UBCA: Utility-Based Clustering Architecture for Peer-to-Peer Systems

Brent Lagesse and Mohan Kumar
Department of Computer Science Engineering
University of Texas at Arlington
{Brent.Lagesse, mkumar}@uta.edu

Abstract

Peer-to-Peer (P2P) systems are currently used in a variety of applications. File sharing applications and ad hoc networking have fueled the usage of these systems. P2P systems generate new challenges in scalability, fairness, and quality of service. Current systems often approach these challenges through incentive-based solutions and structured system design. Incentive-based solutions appeal to the self-interested nature of peers by utilizing payment or penalty to encourage peers to contribute to the system. System design principles, which attempt to improve performance through protocols and system-algorithms, include distributed hash tables and graph-theoretic designs. These approaches have seen some success, but also result in new problems such as overhead costs of authenticity/security for incentives, increased centralization, and decreased ability to handle dynamic peers. We introduce Utility-Based Clustering Architecture, (UBCA) designed to improve quality of service through the use of implicit incentives. UBCA runs on peers and groups them into logical clusters in real time, based on mutual utility gained as a result of the grouping. Simulation studies show with a high confidence that UBCA exhibits improved bandwidth and latency per access.

1 Introduction

Peer-to-Peer (P2P) systems are commonly utilized in software development [6]. Since these systems have direct implications in dynamic and decentralized environments, mobile and distributed computing have formed the foundation for these systems. P2P computing provides several advantages in dynamic systems - resource replication, decentralized control, improved availability, and flexibility [1]. Despite all of its advantages, the P2P system introduces several new problems that must be addressed [1, 2, 6, 8]. Resource discovery introduces overhead costs, even in structured P2P systems. Queries and broadcasts are sent that decrease system performance while not necessarily resulting in improvement in resource quality. As a result of increased overhead, purely decentralized P2P-based systems scale poorly.

Additionally, P2P-based systems also can be dominated by freeloaders that only consume resources, but do not contribute to the system as a whole. These peers add to the system overhead, but fail to contribute to other peers.

The main features of our architecture examine these two problems. We take a different perspective from works that focus on discovering a large quantity of resources and instead focus our efforts on discovering high quality resources. We also look to mitigate the freeloader problem without introducing extra overhead to the system, and instead use implicit incentives of improved performance for peers that share needed resources.

We examine current solutions to problems in P2P-based systems and introduce a Utility-Based Clustering Architecture (UBCA) to improve quality of service through the use of implicit incentives. UBCA runs on peers and groups them into logical clusters during execution time based on mutual utility gained as a result of the grouping. UBCA utilizes utility theory in order to reduce overhead costs and select the best resources to access. One of the key features of UBCA is that it improves the performance of the application while allowing the underlying P2P system to maintain its characteristic features (ie, overlaying Gnutella with UBCA still allows Gnutella to be highly decentralized). Each peer is abstracted into an agent that consumes and provides a set of resources. By performing this abstraction, peers can be clustered together to satisfy each others' requirements best so that there is less need for a peer to request help from outside the group.

In order to provide a concrete example, we explore the application of UBCA to an academic content distribution network (CDN). CDNs are systems which typically reside at key points in the network infrastructure in order to transparently satisfy user requests [1, 2]. P2P-based CDNs accomplish this by moving content between peers in the system in order to more quickly satisfy user requests and re-

duce the bandwidth associated with satisfying the request from a provider "further" away. This can be done by intelligently directing requests to peers near the originating request that can do the best job satisfying the request.

One situation where this application would arise is a peer to peer system of college students living and interacting together. This system consists of students who have accessed and cached academic video and audio streams from their courses. In this system, there are several classifications of students. Suppose, we have Computer Science, Mechanical Engineering, and Philosophy students and each classification of student shares cached streams most similar to each other; however, due to their enrollment in overlapping classes (i.e., Calculus or English), they also share files with students outside of their classification.

The application of UBCA described in this paper will cluster the most similar students together with consideration of the accessibility of those resources. Obviously, each particular major would tend to group together most strongly, but Computer Science students might also group together with Mechanical Engineering students who are very accessible to them because of their similarities in the College of Engineering. Likewise, the engineering students have minimal affinity toward Philosophy students (but a non-zero amount due to their similarities in general education courses). Clusterings would be based on this shared affinity and produce clusters of students which can easily access each other's cached files to minimize the load on the university's servers providing the original files.

Simulation studies reveal that UBCA exhibits positive results in terms of reduced overhead, increased bandwidth per resource access and decreased latency per resource access. UBCA is expected to be used in and improve applications in pervasive, mobile, and distributed computing. The primary contribution of this work is an architecture that results in a reduction of overhead costs of the P2P system, increases the performance of of individual accesses, and is adaptable to specific application needs while allowing the underlying P2P system to maintain its characteristics.

2 Related Work

Since Gnutella was released and shown to scale poorly, researchers and developers have worked to create more scalable P2P systems. Two recent generations of P2P systems include distributed hash table (DHT) based approaches and supernode-based approaches. A new generation of privacy and security enhancing P2P systems are currently being developed; however, we will not review these since the focus of this work is scalability and performance.

DHT-based solutions utilize distributed hash tables to implement a lookup operation for needed resources. The lookup operation requires *O(log n)* time as compared to Gnutella which requires *O(n)* time for its search[4]. Despite the scalability improvement, the DHT approach has many shortcomings. First, it requires a structure and controlled system in order to operate most efficiently, which places a limit on the dynamic nature of P2P systems. The downfall is evident when considering that for each node that fails or exits, the DHT must recover by discovering the failure and repairing lost information. Second, DHTs require exact match searching. While there are ongoing research efforts [4] to support keyword searches, there is no currently available method that works as well as Gnutella. Overall, DHTs have a place for systems that can be controlled, but are insufficient for systems that need to support highly dynamic peers and a purely decentralized environment.

Supernode-based systems utilize specially selected peers in order to index the resources available in a small local group. Query messages are only flooded from supernode to supernode, rather than by all nodes. As a result, the overhead traffic in the system is reduced. The downside to this approach is that it limits the decentralization of the system [11]. Furthermore, the system must include mechanisms which add to the computation overhead in order to find peers that are capable, willing, and trusted to serve as supernodes[11]. As with the DHT approach, supernode approaches compromise the dynamic and decentralized nature of a P2P system such as that found in Gnutella.

Since the aforementioned approaches to create scalable P2P systems do not meet our needs of a low-overhead, decentralized P2P system, we have designed UBCA, a utility-based clustering architecture. UBCA maintains decentralization and supports dynamic environments while providing a scalable and quality-enhanced P2P system.

3 Design

3.1 Goals

At a high level, the objective of our work is to form dynamic communities of peers. The main design goals of UBCA are to increase quality of service through,

- Enhance availability and quality of resources
- Encourage resource sharing in the P2P system
- Application adaptivity
- Maintain underlying system's structure and decentralization

The accomplishment of these goals will create a more efficient and useful architecture for designing applications that rely on P2P. In the results section, we show strong evidence from simulations that UBCA reduces the overhead communications in the P2P system and increases the performance

over that of the original system. Furthermore, we show greater increases in performance caused by UBCA when we allow the application to adapt to the needs of high bandwidth (ie, file downloading system) or low latency (ie, real-time control system). In this section, we also investigate UBCA's ability to overlay a P2P system without changing the system's inherent structure and decentralization properties.

3.1.1 Overheads

Communication overheads result in a decrease in system and individual peer performance. In P2P systems, overhead communications are largely caused by resource discovery. While the individual messages are usually small, the quantity of messages sent out in order to obtain the location of a useful resource can quickly become overwhelming. Furthermore, for each resource found, a query hit message has to be sent back to convey the location of the resource.

For example, a basic decentralized P2P system, Gnutella scales poorly as a result of excess overhead communication. It relies on message forwarding to connected peers in order to discover resources, so the number of requests can potentially increase exponentially [8]. One solution to deter this type of system overload is to set a maximum hop count on query messages [6]. The downside of this approach is that it inhibits a peer's ability to find resources outside of the range of the maximum hop count.

3.1.2 Performance

System performance is measured in two ways. The first is the performance of the system as a whole, and the second is application-tuned performance (ie, optimized for bandwidth). Performance was tested based on the average access bandwidth and access latency. Since the goal of most uses of discovery in a P2P system is to actually access a good resource and not just find many resources, we do not consider the number of resources found to be an important performance metric. In fact, due to the overhead mentioned above, we consider finding too many resources to be detrimental to the system instead of just a small set of good resources.

3.1.3 Application Adaptivity

Incentives are an approach to encouraging self-interested peers to share resources [3, 5]. Most systems with incentives utilize a currency type approach such as karma in Kazaa. The primary problem with incentives is that they introduce overheads involved with securing transactions and preventing counterfeiting. In UBCA, incentives are currency-less. These incentives are implicit in the clustering of the peers. Since the group is based on mutually

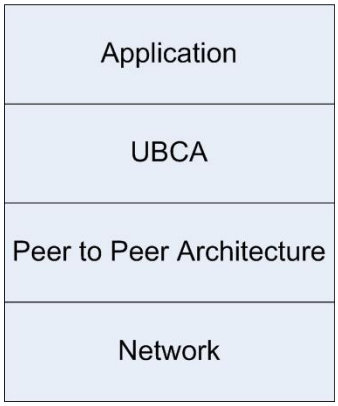

Figure 1. UBCA Network Architecture

increased utility, each peer must contribute sufficiently, and if a peer is not part of a group, their performance will drop back to the levels of non-UBCA systems. This incentive results in increased contributions by rational self-interested peers. These incentives cause performance, resource quality, and resource quantity within clusters to be tuned to the needs of the application. For instance, in our CDN example, the application needs are 1) high bandwidth to download video and audio, 2) and high quality (resolution, bitrate, etc.) files. These two considerations would be heavily weighted and clusters would form based primarily on those parameters.

3.2 P2P Characteristics

Each P2P implementation has an architecture and underlying communication protocols which define it on some range of its type of centralization and structure [1]. UBCA does this by overlaying the system that drives communications. It adds an optional cluster of peers above the system for quick access to resources, but still uses the underlying system it is running on if the resource is unavailable in the group.

3.3 Architecture

As shown in Figure 1, the UBCA network architecture lies between the application and the underlying P2P system. In a UBCA-enabled P2P system the UBCA layer is completely decentralized and distributed to the peers utilizing it. Therefore, there is no centralized control mechanism that would interfere with the underlying P2P system. Additionally, the application layer only has to convey its preferences to the UBCA layer on each peer in order to customize what characteristics it bases group formation on. Not all peers in

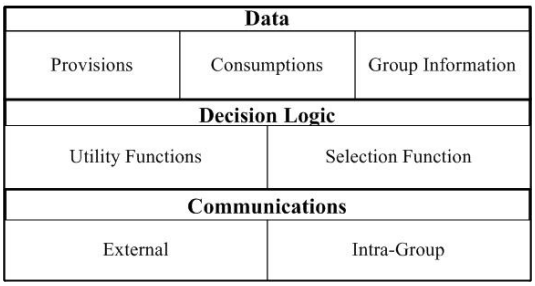

Figure 2. UBCA Peer Architecture

a system need to utilize the UBCA layer. In fact, a peer that does not implement a UBCA layer is essentially the same as a peer that does not derive additional utility from joining a group, so the UBCA peers are not negatively affected by peers that do not participate in clustering. Non-participating peers just experience performance at the level expected of that particular P2P system.

The UBCA peer architecture consists of three parts as shown in Figure 2: data, decision logic, and communications. The data contains all the information necessary to make clustering decisions. The decision logic provides the utility functions for clustering and resource selection decisions based on the data, and the communications convey the decisions to other peers.

3.3.1 Data

The UBCA data structures are used to represent the resource production and consumption of both the individual peer and the collective resources of the group. A resource record contains the type of resource, the expected latency to access the resource, the expected bandwidth to the resource, the qualities of the resource, and the address of the resource.

The resource's expected latency is computed with a historical averaging function. While any historical averaging function will work, we find that an approach similar to TCP timeout by weighting 75% of the value on previous measurements and 25% of the current measurement does well to adapt to changes in latency, but without overreacting to a single sample. By allowing any historical averaging function, we allow the peer to adapt or customize its reaction to fluctuations based on volatility of the system. For example, latency values in a mobile network are very volatile and the past does less to predict the future than in a stationary ethernet-based network. The approach to bandwidth parallels that of latency. Transmission rates are taken empirically during communication and the value of the weighted average is stored. The type of resource is the system's identifier for the resource. The quality of the resource is slightly less obvious. The resource quality is the list of relevant attributes of a resource that could cause its quality to vary.

3.4 Decision Logic

Utility is the defining metric for forming a group. Utility is defined as the sum of the benefits minus the sum of the costs to provide each resource as shown in equation 1. When the utility is greater than 0 for a peer, it implies that the formation of a group is beneficial for those involved. The metrics constraining the value of utility and selection are based on bandwidth (BW), latency (Lat), memory (Mem), and CPU cycles [9]. In each of our equations we have weights (the *w* values in the equations) provided, so applications can adapt the utility value to better suit their needs. We also calculate an intermediate *Q* value which is how a peer perceives the benefit it will derive from a particular resource (Res).

$$Utility \equiv \sum Benefit - \sum CostP \tag{1}$$

$$Q(Res) \equiv w1 \times Quality(Res) + w2 \times Quantity(Res) \tag{2}$$

$$Benefit \equiv \sum w1 \times Q(Res) - CostC(Res) \tag{3}$$

$$CostC(Res) \equiv w1\frac{ResLat}{MeanLat} + w2\frac{MeanBW}{ResBW} \tag{4}$$

$$CostP(Res) \equiv w1\frac{BW\_Req}{BW\_Av} + w2\frac{Mem\_Req}{Mem\_Av} + w3\frac{CPU\_Req}{CPU\_Av} \tag{5}$$

$$SelectionValue \equiv w1 \times Q(Res) - CostC(Res) \tag{6}$$

Benefit, which is represented in (3), is given by the sum of the weighted Q value (perceived utility of the resource) of all the consumed resources minus the cost of consuming those resources.

The cost to consume a resource is the sum of the weighted ratio of the resource's latency to the average latency for that resource and the weighted ratio of the average bandwidth of that resource to the specific resource's bandwidth. The consideration of cost to consume creates a situation in which only the resources that are effectively more accessible to the consumer are selected. The cost to consume a resource is given by (4).

The cost of joining a group is determined by analyzing how taxing it is on the peer to provide that resource. The cost to provide is the weighted sum of the what percentage of available bandwidth, CPU, and memory providing the resource would consume. Weights are determined by the application's needs.

Selection entails determining which resource is optimal to request. The selection value reveals the highest utility instance of a resource with the lowest cost to access. The selection value is given by (6).

3.4.1 Communications

Communications utilize and extend the Gnutella communication protocol [10]. The extension contains two major portions. The first portion is used to establish a group. The second is intra-group communication.

Upon receiving a successful query hit, the peer analyzes the peer generating the query hit with its utility function. If it is beneficial, then the peer initiates a group request by sending a group request message with the current group's set of resource provisions and consumptions. The peer then moves into a response-wait state. Upon receiving this invitation, the peer that supplied the query hit analyzes the provisions and consumptions with its utility function to determine if it accepts. If joining the group is beneficial to the peer, it then returns a response to the request containing its provisions and consumptions and enters a response-wait state. The original querying peer then performs a full utility analysis of the provisions and consumptions and decides if the peer is acceptable. If so, the peer is added to the group. Upon this acceptance, the peer will return a response to the waiting peer informing it of the decision and the peer will accept the group's provisions and consumptions. After each peer has been added, they return to normal state.

Intra-group communication permits either lazy or active communication depending on the needs of the application. The communications that take place within the group are utilized to maintain the link/resource state of the group. Each peer capable of permitting another peer into the group maintains a set of resources of each peer's consumption and production. In order to update these lists, when a new peer is added, the peer that admits it broadcasts a message to the group announcing the new addition of resources provided and consumed. Furthermore, when a peer departs from a group, that peer broadcasts an exit message for the other peers to remove its record. Since peer-to-peer systems are often ad-hoc, mobile, and dynamic[6], peers are likely to exit without warning. If a peer departs the group for any reason without sending the exit message, any peer that fails to access the peer broadcasts a message to warn other of the potential departure. When a peer receives a sufficient number of these messages (by default, one) the peer removes the departed peer's record from the data structure. This property allows for quick healing and provides the ability to recover from a peer that lies about its resources or changes without notifying the group.

4 Results

In this section, we present the results of the simulation in order to demonstrate, key strengths in the architecture in addition to its ability to satisfy our goals established previously. Unless otherwise noted, all simulations were performed with the following parameters:

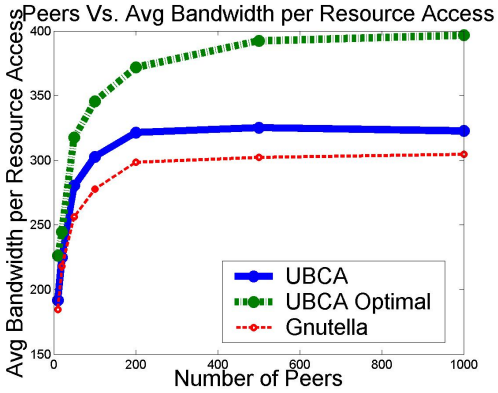

Figure 3. Peers Vs. Avg Bandwidth

- 100 classes of resources
- 5 initial connections at bootstrap
- Normally distributed latency
- Normally distributed bandwidth
- Resources consumed to provided ratio of 1 (25 to 25)
- No Peer Classifications
- The simulations were run for 10, 20, 50, 100, 200, 500, and 1000 peers and graphed values were extrapolated from those results

Simulations were done for Gnutella, UBCA, and UBCA-Optimal. UBCA-Optimal gives all of the weight in the cost to consume equation (4) to the metric being tested. UBCA-Optimal for Figure 3 places all weight on bandwidth and no weight on latency. UBCA-Optimal in Figure 4 places all weight on latency and none on bandwidth. Measurements were taken from 500 trials for each population size of the simulation. As a result, we determined Z values for each distribution at the 1000 peer level. The difference in means resulted in Z values yielding almost no belief that the Gnutella and UBCA statistics were drawn from the same distribution. The smallest Z value came from the UBCA Latency sample. The resulting Z value was 4.67, which produces a probability of about 0.00015% that the samples came from the same distribution.

The plot in Figure 3 compares the average bandwidth per resources access for UBCA, UBCA-Optimal for bandwidth, and standard Gnutella. The UBCA plot is the simulation results from running UBCA with uniform weights on bandwidth and latency. The UBCA-Optimal plot places all weight on bandwidth.

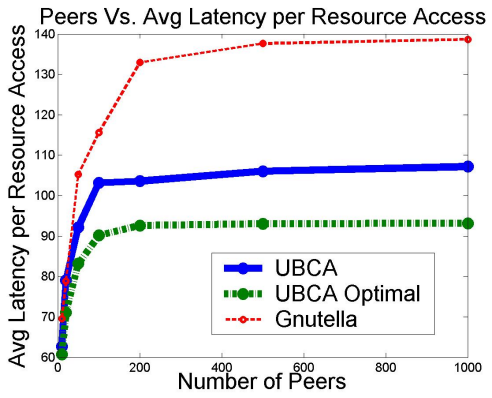

Figure 4. Peers Vs. Avg Latency

The plot in Figure 4 compares the average bandwidth per resources access for UBCA, UBCA-Optimal for latency, and standard Gnutella. The UBCA plot is the simulation results from running UBCA with uniform weights on bandwidth and latency. The UBCA-Optimal plot places all weight on latency.

In Figure 3, a failed access (resource not found) is added into the average bandwidth as an access with a bandwidth of 0 KB/s. In Figure 4, a failed access (resource not found) is added into the average latency as an access with which has timed out waiting for a response (by default 3 seconds).

Table 1 shows the energy consumption of peers in UBCA and Gnutella protocols based on a wireless card that consumes twice as much power to transmit as receive. The values are measured based on one unit of energy required to receive one kilobyte of data. It is evident that the UBCA outperforms Gnutella significantly in communication energy cost. One of the key points in this data is the decrease in average total energy cost per peer as the number of peers increase, as opposed to that in the case of Gnutella.

Table 1: Effective Energy Consumed per Peer

	Sending Peers		Hit Peers		Intermediate Peers		Total	
Peers	**UBCA**	Gnutella	**UBCA**	Gnutella	**UBCA**	Gnutella	**UBCA**	Gnutella
10	**62.50**	75.43	**2.38**	4.03	**97.05**	120.07	**195.13**	241.21
20	**61.86**	76.73	**2.40**	3.96	**92.86**	123.53	**188.38**	247.29
50	**61.18**	77.35	**2.52**	3.91	**84.78**	123.82	**175.90**	248.14
100	**59.92**	78.17	**2.54**	3.81	**81.20**	124.64	**169.61**	249.87
200	**60.16**	77.73	**2.60**	3.76	**79.69**	124.42	**167.58**	249.16
500	**58.58**	77.79	**2.61**	3.76	**77.73**	124.39	**163.46**	249.16
1000	**58.67**	77.72	**2.61**	3.73	**77.54**	124.27	**163.25**	248.89

5 Conclusions

UBCA provides an improvement to P2P systems while maintaining much of the underlying system's characteristics such as with Gnutella. UBCA takes a unique approach to improving P2P systems by clustering peers together based on mutual utility derived from the clustering. UBCA meets its design goals by improving scalability, increasing performance, and increasing resource availability/accessibility.

UBCA has been shown in simulations to increase bandwidth per access, reduce latency per access, and reduce the overhead costs of system operation over Gnutella. Furthermore, UBCA uses the increased performance as a currencyless incentive for peers to share more resources and not free-load. The implementation of UBCA will encourage the replication and access of files in distribution networks.

The next step in this line of research is to implement UBCA in the applications mentioned previously and examine their empirical performance. Another area of work to consider is to examine application specific optimizations of UBCA such as defining the proper weights, or dynamic mechanisms for assigning weights for an application such as streaming multimedia in a MANET. There is also potential for future work in examining implementations of the architecture in mobile environments and testing the result of those implementations. Finally, the issue of trust needs to be considered. As we mentioned previously, UBCA has an inherent ability to heal from peers that lie about their abilities, but in many applications, such as the medical environment mentioned in the applications section, we will want to only form groups of trusted peers due to a secure or sensitive nature of the information being distributed.

References

[1] Androutsellis-Theotokis, S. and Spinellis, D. A Survey of Peer-to-Peer Content Distribution Technologies. ACM Computing Surveys, Vol. 36, No. 4, December 2004, pp. 335-371.

[2] Biddle, P., England, P., Peinado, M., Willman, B. The Darknet and the Future of Content Distribution. ACM Workshop on Digital Rights Management, 2002.

[3] Buragohain, C., Agrawal, D., Suri, . A Game Theoretic Framework for Incentives in P2P Systems. Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P'03). 2003.

[4] Chawathe, Y., Ratnasamy, S., Breslau, L., et al. Making Gnutella-like P2P Systems Scalable. SIGCOMM, 2003.

[5] Golle, P., Leyton-Brown, K., Mironov, I., Lillibridge, M. Incentives for Sharing in Peer-to-Peer Networks. In Proc. of the Third ACM Conference on Electronic Commerce, Tampa, Florida, USA, October 2001.

[6] Krishnan, N. The JXTA Solution to P2P. http://www.javaworld.com/javaworld/jw-10-2001/jw-1019-jxta.html. October 19, 2001.

[7] Kumar, M., Shirazi, B., Das, S. K., et al. Pervasive Information Communities Organization PICO: A Middleware Framework for Pervasive Computing, IEEE Pervasive Computing, July-September 2003, pp. 72-79.

[8] Ritter, J. Why Gnutella Can't Scale. No, Really. Unpublished. 2001.

[9] Schoder, D., Fischbach, K., Schmitt, C. Peer to Peer Computing: The Evolution of a Disruptive Technology. Chapter 1: Core Concepts in Peer-to-Peer Networking. Idea Group Publishing, 2005.

[10] The Gnutella Protocol Specification v0.4.

[11] Zhiyong Xu, Yiming Hu: SBARC: A Supernode Based Peer-to-Peer File Sharing System. ISCC 2003: 1053-1058