Bootstrapping a Distributed Computational Economy with Peer-to-Peer Bartering

Brent Chun
Intel Research Berkeley
bnc@intel-research.net

Yun Fu
Duke University
fu@cs.duke.edu

Amin Vahdat
Duke University
vahdat@cs.duke.edu

Abstract

This paper presents an architecture for distributed computational economies based on peer-to-peer bartering. Our architecture is based on the position that computational economies ought to be bootstrapped based on a layer of simple and robust resource exchange. The architecture is comprised of three pieces: (i) resource discovery, (ii) secure resource peering, and (iii) bartering. Together, these pieces address the end-to-end problem of describing, discovering, and exchanging distributed resources in a secure and decentralized manner. Key in our approach is the ability to securely exchange resources across delegated paths of trust. This, combined with secure resource peering, allows peers to engage in resource exchange with directly connected peers, in addition to peers whom they do not have direct bartering relationships with. Given the bartering economy as a base, we envision an evolutionary path towards more complex scenarios by layering richer functionality at higher layers.

1 Introduction

Over the last two decades, numerous proposals [16, 14, 3, 12, 9, 17] have emerged for economic-based distributed resource management in large-scale distributed systems. Still, we have yet to observe the widespread deployment and use of such systems in practice. We argue that a key reason for such lack of adoption is the initial complexity of the proposals. The vast majority of previous proposals presume both the existence and widespread acceptance of abstract currencies (and associated infrastructure) from the start. In contrast, history shows that real economies do not evolve this way. Real economies virtually always begin with simple bilateral exchange based on bartering. A number of successful large-scale, distributed systems similarly are rooted in simple base functionality and introduce richer functionality at higher layers [15, 5]. In both cases, simplicity and robustness are critical to promoting growth and providing the foundations for layering additional complexity. Given these historical precedents, we propose that a distributed computational economy should be bootstrapped based on a thin layer that enables simple and robust resource exchange through bartering and that richer functionality ought to be layered on top.

Our work is motivated by providing transparent access to remote resources, largely in the context of network services, where it is advantageous to have multiple vantage points on the network. Of course, we believe that many of our goals are applicable to other distributed systems as well, in particular the Grid [10]. We envision a bartering economy as providing the basis for decentralized growth and as a foundation for layering additional functionality at higher layers. Examples of higher level functionality include abstract currencies, computational analogues of financial instruments (e.g., CPU futures), and distributed, incentive-compatible economic mechanisms [8].

In this paper, we present a baseline architecture for bootstrapping distributed computational economies based on *peer-to-peer bartering*, with an eye to its support in the PlanetLab network testbed [13]. The architecture consists of three pieces: (i) resource discovery, (ii) secure resource peering, and (iii) bartering. Together, these pieces address the end-to-end problem of describing, discovering, and exchanging distributed resources in a secure and decentralized manner. Work on an implementation is currently underway.

2 Resource Discovery

Resource discovery is the process of binding specific resources to an abstract description of the services required for a particular user or program. This process requires a number of components: i) a resource description language that allows peers to describe the resources they are making available for sharing, ii) a query language that allows a peer to express resources of interest, iii) a resource discovery system that accepts query requests, evaluates them, and returns query results. A resource discovery system takes the resource availability information provided by peers and exposes that information to other peers through a query language.

2.1 Resource Description Language

The resource description language is a common language used to describe computational resources. Peers in the baseline bartering economy use this language for two purposes. First, the language expresses resources available for sharing before any bartering occurs. A standard protocol for exposing this information might then be used to enable automated querying of peers to dynamically discover what resources are currently available and to subsequently engage in a bartering protocol to agree on an exchange rate. Second, the language expresses local resources being exchanged through ongoing bartering relationships and remote resources available as a result of bartering relationships (i.e., analogous to advertising the reselling of a peer's resources acquired through bartering).

Thus, the resource description language essentially expresses the terms of a bartering exchange. It should be expressive since the resources being advertised will have varying levels of complexity and it should be extensible, since we do not know a priori what resources peers might want to advertise. Concretely, it might describe the number of nodes being made available by a peer, which in turn might represent the interests of an entire administrative domain. Classes of nodes might then be characterized based on their CPU, memory, network, I/O, and storage capacity, which in turn might be provided in contiguous, non-overlapping time intervals ΔT in length. To acquire resources for a long-running application, peers would then need to engage in continuous rounds of bartering to continuously acquire resources for their applications.

2.2 Resource Discovery Systems

Our architecture does not prescribe a single resource discovery system. For flexibility and to encourage healthy competition, it instead provides the baseline language to express available resources at a low-level and allows multiple, competing, co-existing resource discovery systems to use this information and expose it using query languages which provide varying degrees of expressiveness and expose information at different levels of abstraction. Resource discovery systems are initially likely to expose resources at a level of abstraction that mirrors that of the underling resource description language. Longer term, we envision resource discovery systems that allow resources of interest to be succinctly described at a high level. In all cases, queries presented to the system must ultimately be mapped back to a set of distributed resources and the names of the peers where those resources are available.

A resource discovery system must ultimately be be scalable, fault-tolerant, and decentralized. Initially, we might start with simple centralized solutions based on established technology such as relational databases. Longer term, we envision transitioning to to emerging, decentralized, distributed query processing systems such as PIER [11] and IrisNet [6].

3 Secure Resource Peering

Once the peers with required resources have been discovered, a reliable, accountable, and scalable resource exchange framework must be established to provide the mechanisms for peer-to-peer resource bartering. We have implemented a *secure highly available resource peering* (SHARP) system [4] on PlanetLab [13], a global overlay testbed, for discovering and sharing resources based on pairwise resource exchange and resource exchange across paths of delegated trust.

SHARP represents resource delegation using *tickets*, which assert that a peer (*the holder*) controls a set of another peer's resources over some time interval (*its term*). Each ticket is signed with the private key of the resource owner (*the issuer*). A holder may delegate a portion of the resources claimed in a ticket to another peer by issuing a *subticket* for a subset of the resources over a subinterval of the term, signing and concatenating it with the original ticket as a new ticket. The cryptographically signed tickets are unforgeable, non-repudiable, and independently verifiable by third parties.

3.1 Oversubscription

The holder of a ticket can redeem it from the issuer for resources specified in the ticket. However, a resource owner may oversubscribe its resources by issuing more tickets than it can support to improve resource availability and utilization. A ticket holder can also replicate the ticket to multiple peers. Thus, the issuer of a ticket only honors the ticket with probabilistic assurance. The probability that an issuer honors a ticket depends on the oversubscription degree and the rate of ticket redemption. If the issuer honors the ticket, the specified resources are allocated and a *lease* is returned to the ticket holder as a hard guarantee for the ownership of the resources. A lease can be renewed to allow for continuous use of the same resources. Such renewals may or may not involve acquiring additional tickets. Thus, a ticket is a soft claim for resources, while a lease is a hard claim for resources. Peers exchange resources with tickets rather than leases.

Since tickets can be replicated arbitrarily, a ticket issuer must maintain a ticket tree for all redeemed tickets and the amount of redeemed resources with each ticket to detect conflicts of delegation. To build the tree, each delegation in a ticket must be associated with a global unique ID to identify the transaction. Thus each ticket can construct a unique path through a leaf ticket to the root ticket on the issuer ticket tree, where the root ticket must be released by the issuer. If a redeemed ticket does not exceed the resource amount allocated to its ancestors, the issuer honors the ticket. If the ticket causes a conflict at any ancestor, the issuer refuses to redeem the ticket and considers the ancestor at the earliest conflict to be *accountable* for the conflict. So tickets provide an accountable means for SHARP to support transitive resource delegation, the basis for resource exchange in SHARP.

The optimal oversubscription degree is a function of multiple target metrics, including utilization, availability, and rejection rate. An oversubscription degree of 1 implies that a peer issues exactly enough tickets that it can support given its resources. In this case, if some paths of trust to a given peer are not being fully utilized, then tickets issued by that peer to some subset of its immediate peers may go unused while associated resources go idle. By increasing the oversubscription degree, both utilization and availability can increase in the presence of unused tickets. On the other hand, by increasing the oversubscription degree, the probability of a rejection also increases. The observed rejection rate will be a function of the oversubscription degree, the peering graph, and the workload. The optimal oversubscription degree will be one that balances desired utilization and availability against the expected rejection rate (see [4] for a quantititive analysis of these trade-offs).

Since a ticket is a soft claim, it can potentially be rejected when being redeemed for resources. Rejection can occur either because the desired resources were oversubscribed and are being fully utilized, or because a peer is acting maliciously and claiming its resources are fully utilized when in fact they are not. Because oversubscription can potentially result in tickets being rejected under normal operation, peers must already be prepared to handle ticket rejection. Determining whether a peer rejected a request intentionally or not is a separate matter. From a peer's point of view, if requests are being rejected, the bottom line is that resources are not being acquired. In response to this, we propose that peers renegotiate exchange rates based on the number of rejections. If a peering relationship is proving not to be fruitful, then it is probably in the peer's best interest to either renegotiate for a more favorable exchange rate or stop peering altogether.

3.2 Paths of Trust

In SHARP, peers can obtain tickets from each other by pairwise ticket exchange if they establish a trust relationship. Also, peers can dynamically discover and exchange tickets with a remote peer without a direct trust relationship by discovering a path to the remote peer through a series of trust relationships and recursive pairwise ticket exchange. Currently, SHARP uses a BGP-like protocol SRDP (*secure resource discovery protocol*) to discover paths of trust. Each peer advertises to all direct peers the routes it uses to reach other peers. Eventually each peer maintains multiple paths to other peers. When a peer attempts to obtain tickets for resources from a remote peer, the peer selects a path based on any path selection algorithm and starts ticket bartering with the next-hop peer on the path, which in turn repeats pairwise bartering until the required tickets are obtained. Since SRDP enables a peer to maintain trust relationships with a limited number of peers and obtain global resources from all peers reachable through paths of trust, it lowers the barriers of entry for new peers to participate in the system.

4 Bartering

In a distributed computational economy, we propose bartering as the foundation for simple and robust resource exchange. Higher layers then build on this foundation to provide sophisticated methods of exchange and infrastructure for increased functionality. In the bartering economy, we use SHARP as the core by using its secure resource exchange protocols for peer-to-peer bartering and its mechanisms for discovering and utilizing paths of delegated trust to enable resource sharing across chains of peers. Given SHARP as a basis, we must then address three additional aspects of the bartering economy: bartering strategies, advertising of bartering exchange rates, and path selection algorithms and mechanisms.

4.1 Bartering Strategies

Bartering strategies specify how peers negotiate exchange rates for peering and how peers execute the peering protocol. Negotiating exchange rates involves determining what amount of resources a peer X exchanges with a peer Y as part of the peering and how many such exchanges will occur. We can view an execution of the peering protocol as a sequence of rounds, each of which involves X exchanging some amount of resources with Y. The amount of resources exchanged in a given round is based on an assessment of how valuable a peer's resources are. The number of rounds involves a trade-off between locking in a good exchange rate and being able to dynamically respond to changing conditions. Strategies for determining a parameterization can range from very simple ones (e.g., simple static configuration) to complex ones based on dynamic information (e.g., current load, observed supply/demand).

Execution of the peering protocol depends upon peers correctly and faithfully executing the protocol. In a large-scale distributed system, it is infeasible to assume that all peers can be trusted to behave properly. In a bartering economy, each peering relationship can be viewed as an instance of an iterated Prisoner's Dilemma. In each round, peers play an instance of the Prisoner's Dilemma. Here, we assume that if peers engage in a peering relationship, then remote resources are more valuable than local resources. Otherwise, the peers would not have agreed to participate in the peering in the first place. Let R_{local} denote the value of local resources and R_{remote} the value of remote resources. The reward R for cooperation is thus $R_{remote} - R_{local}$. The punishment M for mutual defection is 0. Finally, the temptation to detect T and the sucker's payoff S are R_{remote} and $-R_{local}$, respectively. Hence, we have the necessary conditions for a Prisoner's Dilemma: $T > R > M > S$.

To encourage large-scale cooperation amongst peers, strategies must be cognizant of defections and respond in an appropriate manner to encourage cooperative behavior. Strategies based on reciprocity and feedback have these properties. Such strategies involve two elements. First, we need mechanisms to detect defections. For example, in the context of the SHARP peering protocol, we can observe a defection in a given round as the absence of a peer's sending of appropriate tickets. Second, we need strategies that determine whether a peer cooperates or defects based on historical information on how the peer has behaved in the past. Throughout society in various arenas (e.g., political, social, military, etc.), we observe situations where large-scale cooperation and reciprocity occur despite the temptation of peers to defect. Effective strategies tend to be optimistic (e.g., largely cooperative and forgiving) and responsive to feedback by observing reciprocity and punishing peers when they defect.

One simple strategy based on reciprocity that has proven to be remarkably robust and effective against a wide range of competing strategies is TIT FOR TAT [2]. TIT FOR TAT is the strategy of beginning with cooperation and, thereafter, doing whatever the other peer did in the previous round. It is simple, encourages cooperation, punishes defection (but is forgiving), and in practice outperforms virtually all competing strategies in a number of situations. Given this, one natural strategy for bootstrapping a computational economy is to start with P2P TIT FOR TAT, where resource exchange in a round is rewarded with resource exchange in the next round and reneging in a given round is punished by reneging in the next round. Given this base strategy, we can then augment it with additional features for additional robustness. For example, we could share P2P TIT FOR TAT history information with friendly peers (and use thresholds or quorums to increase our confidence in the information) to better engage in interactions with peers whom we have limited or no previous history information for. This essentially constitutes a form of robust P2P reputation management.

In environments where the set of peers is fairly static and peers tend to interact with large numbers of other peers, P2P TIT FOR TAT is an appropriate strategy. PlanetLab, our initial target environment, fits this profile. As of May 2003, the PlanetLab testbed consists of 151 nodes, hosted by 69 sites, spanning 13 countries. Target applications for PlanetLab are planetary-scale network services (e.g., content distribution networks, global storage systems, etc.) which require wide geographical coverage for reasons including performance, fault-tolerance, crossing of administrative/political boundaries, and having multiple vantage points of the network. Such applications naturally involve sites peering with many other sites in order to allocate resources for an application. This property, combined with a fairly static set of peers, suggests that P2P TIT FOR TAT will be effective. On the other hand, in environments where the set of peers is large and dynamic, the probability that any two peers interacts decreases. Thus, P2P TIT FOR TAT will be less effective. In such environments, more sophisticated strategies will need to be employed. Recent work in distributed trust and reputation systems is addressing this problem.

4.2 Advertising Exchange Rates

Once peering relationships have been arranged through SHARP, associated bartering parameters then need to be advertised so peers can perform appropriate optimizations. Each set of bartering parameters between a pair of peers essentially specifies an exchange rate. A source wanting to acquire resources at a remote destination might use these exchange rates to minimize the amount of local resources used to acquire remote resources across a path of peering relationships. Making exchange rate information available could be achieved in a number of ways. One natural approach is to annotate SHARP paths of trust advertised through the SRDP protocol with exchange rate information. The necessary exchange rate information would then be available locally to each peer in its local SHARP routing table.

4.3 Path Selection

Once the appropriate resources have been discovered, the next step is acquiring tickets for those resources based on paths of trust established by SHARP. This process involves examining the source's local SHARP routing table and selecting a path of trust for each desired destination's resources. Given desired resources at a particular peer, there could be many paths of trust from the source to a given destination. Along each path of trust is a sequence of ongoing pairwise bartering agreements, each with its own exchange rate. Selecting the optimal trust path is a local optimization problem.

Algorithms to select optimal trust paths will use path and exchange rate information from a local SHARP routing table to optimize for some target metrics. In practice, two examples of target metrics might be the number of peering hops to the destination (analogous to BGP) and the amount of resources relinquished to acquire the remote resources. In an economic setting, the latter is a more natural metric since it minimizes the cost of performing the pairwise bartering.

Once a path of trust is selected, we next need a mechanism to coordinate the pairwise bartering exchanges for a source to exchange resources for the desired destination resources along this path. In doing this, the source might provide tickets representing local resources, or it might provide tickets representing remote resources acquired through its local peering relationships. In either case, the source needs to instruct the peers along the path of trust to perform bartering with specific next-hop peers in reaching the destination. This then allows the source to ultimately acquire the remote resources based on the selected path in a manner similar to source-based IP routing.

5 Conclusion

In this paper, we presented an architecture for distributed computational economies based on peer-to-peer bartering. We described the end-to-end process of describing, discovering, and exchanging distributed resources in a secure and decentralized manner. We then presented an architecture that supports this process based on SHARP, a secure highly available resource peering framework. SHARP provides secure resource exchange protocols that enable peer-to-peer bartering and provides mechanisms for discovering and utilizing paths of delegated trust to enable resource sharing across chains of peers. Given SHARP as a base, we then described the remaining mechanisms needed for peer-to-peer resource bartering and discussed bartering strategies based on reciprocity and feedback. Mirroring the evolutionary path followed by economies in the real world, we believe that a distributed computational economy should be bootstrapped using a bartering economy based on simple and robust resource exchange.

Given peer-to-peer bartering as a base, we envision that human nature will lead to the formation of a power-law distribution [7, 1] in the peering graph. That is, a few peers will establish peering relationships with a large number of other peers. The presence of a large number of such high-degree nodes may lead to a ”CSP” (Computational Service Provider) model with analogy to ISPs. These CSP's may accept cash in return for resource rights. In time, the CSP's may become trusted well enough that they no longer barter using their local resources and instead grant generic ”currency” that can be directly redeemed for resources at all of the sites that they peer with. Through some pre-established trust relationship, sites that receive such generic currency will be able to trust that the generic currency is valid and that resources should be delivered appropriately. All this would take place without involving a middle man.

References

[1] L. Adamic and B. Huberman. Power-law distribution of the world wide web. *Science*, 287(2115a), 2000.

[2] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[3] R. Buyya, J. Giddy, and D. Abramson. A case for economy grid architecture for service-oriented grid computing. In *Proc. of HCW '01*, Apr. 2001.

[4] J. Chase, B. Chun, Y. Fu, S. Schwab, and A. Vahdat. Sharp: An architecture for secure resource peering. Under review.

[5] D. Clark. The design philosophy of the darpa internet protocols. In *Proc. of SIGCOMM '88*, Sep. 1988.

[6] A. Deshpande, S. Nath, P. B. Gibbons, and S. Seshan. Cache-and-query for wide area sensor databases. In *Proc. of SIGMOD '03*, Jun. 2003.

[7] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proc. of SIGCOMM '99*, Aug. 1999.

[8] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proc. of DIALM '02*, 2002.

[9] D. F. Ferguson, C. Nikolau, and Y. Yemini. An economy for flow control in computer networks. In *Proc. of INFOCOM '89*, Apr. 1989.

[10] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *Intl. J of Supercomputer Applications*, 11(2):115–128, 1997.

[11] M. Harren, J. M. Hellerstein, R. Huebsch, B. T. Loo, S. Shenker, and I. Stoica. Complex queries in dht-based peer-to-peer networks. In *Proc. of IPTPS '02*, Mar. 2002.

[12] L. Levy, L. Blumrosen, and N. Nisan. On line markets for distributed object services: the majic system. In *Proc. of USITS '01*, Mar. 2001.

[13] L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A blueprint for introducing disruptive technology into the internet. In *Proc. of HotNets-I*, Oct. 2002.

[14] O. Regev and N. Nisan. The popcorn market – an online market for computational resources. In *Proc. of ICE '98*, Oct. 1998.

[15] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *TOCS*, 2(4):277–288, Nov. 1984.

[16] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and S. Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103–177, Feb. 1992.

[17] R. Wolski, J. Plank, and J. Brevik. G-commerce – building computational marketplaces for the computational grid. Technical Report CS-00-439, The University of Tennessee at Knoxville, Apr. 2000.