Scalable Supernode 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

Abstract

We define a problem called the supernode selection problem *which has emerged across a variety of peer-to-peer applications. Supernode selection involves selection of a subset of the peers to serve a special role. The supernodes must be well-dispersed throughout the peer-to-peer overlay network, and must fulfill additional requirements such as load balance, resource needs, adaptability to churn, and heterogeneity. While similar to dominating set and p-centers problems, the* supernode selection problem *must meet the additional challenge of operating within a huge, unknown and dynamically changing network. We describe three generic supernode 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 supernode selection problem can benefit the peer-to-peer community through cross-fertilization of ideas and sharing of protocols.*

1. Introduction

We have identified a fundamental problem, which we call the *supernode 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 supernode selection problem also shows up in the fields of sensor networks, ad-hoc wireless networks, and peer-based Grid computing. Supernode 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 supernode selection problem is highly challenging because in the peer-to-peer environment, a large number of supernodes 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 selection don't work. Supernode 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.

Supernode selection shows up in many peer-to-peer and networking applications. For example, in peer-to-peer file sharing systems, such as Kazaa [11] and Gnutella [6], protocols were developed for the designation of qualified supernodes (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 [5, 16, 14]. In ad-hoc 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 [22, 2, 23] so that every node is within broadcast range of a bridging node. Within sensor networks, supernodes 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 [3].

In this paper, we demonstrate that many peer-to-peer and networking applications are seeking solutions to variations on the same fundamental problem. We first define a general model for the supernode selection problem, describing its key requirements and challenges. We show how the supernode selection problem is related to dominating set and p-centers problems, and describe how the supernode selection problem is instantiated in

[*]Supported in part by NSF 9977524 and an NSF Graduate Research Fellowship

several well-known peer-to-peer applications. We then present three general purpose supernode 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.

SOLE: a label-based supernode selection protocol for applications built on a structured overlay network such as CAN [16], Chord [20] or Pastry [18]. SOLE exploits the regular node labeling schemes used within structured overlays to distribute supernodes evenly throughout the overlay, to ensure fast access from non-supernodes to supernodes, and to maintain a fixed supernode to non-supernode ratio as the overlay grows and shrinks dynamically.

PoPCorn: a protocol for applications that use a coordinate-based overlay network using systems such as Vivaldi [1] or GNP [14]. This protocol distributes a fixed number of supernodes evenly with respect to the topology of the overlay network using a repulsion model.

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

2. Problem Definition and Background

We first describe the unique requirements and challenges of the supernode 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 supernode selection problem. We conclude by surveying examples of applications from peer-to-peer computing and other networking domains that call for scalable supernode selection protocols.

2.1. The Supernode Selection Problem

We broadly define the supernode 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 supernodes providing service to the non-supernodes. A potentially large and undefined number of supernodes must be selected from an unknown, large scale, and dynamically changing overlay network. First, we describe key topological distribution criteria that the supernodes must fulfill relative to the non-supernodes. Second, we enumerate characteristics of peer-to-peer networks that make supernode selection difficult.

Supernode Distribution Criteria. The supernodes 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-supernodes must have low latency access to one or more supernodes. Access can be measured in hop counts or delay.

Dispersal: supernodes must be evenly distributed throughout the overlay network; they should not be clustered within only a few subregions of the overlay.

Proportion: a pre-specified global ratio of supernodes to non-supernodes must be maintained to meet application-specific performance requirements.

Load balance: supernodes should not serve more than k non-supernodes, where k can be configured locally based on the resource capability of each supernode.

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.

Peer-to-Peer Factors. The design of supernode 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.

Heterogeneity: Current large-scale peer-based systems consist of a large number of heterogeneous nodes with differing hardware and software resources. A node might not be eligible to be a supernode 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. Supernode selection protocols must be able to handle churn and respond quickly, especially when supernodes leave the system. Supernode selection protocols must also be adaptive to dynamic changes in network traffic and overlay topology.

Resilience and fault tolerance: When a given supernode dies, other supernodes should quickly take over its functions or a new supernode should be quickly selected.

Security: Supernodes may be vulnerable to denial of service attacks, malicious supernodes can disrupt the system by failing to forward messages or by giving out wrong information.

2.2. 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 supernode selection problem.

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 [8]. Most versions are NP-hard.

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. *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.

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. 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. A *global offensive alliance* is such that every vertex not in S has a majority of its neighbors in S. A *k-defensive dominating set* is a set of vertices that can counter an attack on any k vertices where an attack on a vertex must be countered by itself or by a neighbor.

The p-center problem is applicable when placing a *fixed* number of supernodes in a network. Algorithms and variations on this NP-hard discrete location problem are found in [7]. 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.

The classic leader election problem from distributed computing differs from supernode selection in that the former assumes all nodes vote (directly or indirectly) on the choice of each supernode. 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. [13].

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. These algorithms were not designed for large scale peer-to-peer networks that exhibit a high degree of churn and that are dynamically heterogeneous.

2.3. Ultrapeers, landmarks, and rendezvous points

The best known example of supernode selection in a peer-to-peer application is the *gnutella* protocol [6] for selection of ultrapeers—peers with sufficient bandwidth and processing power to serve as proxies for other peers. The use of ultrapeers reduces network traffic and speeds up content discovery.

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 supernodes 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. Gnutella's supernode 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 this ratio. Security is not addressed in gnutella.

A canonical supernode 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 [17] structured overlay is constructed using a *binning* technique in which carefully placed landmarks are used to construct a topologically sensitive overlay. Also, in GNP [14], a coordinate system is built using landmark nodes. It has been shown that the accuracy of the GNP coordinates is highly sensitive to the location of the landmark nodes. Choosing landmark nodes boils down to supernode selection for a fixed, small number of supernodes that are well-dispersed in the physical network. The notion of evenly distributed or well-dispersed can be captured by metrics based on pairwise distances, N-medians and N-cluster medians [14], or distances from a centroid. Current approaches use manual placement of landmarks or centralized dominating set algorithms [21].

Examples from outside peer-to-peer computing illustrate the wide scope of the supernode selection problem. In ad-hoc wireless networks, a distributed dominating set protocol is used to select nodes to serve as intermediary routing nodes to bridge the distance between wire-

less nodes that are out of broadcast range of each other [22, 2]. The IDMaps [5] 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 [3] under access and battery life constraints.

Table 1 lists a few examples of the supernode selection problem for a range of applications, showing the commonality and diversity in supernode qualifications and supernode distribution requirements.

Projects/papers	Supernode Qualifications	Distribution Criteria
Distance Estimation		
CAN[17], GNP[14], IDMaps (Tracer/Box)[5]	Minimal	Inter-landmark distance,N-medians,N-cluster-median, Fixed # of landmarks
Content Location		
Gnutella[6], Kazaa[11]	Operating system, Bandwidth, Uptime, Memory/CPU	Maintain # of ultra-peer in proportion to leaf nodes
Routing		
Expressway[10], Sequoia (router)[9]	Highly available, Fan-out, Bandwidth, Trust	Close to network access points, k hops to c supernodes, Disjoint paths
Data Aggregation		
Sensor networks[3]	Battery life	Within signal radius
Rendezvous Point		
Sequoia (RP)[9], Ferreira[4]	Bandwidth, Uptime, Memory/CPU, Trust	Easily accessible, Distribution fits density of non-supernodes
Monitors (point-of-presence)		
IDMaps [15], NETI@home[19]	Minimal	Fixed # of monitors, Sensitive to network topology

Table 1. Supernode Selection Examples

3. Three New Supernode 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 networking techniques such as random selection of peers, or flooding-based search for suitable supernodes do not work. Nodes selected by a random strategy may not be qualify to be supernodes and may be poorly dispersed. Flooding-based search for a large number of supernodes is not scalable and does not adapt well to dynamic changes in the network.

For these reasons, we introduce three new supernode 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 supernode selection. Performance evaluation of these protocols is part of our ongoing research agenda, but outside scope of this paper.

3.1. SOLE: Supernode selection in structured overlay networks

SOLE is designed to select a group of supernodes in a structured overlay network, with a goal of keeping the supernode to non-supernode ratio stable as peers join and leave the overlay. SOLE also maintains low access from non-supernodes to supernodes and provides load balancing for each supernode. In the last part of this section, we discuss how SOLE addresses resource heterogeneity, churn, and fault tolerance.

A DHT (Distributed Hash Table) built on a structured overlay network such as CAN [16], Chord [20], and Pastry [18] 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 supernodes. The supernode label expression is stored in the DHT for fast and easy lookup. The number of supernodes 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 supernodes are evenly distributed among physical non-supernode nodes and every non-supernode has one or more nearby supernodes.

Initiation of supernode selection. A node initiates the supernode selection procedure for some service by hashing information about the service into the DHT: the public key of the initiator and the supernode selection policy. This policy contains the supernode label expression, the minimum criteria for a node to be a supernode, and maximum lifetime of a supernode. A non-supernode can discover the identity of supernodes for this service by accessing the DHT using the service related key to lookup the supernode 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 supernodes, 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 supernode 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 supernodes. Figure 1 illustrates this process for 16 supernodes in a two-dimensional CAN space.

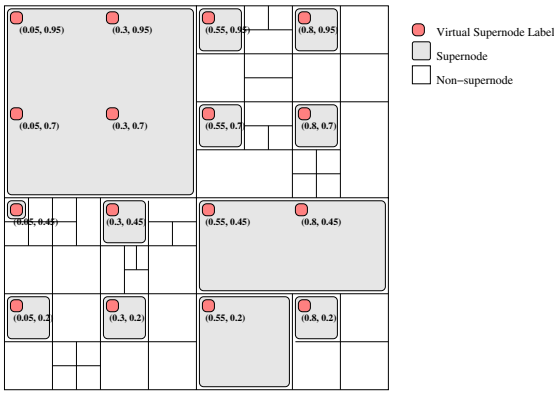

Figure 1. Supernode selection in CAN. (Supernode label expression is $(0.05+0.25*n)\%1, (0.20+0.25*m)\%1$

Example 2: Pastry structured overlay. To select k supernodes, the service initiator needs to generate a node label with $\lceil log_2 k \rceil$ 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 supernodes it will use $\underbrace{\times \times \times \ldots \times}_{\text{10 bits}} \underbrace{1001 \ldots 1}_{\text{118 bits}}$ as the supernode label expression. Any node whose label matches the 118 instantiated bits is a potential supernode. When a message are sent towards the supernode label, the physical node with the closest node label will receive it.

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

Non-supernode joins service. If a non-supernode wants to join a service, it looks up the supernode label expression in the DHT. The non-supernode can then figure out the nearest supernode in the virtual overlay according to the label-based 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-supernode uses the Cartesian distance between its own label and the supernode label to estimate distance. In Pastry, the non-supernode 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 supernodes depart gracefully or die suddenly. A supernode can replicate supernode-related state on the neighbor nodes that will take over the subspace covered by its label if it fails. Supernode failure will be detected by the underlying overlay network maintenance protocol. The most capable neighbor of the failed supernode can then take over the supernode role.

Heterogeneity. Supernodes 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 supernode coordinate, the new node and the existing physical node serving as a supernode can negotiate to see which one is more capable to take the supernode role. Finally, a nearby non-supernode can offer to take over a nearby supernode's role if it is more capable by swapping virtual subspaces.

Adaptability. A non-supernode can switch to another supernode when it is not satisfied with the performance of its current supernode. It can determine the distance to other supernodes by comparing its own node label expression with the supernode's node label expression, or it can query the DHT to see if new supernodes have been selected by the initiator. When more supernodes are needed the initiator simply expands the supernode 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 supernodes to act as rendezvous points for applications such as cycle sharing, publish/subscribe, or storage sharing. 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 [24].

3.2. PoPCorn: Supernode Selection on a Coordinate-based Overlay Network

PoPCorn assumes an n-dimensional Euclidean coordinate space using an Internet coordinate system such as GNP [14] or Vivaldi [1]. PopCorn is suited for applications that wish to select a fixed set of k supernodes 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 supernodes. PoPCorn also achieves good *access* from non-supernodes to supernode, and can be easily extended to address heterogeneity, adaptability, and fault tolerance.

The PoPCorn protocol selects k supernodes 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 supernodes 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 supernode.

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 supernode. 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 supernode, 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 supernode will recalculate the combined force vector of repulsions from nearby tokens. If the magnitude of the combined repulsions exceeds a threshold *TR*, this potential supernode will pass its token to the neighbor 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 supernodes: 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 supernode.

Finalization. If the time that a token stays on one potential supernode 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.

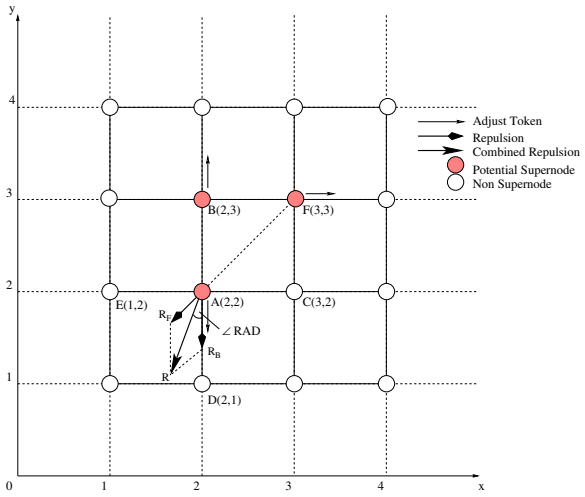

Figure 2. PoPCorn Repulsion Protocol

Heterogeneity The threshold function *TR* for a node can be changed according to the qualifications of that node to be a supernode. Nodes that are well qualified to serve as supernodes 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 messages 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 supernode can query its neighbors and gather local information about the number of nearby nodes. Based on this information, if a potential supernode 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 supernodes 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) [12] 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 security monitors, Internet measurement monitors, and distributed protocol testing. Compared to volunteer systems, like NETI@home, PoPCorn can better satisfy the distribution criteria and place tasks evenly throughout the overlay.

3.3. H_20: Supernode selection protocol for unstructured overlay networks

The H_20 (Hierarchical 2-level Overlay) protocol for supernode selection is a distributed negotiation protocol for unstructured overlay networks. It is essentially a scalable protocol for multiple (c, d) colored domination that addresses the following supernode 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 supernodes advertise supernode information, and non-supernodes cache these advertisements. Non-supernodes can then choose to join the best supernode(s) using locally cached information. This protocol gives full autonomy to both supernodes and non-supernodes, 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 supernode selection process e.g., it can consider trust, secure paths, and routing performance.

Supernode Advertisement. A node capable of serving as a supernode advertises itself to its neighbors within a certain scope. The advertisement includes information about its qualifications to be a supernode (such as trust level, uptime, bandwidth, and neighborhood size). Each advertisement is propagated a certain number of hops (set by each individual supernode) and carries with it the route travelled, i.e. an ordered list of the overlay nodes visited. This information is used by non-supernodes as part of the selection criteria. A node that receives an advertisement message caches the advertisement about the potential supernode in its local cache.

Supernode Search. If a new node want to find supernodes, it first consults its local cache for supernode candidates. If there are no suitable candidates in the cache, it queries its immediate neighbors. If a contacted neighbor is a supernode, it replies to the requestor with its qualifications. If a contacted neighbor is not a supernode, 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 supernodes. The contacted supernodes 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 supernodes, it has several choices: if it is qualified, it can declare itself a supernode and begin the advertisement protocol; it can join the overlay at another node, or it wait for a random amount of time and try again.

H_20 is currently used to create a 2-level hierarchy for communication among security monitors within the Sequoia collaborative security monitoring system [9]. The supernodes 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 supernodes. A non-supernode can check the trust level information in the certificate presented by a supernode. A non-supernode can also choose a supernode based on trust-based routing criteria by evaluating the trust level of nodes along the path to the potential supernode. For fault tolerance, a given non-supernode can connect to several supernodes, verifying that its connections to those supernodes traverse disjoint paths. Latency criteria can also be considered.

H_20 can also be used to support threshold cryptographic techniques (secret sharing) 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. 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.

4 Discussion and Conclusions

The challenge of the supernode selection problem for peer-to-peer applications lies in its need to fulfill dispersal and access criteria in a way that is scalable, highly adaptive, fault tolerant, and that respects local node autonomy. Furthermore, the supernode selection protocols must cope with an unknown, large scale, dynamically changing, and heterogeneous network.

We have described three supernode selection protocols that deal with some of these challenges in unique ways. SOLE and any protocol based on structured over-

lays capitalizes on the use the supernode label expression to efficiently encode the identity of *all* the supernodes as well as the distances to supernodes. Asynchronous distributed protocols such as PoPCorn and H_20 are maximally flexible because they provide local control over supernode selection criteria. 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 supernodes.

There are many open problems for the supernode selection problem. (1) *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. (2) *Performance analysis.* Measuring the performance of a supernode 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 supernode selection protocols in dynamic large scale networks. It is not clear yet what metrics best capture dispersal, access, proportion, and load balance. (3) *Security.* Clearly a wide range of questions related to secure supernode selection remain open. These include dealing with denial of service attacks on the supernodes, as well as how to detect and neutralize malicious supernodes.

We believe an integrated approach to the supernode 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.

References

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

[2] B. Das and V. Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *ICC (1)*, pages 376–380, 1997.

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

[4] R. A. Ferreira, S. Jagannathan, and A. Grama. Enhancing locality in structured peer-to-peer networks. In *ICPADS*, 2004.

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

[6] Gnutella Protocol Development. `http://www.the-gdf.org`, 2005.

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

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

[9] 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*.

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

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

[12] 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*.

[13] A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, San Francisco, 1997.

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

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

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

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

[18] 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.

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

[20] 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*, San Diago, CA, Aug. 2001.

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

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

[23] J. Wu, F. Dai, M. Gao, and I. Stojmenovic. On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks. *Journal of Communications and Networks*, 4(1), March 2002.

[24] 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.