Scalable Leader Selection in Peer-to-Peer Overlay Networks

Virginia Lo, Dayi Zhou, Yuhong Liu, Chris GauthierDickey, and Jun Li
{lo|dayizhou|liuyh|chrisg|lijun} @cs.uoregon.edu
Network Research Group
University of Oregon

A common problem found in a wide variety of peer-to-peer applications involves selection of a subset of the peers to serve a special role. The specially selected peers must be well-dispersed throughout the peer-to-peer overlay network, and must typically fulfill additional requirements such as load balance, resources, access, and fault tolerance. While similar to the classic dominating set and p-centers problems from graph theory, the leader selection problem *must meet the additional challenge of finding leaders within a huge and dynamically changing network in which neither the node characteristics nor the network topology are known a priori.*

In this paper, we define the leader selection problem and discuss its requirements and challenges, driven by a wide range of peer-to-peer and networking applications. We describe three generic leader selection protocols we have developed for peer-to-peer environments: a label-based scheme for structured overlay networks, a distributed protocol for coordinate-based overlay networks, and a negotiation protocol for unstructured overlays. We believe an integrated approach to the leader selection problem across a wide range of applications can benefit the peer-to-peer community through cross-fertilization of ideas and sharing of protocols.

I. Introduction

We have identified a fundamental problem, which we call the *leader selection problem*, which occurs across a spectrum of peer-to-peer applications, including file sharing systems, distributed hash tables (DHTs), publish/subscribe architectures, and peer-to-peer cycle sharing systems. The leader selection problem also shows up in the fields of sensor networks, adhoc wireless networks, and peer-based Grid computing. Leader selection involves the selection of a subset of the peers in a large scale, dynamic network to serve a distinguished role. The specially selected peers must be *well-dispersed* throughout the network, and must typically fulfill additional requirements such as load balance, resources, access, and fault tolerance.

The leader selection problem is highly challenging because in the peer-to-peer environment, a large number of leaders must be selected from a huge and dynamically changing network in which neither the node characteristics nor the network topology are known a priori. Thus, simple strategies such as random leader selection don't work. Leader selection is more complex than classic *dominating set* and *p-centers* from graph theory, known to be NP-hard problems, because it must respond to dynamic joins and leaves (churn), operate among potentially malicious nodes, and function in an environment that is highly heterogeneous.

Leader selection shows up in many peer-to-peer and networking applications. For example, in peer-to-peer file sharing systems, such as Kazaa [2] and Gnutella [1], negotiation protocols were developed for the designation of qualified leaders (supernodes and ultrapeers) to serve the ordinary peers for scalable content discovery. Peer-to-peer infrastructure services require methods for the judicious placement of monitors/landmarks/beacons throughout the physical network for purposes of measurement, positioning, or routing [26], [28], [25]. In adhoc wireless networks, connectivity under highly dynamic conditions is achieved by identifying a subset of the nodes to serve as bridging nodes. This subset is formed using a distributed dominating set protocol such as in [42], [40], [10], [34] so that every node is within broadcast range of a bridging node. Within sensor networks, leaders are selected for the purpose of data aggregation under the conditions that they are well-distributed among the sensors and also have sufficient remaining battery life [8].

In this paper, we demonstrate that many peer-to-peer and networking applications are seeking solutions to variations on the same fundamental problem. We believe an integrated approach to the leader selection problem, built on strong graph theoretic foundations and guided by realistic applications, can yield benefits for the field of peer-to-peer computing and beyond.

We first define a general model for the leader selection problem, describing its key requirements and challenges. We show how the leader selection problem is related to dominating set and p-centers problems, and describe how the leader selection problem is instantiated in several well-known peer-to-peer applications. We then present three general purpose leader selection protocols we have developed for the three basic types of overlay networks used by peer-to-peer applications: structured overlay networks, coordinate-based overlay networks, and unstructured overlay networks. We illustrate how each protocol can be be extended for use within a variety of specific peer-to-peer applications.

The three leader selection protocols we have developed are:

- **SOLE**: a label-based leader selection protocol for applications built on a structured overlay network such as CAN [28], Chord [38] or Pastry [30]. SOLE exploits the regular node labeling schemes used within structured overlays to distribute leaders evenly throughout the overlay, to ensure fast access from non-leaders to leaders, and to maintain a fixed leader to non-leader ratio as the overlay grows and shrinks dynamically.
- **PoPCorn**: a protocol for applications that use a coordinate-based overlay network using systems such as

Supported in part by NSF 9977524 and an NSF Graduate Research Fellowship

Vivaldi [9] or GNP [25]. This protocol selects a fixed number of leaders and disperses them evenly with respect to the topology of the overlay network using a a repulsion model.

- $\mathbf{H_20}$: an advertisement-based protocol deployable in an unstructured overlay network. This protocol can be used to find *qualified* leaders from among the peers such that every ordinary peer is within k hops of C leaders (multiple distance domination).

II. Problem Definition and Background

We first describe the unique requirements and challenges of the leader selection problem in the context of peer-to-peer systems. We then describe dominating set and p-centers problems from graph theory that form the theoretical underpinnings of the leader selection problem. To conclude this section, we survey examples of applications from peer-to-peer computing and other networking domains that call for scalable leader selection protocols.

A. The Leader Selection Problem

We broadly define the leader selection problem as that of selecting some subset of the peers in a large scale peer-to-peer overlay network to take a special role, with the designated leader nodes providing service to the non-leaders. A potentially large number of leaders must be selected from an unknown, large scale, and dynamically changing overlay network. First, we describe key topological distribution criteria that the leaders must fulfill relative to the non-leaders. Second, we enumerate characteristics of peer-to-peer networks that make leader selection difficult.

Leader Distribution Criteria. The leaders must be distributed throughout the peer-to-peer overlay network in a topologically sensitive way to meet one or more of the distribution criteria listed below.

- *access*: non-leader nodes must have low latency access to one or more leader nodes. Access can be measured in hop counts or delay. For example, peers in gnutella must be close to ultrapeers in order to reduce network traffic and speed up content discovery.
- *dispersal*: leader nodes must be evenly distributed throughout the overlay network, i.e. leaders should not be clustered within only a few subregions of the overlay. Dispersal criteria apply to monitoring applications and to placement of landmark nodes or beacon nodes.
- *proportion*: a pre-specified global ratio of leaders to non-leaders must be maintained to meet application-specific performance requirements.
- *load balance*: leader nodes should not serve more than k non-leaders, where k can be configured locally according to the resource capability of each leader.

Note that these criteria are inter-related in that specification of requirements for one may impact another, or a given application may require multiple criteria, e.g. it may specify that the leaders must be both proportional and load balanced.

Peer-to-Peer Factors. The design of leader selection protocols within a peer-to-peer environment is challenging because in addition to fulfilling the distribution requirements outlined above, they must also deal with factors arising from the underlying nature of large scale, highly dynamic systems. These criteria include *heterogeneity*, *adaptability*, *fault tolerance* and *resiliency*, and *security*.

- *heterogeneity:* Current large-scale peer-based systems consist of a large number of heterogeneous nodes with differing hardware and software resources. A node in a peer-based system might not be elibible to be a leader unless it meets certain minimum qualifications. These qualifications include *resources*, such as CPU power, disk or memory space, or battery life; *stability*, such as uptime or fault tolerance; *communication*, such as bandwidth or fan-out; and *safety*, such as trust or security.
- *adaptability to churn:* Peer-to-peer environments are extremely dynamic. Nodes join and leave rapidly. Leader selection protocols must be able to handle this churn and respond quickly, especially when leader nodes leave the system. Leader selection protocols must also be adaptive to dynamic changes in network traffic and overlay topology.
- *resilience and fault tolerance:* Leader selection protocols may need to be fault-tolerant, e.g. when a given leader dies, other leaders can quickly take over its functions or a new leader can be quickly selected.
- *security:* Leader nodes may be vulnerable to denial of service attacks, malicious leaders can disrupt the system by failing to forward messages or by giving out wrong information.

B. Dominating sets, p-centers, and leader election

A wealth of research in graph theory, location theory, and distributed computing provides a formal foundation for the leader selection problem. Below we state the key classic problems that have been studied as well as some extensions that are particularly applicable to peer-to-peer computing.

The basic *dominating set problem* is the problem of finding a minimal subset of the vertices in graph G, called the dominator set, such that every node is either a dominator or adjacent to a dominator. Dominating set problems and algorithms are described thoroughly in [16]. Most versions are NP-hard.

The following variations on the basic dominating set problem are useful for peer-to-peer applications since they address access, dispersal, proportion, load balance, security, and heterogeneity of peers.

- *Distance domination* seeks to find a minimum size d-dominating set such that the distance from an arbitrary node to a dominator is $\leq$ d.
- *Multiple (c,d)-domination* requires that every peer be within distance d of c dominators. This is possible in every connected graph having more than c vertices when $d \geq c$ [21]. We have developed algorithms for solving this optimization problem (minimizing the size of the set) on certain regular networks [6] and for unstructured overlay networks [22].

- *Colored domination* presumes that each node in graph G has an associated color from the set $c_1, c_2, ...c_n$. A dominating set of color c_i is one in which the dominators are all of that color. Colored domination can be used to model heterogeneous networks in which only certain nodes are qualified to be dominators. Variations of multiple, colored, and distance domination can be posed, e.g., every ordinary peer must be within k hops of a dominator of each color.
- A *secure dominating set* is a subset of vertices S such that for any vertex v not in S there exists a neighbor u of v in S such that, if we add v to S and remove u from S, we get another secure dominating set S [7]. This allows leaders to move among certain sites in the network while maintaining a secure dominating set as the leaders move.
- A *global defensive alliance* is a variant of dominating set where the set S is such that every vertex v not in S has a neighbor in S and every vertex v in S has a majority of its neighbors in S [17]. A *global offensive alliance* is such that every vertex not in S has a majority of its neighbors in S. These sets take a more combative perspective on security, perhaps allowing development of a protocol to squelch distributed attacks on leaders.
- We recently investigated the notion of *k-defensive domination*. A *k-attack* on a graph G consists of a selection of k distinct vertices of V, which vertices are said to be *under attack*. A k-attack A, represented as the vertices $\{a_1, ...a_k\}$, can be *countered* by a subset of vertices X iff there exists an assignment of the k distinct members of X to the k members of A, represented as $\{(a_1, x_1), ..., (a_k, x_k)\}$, such that either $a_i = x_i$ or (a_i, x_i) is an edge of G, for all i, $1 \leq i \leq k$. In other words, an attack at a vertex must be uniquely countered by itself or by a neighbor in X. Given a graph G, a subset D_k of V is a *k-defensive dominating set* of G if and only if it can counter any k-attack in G. We provided solutions for minimum k-defensive dominating sets on certain classes of graphs, including an efficient algorithm on trees [12].

The p-center problem is related to the dominating set problem and is applicable when placing a *fixed* number of leaders in a network. Algorithms and variations on this NP-hard discrete location problem are found in [15].

- The *p-center problem* is the problem of finding a subset of p vertices in a graph G, called centers, to minimize the maximum (or total) distance between a non-center node and its nearest center.
- *Colored p-centers* can be used in a colored graph for the problem of finding a subset of p_i vertices of color c_i to minimize the above distance criteria.

Finally, we note that the scalable leader selection problem is also related to the leader election problem from distributed computing. Leader election differs from leader selection in that the former assumes all nodes vote (directly or indirectly) on the choice of each leader. Leader election algorithms are not scalable because they require broadcasting or passing a token to all nodes. The best known *leader election* protocols electing a leader (typically the node with highest ID number) under various fault tolerant scenarios, such as Ring, Bully, etc. [24], [4].

Heuristic algorithms developed for these classic problems have been utilized in the field of networking, but their applicability is usually limited to smaller scale, static networks. For the most part, they involve centralized algorithms or high message passing overhead [39], [14], [24]. These algorithms were not designed for large scale peer-to-peer networks that exhibit a high degree of churn and that are dynamically heterogeneous. Hochbaum [18] studied the p-center problem in a network with changing network distances within a fixed topology. Leader selection problems in peer-to-peer environments need to consider not only the changing network distance but also the changing overlay network topology.

C. Ultrapeers, landmarks, and rendezvous points

The best known example of leader selection in a peer-to-peer application is the *gnutella* [1], [3] protocol for selection of ultrapeers—peers with sufficient bandwidth and processing power to serve as proxies for other peers. The use of ultrapeers decreases the amount of network traffic generated by Gnutella's flooding-based search mechanism. Any peer can select itself as an ultrapeer if it meets the following criteria: it has been up for at least 5 minutes, has high bandwidth, sufficient processing power, runs an OS that can handle a large number of simultaneous TCP connections, and is not firewalled/NATed. The ultrapeer selection protocol dynamically adjusts the number of leaders as follows: if a leaf cannot find an ultrapeer with free slots, it can promote itself to be an ultrapeer. Ultrapeers also downgrade themselves when they no longer serve any leaf nodes, or through negotiation with nearby peers.

Thus, gnutella's leader selection (ultrapeer selection) protocol loosely meets the distribution criteria defined above and adapts dynamically to users' needs in a best effort fashion. One of gnutella's goals is to achieve a certain ratio of ultrapeers to leaf nodes, but currently there is no way to control or measure this ratio. Security is not addressed in gnutella.

A canonical leader selection problem that has shown up as a bootstrapping step in a number of prominent peer-to-peer systems is the designation of landmark nodes in the Internet. The CAN [29] structured overlay is constructed using a *binning* technique in which carefully placed landmarks are used to construct a topologically sensitive overlay. Also, in GNP [25], a coordinate system is built using landmark nodes. Studies have shown that the accuracy of the GNP coordinates is highly sensitive to the location of the landmark nodes. Choosing landmark nodes boils down to leader selection for a fixed, small number of leaders that are well-dispersed in the physical network (p-centers). The notion of evenly distributed or well-dispersed can be captured by metrics based on pairwise distances, N-medians and N-cluster medians [25], or distances from a centroid. Current approaches use manual placement of landmarks or centralized dominating set algorithms [39].

Examples from outside peer-to-peer computing illustrate the wide scope of the leader selection problem. In adhoc wireless networks, a distributed dominating set protocol is used to select nodes to serve as intermediary routing nodes to bridge the distance between wireless nodes that are out of broadcast range of each other [41]. The IDMaps [26] distance map of the Internet used a centralized p-centers algorithm to place *Tracer* monitors and *Boxes* throughout a known network topology. Sensor networks utilize specially placed monitor nodes as rendezvous points to collect and aggregate sensor data [11]. This protocol focuses on access and battery life as its primary criteria.

Table I lists examples of the leader selection problem for a range of applications, showing the commonality and diversity in leader qualifications and leader distribution requirements.

Projects/papers	Leader Qualifications	Distribution Criteria
Distance Estimation		
CAN[29], GNP[25], IDMaps (Tracer/Box)[26]	Minimal	Inter-landmark distance,N-medians,N-cluster-median, Fixed # of landmarks
Content Location		
Gnutella[1], [3], Kazaa[2]	Operating system, Bandwidth, Uptime, Memory/CPU	Maintain # of ultra-peer in proportion to leaf nodes
Routing		
Expressway[20], Sequoia (router)[19]	Highly available, Fan-out, Bandwidth, Trust	Close to network access points, k hops to c leaders, Disjoint paths
Data Aggregation		
Sensor networks[11]	Battery life	Within signal radius
Rendezvous Point		
Sequoia (RP)[19], Ferreira[13]	Bandwidth, Uptime, Memory/CPU, Trust	Easily accessible, Distribution fits density of non-leaders
Monitors (point-of-presence)		
IDMaps [27], NETI@home[36]	Minimal	Fixed # of monitors, Sensitive to network topology

TABLE I
LEADER SELECTION EXAMPLES

III. THREE NEW LEADER SELECTION PROTOCOLS

Heuristic algorithms for classical problems such as dominating set and p-center cannot be applied to large scale, dynamic peer-to-peer environments. Standard techniques such as random selection of peers, or flooding-based search for suitable leaders do not work. Nodes selected by a random strategy may not be qualify to be leaders and may be poorly dispersed. Flooding-based search for a large number of leaders is not scalable and does not adapt well to dynamic changes in the network.

For these reasons, we introduce three new leader selection protocols for the three basic types of overlay networks: structured, coordinate-based, and unstructured. These protocols take advantage of properties of the respective overlay types to better address the leader selection. Our goal is to describe a variety of protocols for leader selection and how they address distribution criteria. Performance evaluation of these protocols and of their use in specific peer-to-peer applications is part of our ongoing research agenda, but outside scope of this paper.

A. SOLE: Leader selection in structured overlay networks

SOLE (Structured Overlay Leader sElection) is designed to select a group of leaders in a structured overlay network, with a goal of keeping the leader to non-leader ratio stable as peers join and leave the overlay. SOLE also maintains low access from non-leader nodes to leader nodes and provides load balancing for each leader. In the last part of this section, we discuss how SOLE addresses resource heterogeneity and how SOLE copes with the departure or failure of leader nodes.

A DHT (Distributed Hash Table) built on a structured overlay network such as CAN [28], Chord [37], and Pastry [30] makes use of a symmetric, regular node label space, in which each physical node owns a virtual subspace in the overlay. In these structured overlay networks, a compact *node label expression* can encode a (large) collection of virtual nodes.

SOLE exploits this notation and uses a node label expression to designate a subset of the virtual node label space as leaders. The leader label expression is stored in the DHT for fast and easy lookup. The number of leaders can be expanded simply by changing the node label expression (see examples below). Because a structured overlay network maps physical nodes to virtual subspaces in a manner that is sensitive to both density and topology, the leader nodes are evenly distributed among physical non-leader nodes and every non-leader node has one or more nearby leader nodes.

Initiation of leader selection. A node that wishes to initiate the leader selection procedure for some service hashes information about the service into the DHT. This includes the public key of the initiator and the leader selection policy. This policy contains the leader label expression, the minimum criteria for a node to be a leader, and maximum lifetime of a leader. A non-leader node can discover the identity of leader nodes for this service by accessing the DHT using the service related key to lookup the leader label expression.

Example 1: CAN structured overlay. Assume the CAN space is a d-dimensional space and the length of the i_{th} dimension is d_i. To select k leaders, the service initiator first factors k into k_1,k_2,k_3,...,k_d where $k = \prod_{i=1}^{d} k_i$. The i_{th} dimension of the leader label should obey the formula $(b_i + n * d_i/k_i)\%d_i$, where b_i is a random number chosen in the range of [0,d_i] and n is an integer. The randomness prevents different service initiators from choosing the same group of leaders. Figure 1 illustrates this process for 16 leaders in a two-dimensional CAN space.

Example 2: Pastry structured overlay. To select k leaders, the service initiator needs to generate a node label with

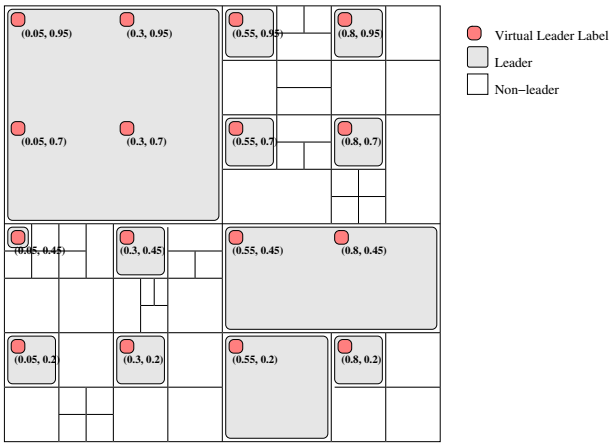

Fig. 1. Leader selection in CAN. (Leader label expression is $(0.05 + 0.25 * n)\%1, (0.20 + 0.25 * m)\%1$

$\lceil log_2 k \rceil$ don't-care bits as the highest order bits (or the don't-care bits can be distributed randomly within the label); the remaining bits in the node label are randomly set to 0 or 1. For example, if the service initiator needs 1024 leaders it will use $\underbrace{\times \times \times \ldots \times}_{\text{10 bits}} \underbrace{1001 \ldots 1}_{\text{118 bits}}$ as the leader label expression. Any node whose label matches the 118 instantiated bits is a potential leader. When a message are sent towards the leader label, the physical node with the closest node label will receive it.

Leader takes charge of the service. The initiator can send a message towards the leader labels to inform the chosen leaders. The notification is done via multicast in the overlay network. Each physical node that owns a node whose label matches the leader label expression will receive the message and become a leader. Alternatively, a leader node takes charge upon receiving the first request from a non-leader node.

Non-leader joins service. If a non-leader node wants to join a service, it looks up leader label expression in the DHT. The non-leader node can then figure out the nearest leader in the virtual overlay according to the routing protocol. The structured overlay network provides bounded virtual routing with path length proportional to the distance between labels in the label space. In CAN the non-leader node uses the Cartesian distance between its own label and the leader label to estimate distance. In Pastry, the non-leader uses the bit-wise XOR operation to compute the label distance from which it estimates the physical distance.

Resilience and fault tolerance. SOLE must function in situations in which leader nodes depart gracefully or die suddenly. A leader can replicate leader-related state on the neighbor nodes that will take over the subspace covered by its label if it fails. Leader failure will be detected by the underlying overlay network maintenance protocol. The most capable neighbor of the failed leader can then take over the leader role.

Heterogeneity. Leader nodes should be those nodes with better capability with respect to CPU speed, network connections, and other resources. If a new node joins towards an existing leader coordinate, the new node and the existing physical node serving as a leader can negotiate to see which one is more capable to take the leader role. Finally, a nearby non-leader node can offer to take over a nearby leader node's role if it is more capable by swapping virtual subspaces.

Adaptability. A non-leader can switch to another leader when it is not satisfied with the performance of its current leader. It can determine the distance to other leaders by comparing its own node label expression with the leader's node label expression, or it can query the DHT to see if new leaders have been selected by the initiator. When more leaders are needed the initiator simply expands the leader label expression to cover more virtual node labels.

Many peer-to-peer applications built on structured overlay networks can benefit from SOLE. For example, SOLE can be used to dynamically select leaders to act as rendezvous points for applications such as cycle sharing [23], publish/subscribe [32], or storage sharing systems [31]. The rendezvous points allow publishers/producers to advertise resources or services and subscribers/consumers to learn about them. The rendezvous point is also a place where publishers or subscribers with common interests learn about each other to form collaborative groups. Research in resource discovery in peer-to-peer cycle sharing systems has shown that if rendezvous points are used to collect resource information and to match queries from clients, the performance will be dramatically improved compared with probing-based or advertisement-based resource discovery methods [43].

B. PoPCorn: Leader Selection on a Coordinate-based Overlay Network

PoPCorn assumes an n-dimensional Euclidean coordinate space using an Internet coordinate system such as GNP [25] or Vivaldi [9]. PopCorn is suited for applications that wish to select a fixed set of k leaders and distribute them evenly throughout the overlay, to perform a service such as security monitoring, protocol testing, or data repositories. PoPCorn's primary distribution criteria, *dispersal*, is achieved by maximizing the sum of inter-node distances between all pairs of leader nodes. PoPCorn also achieves good *access* from non-leaders to leader nodes, and can be easily extended to address heterogeneity, adaptability, and fault tolerance.

The PoPCorn protocol selects k leaders by dispersing k tokens through the overlay coordinate space using a repulsion model among the tokens, analogous to forces among charged particles. Each token represents one of the leaders which moves through the overlay based on the forces exerted on it by other tokens. When equilibrium is reached, each node holding a token is selected as a leader.

PoPCorn has three phases: initial token placement, token adjustment, and finalization.

Initial Token Placement. The Initiator sends out k tokens to random peers in the overlay. Each peer can receive at most one token. Any peer which receives a token becomes a potential leader. The initiator can distribute the tokens itself or it can ask its neighbors to help distribute tokens.

Token Adjustment. The repulsion model is used to adjust the location of the tokens. After a node receives a token and

becomes a potential leader, it will start a scoped gossiping session with its neighbors to tell them the coordinates of the token it holds. Whenever a gossiping message arrives, a potential leader will re-calculate the combined force vector of repulsions from nearby tokens. If the magnitude of the combined repulsions exceeds a threshold *TR*, this potential leader will pass its token to one of the neighbors whose position is closest to the direction of the combined repulsion vector.

Figure 2 illustrates the repulsion model with a simple example in which the nodes are evenly distributed in a 2-dimensional Euclidean space. There are three potential leaders: A, B, and F. Node A receives two repulsions R_F and R_B, from nodes F and B, respectively. The combination of the two repulsions is vector R. Among A's neighbors, D has the smallest angle with R ($\angle RAD$ is the smallest). If the magnitude of R exceeds A's threshold, A will pass its token to node D, which will became a new potential leader.

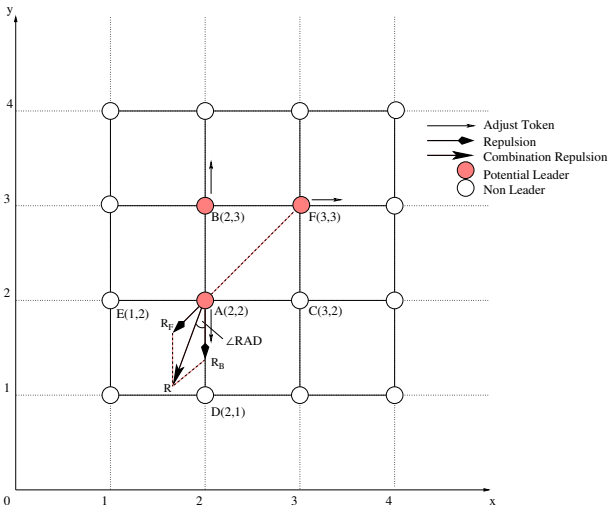

Fig. 2. PopCORN Repulsion Protocol

Finalization. If the time that a token stays on one potential leader node exceeds some time limit T, the node will mark its status as stable and it will no longer participate in token migration. After the PoPCorn tokens have stabilized, each node holding a token reports its position to the PoP initiator who will eventually ship the code for the PoP task as well as the location of the other tokens.

Heterogeneity The threshold function *TR* for a node can be changed according to the qualifications of that node to be a leader. Nodes that are well qualified to serve as leaders have higher threshold values, while poorly qualified nodes have lower thresholds. An unqualified node can set its threshold to zero.

Adaptiveness and Resilience. Each token has a unique ID and nodes which hold the tokens gossip the IDs of the tokens. Loss of a token caused by node leave or failure will be noticed by missing heartbeat message from that token. The neighbor tokens can decide to replace this missing token by generating a token with an ID identical to the missing token. They will place the new token in the last reported location of the missing token, and then let the PoPCorn protocol adjust the location of that new token based on repulsion forces. Alternatively, the node which discovers the loss of a token can report to the initiator and let it decide whether it needs to inject a new token or not.

Load Balance. The threshold function in the PoPCorn protocol can be used in other creative ways, for example, to meet load balancing criteria. In order to make more tokens stay in high density regions of the overlay, a potential leader can query its neighbors and gather local information about the number of nearby nodes. Based on this information, if a potential leader detects that it is in a high density region, it can set a higher threshold which must be overcome for the token to move. Similarly, potential leaders in lower density regions will set a lower threshold and will be more likely to pass tokens on to other nodes.

PoPCorn was designed as part of the CCOF (Cluster Computing on the Fly) [23] project for peer-to-peer cycle sharing (harnessing idle cycles throughout the Internet). PoPCorn places tasks that collectively form a distributed point-of-presence (PoP) application into a cycle sharing overlay network. PoP applications typically have low CPU and moderate communication requirements. Examples include distributed monitoring applications such as security monitors and Internet measurement monitors, and distributed protocol testing. Compared to volunteer-based systems, like NETI@home [36], which allows hosts to volunteer to run network measurement code, PoPCorn can better satisfy the distribution criteria and place tasks evenly throughout the overlay.

C. H_20: Leader selection protocol for unstructured overlay networks

The H_20 (Hierarchical 2-level Overlay) protocol for leader selection is a distributed negotiation protocol for unstructured overlay networks. It is basically a scalable protocol for multiple (c, d) colored domination that addresses the following leader selection requirements: access, load balance, and fault tolerance in a dynamic and heterogeneous environment, and some security issues.

H_20 uses a classic advertisement-based protocol, in which leaders advertise leader information, and non-leaders cache these advertisements. Non-leaders can then choose to join the best leader(s) using information contained in the cached advertisements. This protocol gives full autonomy to both leaders and non-leaders, allowing each to negotiate using its own local policy.

H_20 is similar in many ways to the gnutella protocol, but allows for finer-grained control over the leader selection process e.g., it can consider trust, secure paths, and routing performance as part of the leader selection process.

Leader Advertisement. A node capable of serving as a leader advertises itself to its neighbors within a certain scope. The advertisement includes information about its qualifications to be a leader (such as trust level, uptime, bandwidth, and neighborhood size). Each advertisement is propagated a certain number of hops (set by each individual leader node) and

carries with it the route travelled, i.e. an ordered list of the overlay nodes visited. This information is used by non-leaders as part of the selection criteria. A node that receives an advertisement message caches the advertisement about the potential leader in its local cache.

Leader Search. If a new node want to find leaders, it first consults its local cache for leader candidates. If there are no suitable candidates in the cache, it queries its immediate neighbors. If a contacted neighbor is a leader, it replies to the requestor with its qualifications. If a contacted neighbor is not a leader, it replies with entries from its local cache and the requestor will cache these new responses. The requestor then chooses the best candidate(s) according to its own criteria, and applies to those candidate leaders. The contacted leaders can confirm or reject such requests.

If the requestor does not hear from anyone within a given time interval or is not satisfied with the current leaders, it has several choices: if it is qualified, it can declare itself a leader and begin the advertisement protocol; it can join the overlay at another node, or it wait for a ranodm amount of time and try again.

Handshake. The handshake occurs when a requestor has identified a node that it wishes to have as its leader. It then sends an application message to the candidate leader, and waits for an acknowledgment. A leader that receives an application decides, based on its own local criteria, and returns an acknowledgment. The leader to non-leader relationship is solidified when the non-leader sends a confirmation message to the leader.

H_20 is currentlyused to create a 2-level hierarchy for communication among security monitors within the Sequoia collaborative security monitoring system [19]. The leaders form themselves into an overlay network which is utilized as a backbone for fast and secure information dissemination. Only nodes that hold a security certificate (obtained from a central authority) are eligible to be leaders. A non-leader can check the trust level information in the certificate presented by a leader node. A non-leader can also choose a leader node based on trust-based routing criteria by evaluating the trust level of nodes along the path to the potential leader. For fault tolerance, a given non-leader can connect to several leader nodes, verifying that its connections to those leaders traverse disjoint paths. Latency criteria can also be considered.

H_20 can also be used to support threshold cryptographic techniques (secret sharing) [35], [5], [33] to certify public keys. An (n, t) threshold scheme distributes partial signatures to each of n parties in such a way that any $t \leq n$ parties can authenticate, but it is infeasible for any subset of $t-1$ parties to do so. In this context, H_20 selects a subset of the nodes in the network to act as certificate authorities (CAs) such that any ordinary node has one-hop access to at least t CAs, a colored $(t, 1)$ domination problem.

IV. Discussion and Conclusions

The challenge of the leader selection problem for peer-to-peer applications lies in its need to fulfill the leader selection criteria described earlier, in a way that is scalable, highly adaptive, fault tolerant, and that respects local node autonomy. Furthermore, the leader selection protocols must cope with the fact that in a heterogenous system, not all nodes are qualified to serve as leaders.

We have described three leader selection protocols that deal with some these challenges in unique ways.

SOLE and any protocol that uses a regular node labeling scheme for structured overlays are able to be adaptive through simple, efficient mechamisms such as changing a bit to a *don't care* within the leader label expression. In addition, the number of leaders grows and shrinks dynamically using the native structured overlay scheme. Fault tolerance is easy because the leader label expression encodes the identity of *all* the leaders as well as the distances to leaders, enabling non-leader nodes to locate more than one leader. This feature also provides maximal access.

Asynchronous distributed protocols such as PoPCorn and H_20 are maximally flexible because they provide local control over criteria such as leader qualifications, desired distance to leaders, number of leaders to access, load balancing, and security. Thus, they can be easily tailored to a wide variety of applications. However, they must be carefully designed to minimize message-passing overhead and converge to a (reasonably) stable set of leaders.

There are many open problems for the leader selection problem.

- *Algorithm and protocol development*. There is a need for fully distributed algorithms for the underlying graph theoretic dominating set and p-centers that operating under dynamic conditions. In addition, protocols that emphasize specific distribution criteria, or that are tailored to classes of applications need to be developed. Bridging the gap between abstract algorithms and practical protocols is challenging.
- *Performance analysis*. Measuring the performance of a leader selection protocol is a hard problem in a large dynamic network. Metrics, workloads, benchmarks from peer-to-peer computing need to be assembled that can be used to evaluate and compare leader selection protocols in dynamic large scale networks. It is not clear yet what metrics best capture disperal, access, proportion, and load balance.
- *Security*. Clearly a wide range of questions related to secure leader selection remain open. These include dealing with denial of service attacks on the leaders, as well as how to detect and neutralize malicious leaders.

The leader selection problem shows up in peer-to-peer applications ranging from content and space sharing to cycle sharing (grid computing under a peer-to-peer model), to peer-to-peer telephony and spam protection. By presenting generic protocols tailored to structured, coordinate-based, and unstructured overlay networks, we hope to useful groundwork for researchers and practioners in peer-to-peer computing.

Acknowledgements: We would like to thank Professors Art Farley and Andrzej Proskurowski for discussions about dominating sets and p-centers.

REFERENCES

[1] The gnutella protocol specification (version 0.6).

[2] Kazaa http://www.kazaa.com.

[3] Gnutella Protocol Development. `http://www.the-gdf.org/wiki/index.php?title=Main_Page`, 2005.

[4] H. Attiya and J. Welch. *Distributed Computing Fudamentals, Simulations and Advanced Topics*. Jonh Wiley & Sons, INC., 2004.

[5] M. Bishop. *Computer Security Art and Science*. Addison-Wesley, Boston, 2002.

[6] B. Bose, B. Broeg, and V. Lo. Lee distance, gray codes, and the torus. *International Journal of Telecommunication Systems Special Issue on High Performance Computing and Interconnection Networks*, 10, 1998.

[7] E.J. Cockayne, O. Favaron, and C.M. Mynhardt. Secure domination, weak roman domination, and forbidden subgraphs. *Bulletin of the Inst. of Comb. and Apps.*, 39:87–100, 2003.

[8] D. Culler, D. Estrin, and M. Srivastava. Overview of sensor networks. *IEEE Computer*, pages 41–49, Aug 2004.

[9] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *SIGCOMM '04*.

[10] B. Das and V. Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *ICC (1)*.

[11] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: scalable coordination in sensor networks. In *MobiCom'99*.

[12] A.M. Farley and A. Proskurowski. Defensive domination. In *the 35th South–Eastern Conference on Combinatorics, Graph Theory and Computing*, 2005.

[13] Ronaldo A. Ferreira, Suresh Jagannathan, and Ananth Grama. Enhancing locality in structured peer-to-peer networks. In *ICPADS*, 2004.

[14] M. R. Garey and D. S. Johnson. Computers and intractability: A guide to the theory of np-completeness. In *W. H. Freeman Co.*, San Francisco, 1979.

[15] G. Handler and P. Mirchandani. *Location on Networks: Theory and Algorithms*. The MIT Press, 1979.

[16] T. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of Domination in Graphs*. 1998.

[17] T.H. Haynes, S.T. Hedetniemi, and M.A. Henninga. Global defensive aliances in graphs. *Electronic Journal of Combinatorics*, 10, 2003.

[18] D. Hochbaum and A. Pathria. Locating centers in a dynamically changing network and related problems. *Location Science*, pages 243–256, 1998.

[19] X. Kang, D. Zhou, D. Rao, J. Li, and V. Lo. Sequoia: A robust communication architecture for collaborative security monitoring systems. In *Poster session, SIGCOMM'04*.

[20] M. Karlsson, M. Mahalingam, and Z. Xu. Turning heterogeneity into an advantage in overlay routing. In *INFOCOM'02*.

[21] J. Kratochvil, P. Manuel, M. Miller, and A. Proskurowski. Disjoint and fold domination in graphs. *Australasian Journal of Combinatorics*, 18:277–292, 1998.

[22] J. Li and A. Farley. Dominating set working notes, university of oregon, networks research group, 2005.

[23] V. Lo, D. Zappala, D. Zhou, Y. Liu, and S. Zhao. Cluster Computing on the Fly: P2P scheduling of idle cycles in the internet. In *IPTPS'04*.

[24] A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1997.

[25] T. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *INFOCOM'02*.

[26] V. Paxson L. Zhang D. Gryniewicz P. Francis, S. Jamin and Y. Jin. An architecture for a global Internet host distance estimation service. In *INFOCOM'99*.

[27] V. Paxson. End-to-end routing behavior in the Internet. In *SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 1996.

[28] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *SIGCOMM'01*.

[29] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of IEEE INFOCOM'02*, 6 2002.

[30] A. Rowstron and P. Druschel. Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proc. 18th IFIP/ACM Int'l Conf. on Distributed Systems Platforms*, Nov. 2001.

[31] A. Rowstron and P. Druschel. Storage management and caching in Past, a large-scale, persistent peer-to-peer storage utility. In *Proc. 18th ACM SOSP'01*, 2001.

[32] Antony I. T. Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *Networked Group Communication*, pages 30–43, 2001.

[33] F. Schlicher. Key management and distribution for threshold cryptography schemes. Technical report, Technischen Universitat Munchen, January 2004.

[34] J. Shaikh, J. Solano, I. Stojmenovic, and J. Wu. New metrics for dominating set based energy efficient activity scheduling in ad hoc networks. In *IEEE International Conference on Local Computer Networks*, Oct 2003.

[35] A. Shamir. How to share a secret. *Comm. ACM*, 22(11):612–613, 1979.

[36] Charles Robert Simpson, Jr., and George F. Riley. Neti@home: A distributed approach to collecting end-to-end network performance measurements. In *5th Passive and Active Measurement Workshop*, Apr. 2004.

[37] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup service for internet applications. In *Proc. ACM SIGCOMM*, pages 149–160, Aug. 2001.

[38] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proc. ACM SIGCOMM*, pages 149–160, Aug. 2001.

[39] V. Vazirani. *Approximation Methods*. Springer-Verlag, 1999.

[40] P. Wan, K. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. *Mob. Netw. Appl.*, 9(2), 2004.

[41] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *DIALM'99*.

[42] J. Wu and H. Li. Domination and its applications in ad hoc wireless networks with unidirectional links. In *Proceedings of the Proceedings of the 2000 International Conference on Parallel Processing*, 2000.

[43] D. Zhou and V. Lo. Cluster Computing on the Fly: in a cycle sharing peer-to-peer system. In *Proceedings of the 4th International Workshop on Global and P2P Computing (GP2PC'04)*, 2004.