for keeping the state balance of a set of users. In KARMA, every transaction between users Alice and Bob involves communication between them, between Alice and Bob and their bank sets, and most importantly, it involves r to r communication between the bank set of Alice and the bank set of Bob. This incurs a large overhead, especially in cases where the transaction rate is high.

Another interesting approach is PPay [YGM03]. PPay is a lightweight micropayment scheme for P2P systems, at least from the point of view of the users. In PPay the issuer of the coin is responsible for keeping trace of it. with every transaction the issuer of the coin updates a pointer to the new owner, in a secure manner. The main drawback with PPay is that it uses a central server (called broker) when the issuer of a coin is off-line. This means that when Alice, who owns a coin minted by Carol who is off-line, wants to spend it at Bob, Alice should make the transaction via a central broker. In some frameworks, were the ratio of off-line users is high or in very dynamic systems were users join at some point and never reconnect again, the probabil-

[1] MojoNation no longer exists, but the website has been archived. See web.archive.org/web/*/mojonation.net/*.

ity of finding the original issuer of the coin on-line is very low. In this situation PPay converges to a system with a centralised accounting bank.

1.2 Our contribution

We present a completely decentralised, off-line karma implementation for P2P and grid systems, that detects double spending and other types of fraud under varying adversarial scenarios. The system is based on the tracing of the spending pattern of coins, and distributing the normally central role of a bank over a predetermined, but random, selection of nodes. Transactions between users do not require the cooperation of this distributed bank. Instead, karma coins need to be occasionally reminted to detect fraud. The system described in this paper assumes that nodes do not join or leave the system. However, the system can be extended to allow nodes to join and leave the system at arbitrary times. Due to space considerations, we do not discuss these extensions in this paper. In our system, a transaction only involves communication between the two users exchanging the coin. As a trade off, we can only provide double-spending detection instead of prevention.

We focus on the payment for CPU cycles as an application of our techniques, and show how a special minting technique allows us to initialise the system and provide its users with coins in a quite autonomous fashion. Then the karma coins correspond to CPU cycles, and the bag of coins owned by a user corresponds, in a sense, to a battery charged with CPU cycles. When a user connects to the system this battery is empty. It can be charged by minting coins, but this takes quite a few CPU cycles (because it involves finding collisions in a hash function). Alternatively, a coin can be obtained by performing roughly the same amount of work, but for another user. Extensions of our protocols to trade coins for other resources are certainly possible, and only involves initialising the system with a set of coins in a different manner.

In order provide some abstraction, we build our system on top of an arbitrary overlay network which provides certain services as described in Section 2. The reader can think in a layered structure where TCP/IP is on bottom, and over it there is a P2P routing layer, and then a secure routing layer and on top of them is build the Off-line karma protocol.

2 System Model

In the context of this paper we want to stay abstracted from the underlying overlay network. We are going to model common characteristics that apply to routing overlays like CAN [RFH+01], Chord [SMK+01], Pastry [RD01] and Tapestry [ZHR+04] as in [CDG+02], were the reader can find also a nice and brief description of each system. In this abstract model, every node that joins the system is assigned a uniform random identifier u from the identifier space Π. We assume that the overlay network provides primitives for both user look-up and message routing. Furthermore, for each possible identifier u (whether u is part of the network or not) the overlay network can efficiently and reliably compute the *neighbour set* $\aleph_r(u)$, which consist of the first r on-line nodes *close* to u. The definition of *close* varies in each of the above mentioned systems, although it is always well-defined. We also assume that communication within this set is *efficient*, because nodes keep updated routing information of their neighbours. Given that the node identifiers are distributed randomly, any neighbour set represents a random sample of all participating nodes [CDG+02].

Off-line Karma requires every user to have his own public key pair (PK, SK) and a certificate that binds the public key with a node identifier. This may be provided by a trusted *Certification Authority* (aka. *CA*). We want to remark that the CA is only needed when a new user joins the system. After that communication with the CA is no longer needed.

Routing information in the overlay network is kept updated, in practise, by node join and node leave messages and periodic queries and fingers to detect when a node suddenly disconnects. This mechanism introduces propagation and update delays. This is, in fact, a discrete approximation of an ideal situation where any modification in the network topology is instantaneously detected by the overlay network. We assume such an ideal situation, and leave responsibility for emulating this ideal functionality in an efficient fashion to the overlay network. We also assume that node joins and leaves are atomic operations.

We also assume that the overlay network is capable of safely distributing a blacklist of banned users. Whenever a user detects fraud and has a proof of that, he can submit it to the overlay network and this information is available for every user. How to implement the distribution of blacklist securely is beyond the scope of this paper. We simply assume the overlay provide this service.

2.1 Notation

We write $\{m\}_u$ for u's signature on message m, C_u for u's certificate, and $\texttt{validSig}(m, u, C_u)$ for the function that checks u's certificate C_u, and if valid uses the key in C_u to verify a signed message m.

We also use a multisignature scheme. A multisignature scheme [OO99, MOR01] is a signature scheme where a set R of users sign a message. A multisignature $\{m\}_R$ for a message m has the same properties as if each user in R concatenates his own traditional public key signature to m, the only difference is that a multisignature is more efficient in size and in verification time (comparable to a single signer Schnorr's signature scheme). Unlike a threshold signature scheme however it does not provide anonymity. We define $C_R = \{C_i : i \in R\}$ and $\texttt{validSig}(m, R, C_R)$ is the function that checks the certificates and verifies the multisignature.

3 System Objectives and Threat Model

3.1 Threat Model

In our threat model, we assume to have a set of n users U of which at most t are under control of the adversary. In the context of P2P networks, we also need to model the fact that there is an important difference in the difficulty for an adversary between adding new corrupted users to the system and getting control over chosen users. Therefore, we also define $0 \leq c \leq t$ to be the number of corrupt users chosen by the adversary after they joined the overlay network. Then, when $c = t$ we give the adversary full control over which nodes in the overlay get corrupted, while for $c = 0$ the adversary is only able to get a randomly chosen set of corrupted users of size t.

Furthermore, we assume that the adversary cannot make excessively many nodes join and leave the system, or let some nodes join and leave in a very high frequency (in attempts to mount syble attacks, to use them as straw men, or to overcome the random assignment of node

identifiers). In fact, we do not allow the adversary any control over when nodes join the system. In practise, this could be achieved by requiring nodes to pay each time they register, or making node joins a time-intensive procedure (e.g., by requiring them to compute a moderately hard, memory bounded function [ABMW03, DGN02]).

3.2 System objectives

Ideally, the system should *prevent* double spending of coins. However, for any system offering off-line currency, double-spending *prevention* is generally speaking not possible, unless extra assumptions (e.g., special tamper proof hardware) are made. Therefore, we only require double spending detection. We stress that we do not consider issues like fair exchange or coin stripping. We focus on the payment itself and not on the exchange of coins for goods. To be precise, our requirements on a usable karma system for P2P and grid applications are the following.

Scalability Transaction cost should be independent of the size of the network.

No centralised control The system should not rely on one or several central, special, nodes (e.g., banks or brokers) and should not require any predetermined hierarchy. We do allow a centralised registration procedure.

Load Balance The overhead of the protocol is, on average, evenly distributed over the peers.

Availability Transactions among users can be processed uninterrupted even when users join or leave the system.

Double-spending detection The system must detect double spending, and for every double spent coin, a fraudulent user should be blacklisted.

4 The Off-Line Karma Protocol

4.1 Informal description

The system manages the minting and transfer of karma coins and double spending detection. To implement the CPU cycles battery metaphor presented in the introduction, coins can be minted by a user by finding collisions on a hash function (a la hashcash [Bac97]). A minted coin contains the name of the minting user as well as a sequence number (limiting the number of coins a single user can mint). User identity and sequence number together constitute the unique coin identity. Coins also contain a time stamp recording the time they were minted.

The coins are transferable [CP92]. A user can pay for resources by transferring a coin to another user. The sender signs the coin, and the receiver verifies this signature and stores the coin (with signature) for further use. With every transfer, a coin is extended with another signature. Thus, the sequence of signatures on a coin record the payment history of that coin. Double spending is detected by comparing the history of two coins with the same coin identity, and the culprit (or his accomplice) will be found at the node where both histories fork. This check is performed whenever a coin is reminted.

Every once in a while (but at least before the coin expires), coins must be reminted. Reminting is used to detect double spending, and at the same time to reduce the size of the coin by removing its history. In classical systems, reminting is done by a central bank. Here the

function of the bank is distributed over a set of users on the network called the *reminters* for the coin. The set of reminters is constructed in such a way that

- at least one of the reminters is a non-corrupted node, and
- all honest reminters possess the history of previously reminted coins with the same identity.

4.2 Detailed description

We now present a detailed description of the minting, spending and reminting procedures in our system. Due to space considerations, a detailed protocol description as well as the security analysis have been omitted from this extended abstract.

Mint Let $h_1 : A \to C$ and $h_2 : B \to C$ be hash functions, and suppose every user is allowed to mint 2^q karma coins. A user u has to spend some CPU time finding a collision y satisfying: $h_1(x) = h_2(y)$ and $x \neq y$, were:

$$\begin{aligned} x &= \underbrace{u||sn}_{coinId}||ts \\ sn &= \text{serial number} : |sn| \leq q \\ ts &= \text{time stamp} \end{aligned}$$

This is an expensive but feasible operation, for suitable functions h_1 and h_2. In analogy with the monetary system, one can think that the cost of the metal needed for minting a coin is greater than its nominal value. We define the new karma coin as

$$k_0 = \langle x, y \rangle$$

Spending For spending a coin the owner just endorses it to the merchant, including merchants's Id and random challenge z. The reason for the random challenge is to avoid the uncertainty in the case that the double-spender gives the same coin twice to the same user. Otherwise, a fair user would look like the double-spender or should keep the history of the received coins forever. Concretely, suppose that the user s owns the coin k_i and wants to spend it at the user m. Then, the last one sends a random challenge z to the first one who computes:

$$k_{i+1} = \{k_i, z, m, C_s\}_s$$

and sends it to m.

Reminting To prevent the coins to grow unreasonably large and to bound the amount of history that needs to be kept, coins must be reminted regularly, at least within the time to live T.

In a general micropayment schemes, this process is performed at the bank. In our system, instead, the bank functionality is performed by a random but predefined set of users R_k. The selection of this set must be done in such a way that

- each user is responsible for reminting roughly the same amount of coins (load balance) and
- at least one honest user is a member of the remint set.

Whenever a user u has to remint a coin k, he sends it to each user in the remint set $R_k = \aleph_r(h(id(k)))$. Here the hash function is used as a consistent mapping from the coin identifier space to Π. Each user in R_k must verify coin k and store it in his local history database. Verification entails

- checking whether the coin has not expired, and
- checking the chain of signatures on the coin, and
- checking the collision in the base of the coin, or the multisignature on a reminted coin, and
- checking that the coin was not reminted before (and if it was reminted before, find the perpetrator by comparing the chain of signatures on both coins).

The size r of the remint set R_k is chosen in such a way that the probability that R_k contains at least one honest node is exponentially close to 1. In fact, one can show that if $r > \alpha s + c$ for constants α and c, then this probability is larger than $1 - 2^{-s}$ (where s is the security parameter of the system).

Note that for a static network, R_k solely depends on $id(k)$. Therefore, any double-spent coin will be noticed by the honest nodes in R_k, which will refuse to remint the coin. If the verification succeeds, the reminters will create a multisignature

$$k_{new} = \{\mathtt{XY}(k), ts, R_k, C_{R_k}, u\}_{R_k}$$

for the new coin with the same coin identifier and owner, but with a new time stamp (where $\mathtt{XY}()$ returns the pair x, y in the base of the coin). If the verification fails, either because the coin is invalid or because a coin with the same identifier and time stamp was already reminted, the reminters will audit the coin and trace back the cheater in the signature chain.

5 Conclusions

We have presented a completely decentralised, off-line karma implementation for P2P and grid systems, that detects double spending and other types of fraud under varying adversarial scenarios. In this extended abstract, we focussed on the static case where no nodes join or leave the system. By making the remint set larger, and by restricting the rate of growth (or shrinkage) of the number of nodes in the system, we are also able to handle the dynamic case efficiently.

References

[ABMW03] ABADI, M., BURROWS, M., MANASSE, M., AND WOBBER, T. Moderately hard, memory-bound functions. In *Proc. NDSS 2003* (San Diego, CA, 2003), Internet Society, pp. 25–39.

[Bac97] BACK, A. Hashcash - a denial of service counter-measure. http://www.cypherspace.org/hashcash, 1997.

[CDG+02] CASTRO, M., DRUSCHEL, P., GANESH, A. J., ROWSTRON, A. I. T., AND WALLACH, D. S. Secure routing for structured Peer-to-Peer overlay networks. In *Proc. 5th OSDI* (New York, 2002), ACM Press, pp. 299–314.

[CP92] CHAUM, D., AND PEDERSEN, T. P. Transferred cash grows in size. In *EUROCRYPT 92* (1992), R. A. Rueppel (Ed.), vol. 658 of *LNCS 658*, Springer-Verlag, pp. 390–407.

[Coh03] COHEN, B. Incentives build robustness in bittorrent. In *Proc. of the Workshop on Economics of Peer-to-Peer Systems* (Berkeley, CA, USA, 2003).

[DGN02] DWORK, C., GOLDBERG, A., AND NAOR, M. On memory-bound functions for fighting spam. In *CRYPTO 2003* (2002), D. Boneh (Ed.), LNCS 2729, Springer-Verlag, Berlin Germany.

[GMA+95] GLASSMAN, S., MANASSE, M., ABADI, M., GAUTHIER, P., AND SOBALVARRO, P. The MilliCent protocol for inexpensive electronic commerce. In *4th Int. Conf. on the World-Wide-Web* (MIT, Boston, 1995).

[Kir] KIRK, P. Gnutella. http://rfc-gnutella.sourceforge.net.

[MOR01] MICALI, S., OHTA, K., AND REYZIN, L. Accountable-subgroup multisignatures: extended abstract. In *Proc. 8th CCS* (Philadelphia, PA, USA, 2001), P. Samarati (Ed.), ACM Press, pp. 245–254.

[NLRC98] NISAN, N., LONDON, S., REGEV, O., AND CAMIEL, N. Globally distributed computation over the internet – the POPCORN project. In *18th ICDCS'98* (Amsterdam, The Netherlands, 1998), IEEE, pp. 592–601.

[OO99] OHTA, K., AND OKAMOTO, T. Multi-signature scheme secure against active insider attacks. In *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences* (1999), pp. E82–A(1): 21–31.

[RFH+01] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., AND SHENKER, S. A scalable Content-Addressable network. In *Proc. ACM SIGCOMM 2001* (New York, 2001), R. Guerin (Ed.), ACM Press, pp. 161–172.

[Riv04] RIVEST, R. L. Peppercoin micropayments. In *Proc. FC '04* (2004), A. Juels (Ed.), LNCS 3110, Springer, pp. 2–8.

[RS97] RIVEST, R. L., AND SHAMIR, A. PayWord and MicroMint: Two simple micropayment schemes. In *Proc. Int. Workshop on Security Protocols* (Cambridge, United Kingdom, 1997), M. Lomas (Ed.), LNCS 1189, Springer-Verlag, Berlin Germany, pp. 69–87.

[RD01] ROWSTRON, A., AND DRUSCHEL, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)* (2001), pp. 329–350.

[SMK+01] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. 2001 Conf. on Applications, technologies, architectures, and protocols for computer communications* (2001), ACM Press, pp. 149–160.

[VCS03] VISHNUMURTHY, V., CHANDRAKUMAR, S., AND SIRER, E. G. KARMA: a secure economic framework for peer-to-peer resource sharing. In *Proc. of the Workshop on the Economics of P2P Systems* (Berkeley, California, 2003).

[YGM03] YANG, B., AND GARCIA-MOLINA, H. PPay: micropayments for peer-to-peer systems. In *Proc. 10th CCS* (New York, 2003), V. Atluri and P. Liu (Eds.), ACM Press, pp. 300–310.

[ZHR+04] ZHAO, B. Y., HUANG, L., RHEA, S. C., STRIBLING, J., JOSEPH, A. D., AND KUBIATOWICZ, J. D. Tapestry: A global-scale overlay for rapid service deployment. *IEEE J-SAC* **22**, 1 (2004), 41–53.