Creating an Adaptive Network of Hubs Using Schelling's Model

Atul Singh and Mads Haahr
Distributed Systems Group,
Department of Computer Science,
Trinity College,
Dublin, Ireland
Email: Atul.Singh@cs.tcd.ie, Mads.Haahr@cs.tcd.ie

***Abstract*— Thomas Schelling's model suggests an explanation for the existence of segregated neighborhoods in America. This paper presents a study on utilizing Schelling's model to create an adaptive network of hubs in an unstructured decentralized P2P network. The hub network is attractive because it can be used to improve the performance of the overlay network. The paper describes an abstract version of Schelling's algorithm, which can be used to create a family of topology adaptation algorithms for P2P networks. This paper presents one such algorithm which can be executed by the peers to create a network of hubs within a decentralized unstructured network.**

I. Introduction

The term Peer-to-Peer (P2P) is used to refer to distributed systems without any central control, where all the nodes (called peers) are equivalent in functionality. In a P2P system, peers can collaborate and communicate with each other without the need for centralized components. P2P systems organize the peer computers in a virtual communication network called the overlay network. The overlay network generally has self-organizing characteristics. It is established and maintained by the P2P software without any human intervention. P2P software manages events like peers joining and leaving the network. This self-organizing nature of P2P helps in reducing the management cost of the computer infrastructure. However the decentralized nature of P2P networks makes it difficult to develop efficient algorithms for tasks like clustering and search, which are required for many P2P applications.

The *topology* of the overlay network is the graph whose vertices are the peers in the network and edges are all the connections between the peers. *Topology Adaptation* involves adjusting an overlay network topology to satisfy certain criteria when peers leave or join the network. On the basis of the overlay network architecture, P2P applications can be divided into three major categories: centralized, decentralized structured and decentralized unstructured [1]. The algorithm presented in this paper is useful for decentralized unstructured networks only. In a decentralized structured network the location of a peer is determined by the key space it is responsible for, which makes topology adaptation difficult.

In existing P2P overlay networks (e.g., Gnutella [2]), there is typically no control over the type of peers which are connected as neighbors. This leads to a suboptimal grouping of peers. For example, in a file-sharing application, peers with high bandwidth capacity may be grouped together with peers with low bandwidth capacity, which may lead to a degradation in performance. P2P applications can benefit by connecting peers on the basis of their characteristics to use the capabilities of all the peers more efficiently. For example, in file-sharing applications, it is beneficial if peers with similar properties (e.g., bandwidth or geographic location) are connected to each other.

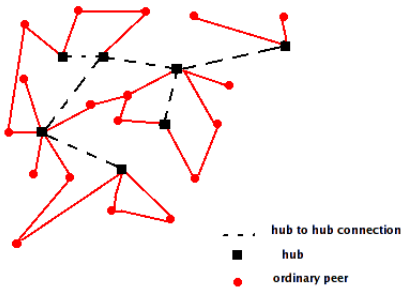

Fig. 1. *Hub-Based Topology* A network of hubs is created within the pure P2P network. The ordinary peers are connected with each other and can be connected to more than one hub unlike the super-peer topology.

In 1960, American economist Thomas Schelling proposed a model [3], [4], [5], [6] to explain the existence of the segregated neighborhoods in America. He observed that the segregated American neighborhoods are not caused by a central authority, or the desire of people to stay away from dissimilar people; but is a cumulative effect of simple actions of individuals.[1] Schelling's model is decentralized and self-maintaining in nature. This makes the model suitable for topology adaptation in the dynamic environments of unstructured decentralized P2P networks, which lack a central authority. This paper presents an approach based on Schelling's model that can be used to develop algorithms, which can be executed by the peers to create a P2P topology satisfying certain criteria. The topologies developed using this approach can adapt to the continuous arrival of new peers on the network.

The existing unstructured decentralized P2P (also called

[1]The algorithm executed by individuals in Schelling's model is referred as Schelling's algorithm in the paper.

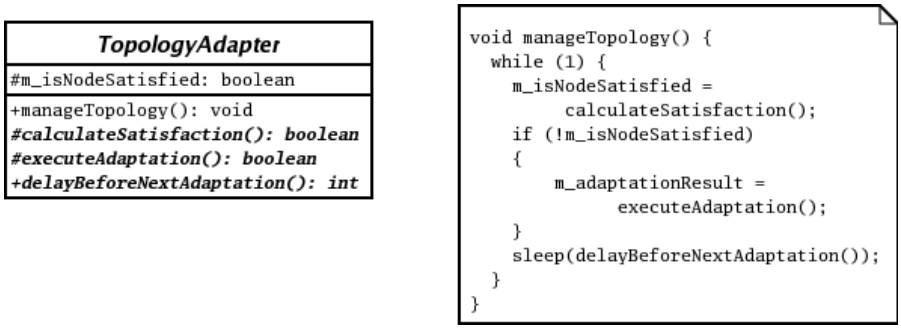

Fig. 2. UML Class diagram of a TopologyAdapter Interface. The manageTopology method presents the structure of the *abstract Schelling's algorithm*. The methods calculateSatisfaction, executeAdaptation and delayBeforeNextAdaptation are virtual abstract methods implemented in subclasses.

Operation	Details
count(*property*)	The number of neighbors of a given node matching the given property.
add(*peer*)	Add the given peer or peer's as a neighbor
drop(*peer*)	Drop the given peer as a neighbor
neighbor(*property*)	Returns a neighbor with the given property.
search(*property*)	Search for a peer on the overlay network with the given property.

TABLE I

SOME OPERATIONS THAT CAN BE EXECUTED BY THE PEERS. THE OPERATIONS ARE USED TO DESCRIBE THE SATISFACTION CRITERION AND THE TOPOLOGY ADAPTATION STEPS.

pure P2P) overlay networks tend to be inefficient when it comes to performing a search on the network. Pure P2P systems like Gnutella flood the network with the query message. Considerable research has been done to improve the efficiency of search in a pure P2P network. The two prominent approaches that have been used are: using random routes to propagate the search query [7] and using super peers which maintain a directory of resources on the network to resolve a query. In super-peer systems like KaZaA [8] the clients send the search request to the super peer and receive replies from it. This reduces the need for expensive broadcasts on the overlay network. In the super-peer topology, the ordinary peers are connected to one super peer only, and are not connected with each other. The super peers are in turn connected to each other to form a pure P2P system. However the failure of super peers can have a catastrophic consequences causing a complete communication failure for the cluster of nodes attached to the super peer.

This paper presents an algorithm based on Schelling's model that can be used to create a variation (see Figure 1) of the super-peer topology in which the ordinary peers are connected with each other and can be connected to more than one super peer (called a *hub* in this paper). The hubs are connected to each other to form a network of hubs. In this topology the failure of a hub is not a catastrophic failure, because the connections between the peers can be used for communication in case of a hub failure. This topology is similar to the topology used in JXTA [9]. The results of simulations done using the algorithm to create the hub-based topology are also presented in the paper.

The rest of the paper is organized as follows. Section II describes Schelling's work in detail and presents an abstract form of Schelling's algorithm which can be executed by the peers to change the overlay network topology to satisfy particular constraints. It also discusses the design of the simulator used for the experiments. Section III presents a concrete realization of Schelling's algorithm that can be used to create an adaptive network of hubs within a pure P2P network. The results of the simulation to create a hub-based topology are also presented in this section. Section IV presents the conclusion of this work.

II. SCHELLING'S MODEL

In Schelling's model, the world is an $m \times n$ grid. A random number of cells in the grid are populated by blue or red turtles.[2] A cell can host only one turtle. In the beginning, a random number of blue and red turtles are randomly distributed on the grid. About one third of the cells in the grid are left empty. All the turtles desire a certain percentage of their neighbors to be of the same colour. If a turtle is not satisfied with its neighbors, it moves to an adjacent empty cell, chosen randomly. The simulation goes on till all the turtles are satisfied with their neighbors. As the simulation progresses, segregation can be observed on the grid. The segregation is an emergent behavior caused by the desire of the turtles to stay with a very small percentage of similar neighbors.

[2]Schelling studied his model using nickels and pennies on a chess board.

Satisfaction Criteria	Topology Adaptation Steps
$\frac{\textbf{count}(\textit{same property})*100}{\textbf{count}(\textit{all})} > PNSP$ where, PNSP is the desired Percentage of Neighbors with Similar Property	*step 1:* **drop**(**neighbor**(*different property*)) *step 2:* **add**(**search**(*same bandwidth*))
Same as above	**drop**(**neighbor**(*different property*))

TABLE II

TWO SET OF SATISFACTION CRITERIA AND TOPOLOGY ADAPTATION STEPS THAT CAN BE USED TO BRING TOGETHER PEERS WITH SIMILAR PROPERTIES (EG. BANDWIDTH).

In Schelling's model, the turtles act using their awareness of the local network topology, which makes this model especially attractive for P2P systems in which the peers lack a global picture of the network topology. In the model, grouping is maintained even when turtles join or leave the system, which makes this model ideal for the dynamic environments of P2P networks. The self-organizing and decentralized nature of Schelling's model makes it a suitable candidate solution for adapting P2P topologies. To the authors' knowledge, the effect of applying Schelling's algorithm to a P2P overlay network has not yet been studied. This paper studies the effect of applying Schelling's algorithm to peers in a P2P network.

In this work, the *Template method* [10] design pattern is used to create an abstract form of Schelling's algorithm. In the *Template method* design pattern, the skeleton of an algorithm is defined in an operation, deferring the steps which may change to a subclass. The subclasses implement the steps that vary. This makes it possible to create variations of the algorithm without changing its structure. For Schelling's algorithm, the steps which may vary are the satisfaction criteria, the actions to be performed if a peer is not satisfied and the frequency with which the satisfaction state should be checked. The manageTopology method pseudo-code in figure 2 presents the abstract Schelling's algorithm. A peer calculates its satisfaction state at pre-defined intervals and if it not satisfied then it executes its *topology adaptation steps* (TAS).

Satisfaction state is a boolean value indicating whether a peer is satisfied with its local view of the overlay network's topology. The satisfaction state of a peer is calculated using the calculateSatisfaction method. If a peer is not satisfied with its neighbors then *topology adaptation steps* are performed by calling the executeAdaptation method. The topology management algorithm, specified in the manageTopology method, is executed repeatedly. The time delay between successive executions of the topology management algorithm is determined by the return value of the method delayBeforeNextAdaptation. The satisfaction criteria (SC), the topology adaptation steps and the time delay will vary with the application and the topology desired. Table II presents two sets of examples for SC and TAS which have been used to cluster peers with similar bandwidths in a pure P2P network to utilize the bandwidth available on the network efficiently [11].

An overlay network simulator has been developed to evaluate the different variations of Schelling's algorithm. All the peers are within the same process in the simulator. The simulator is single-threaded, which means that the peers execute the algorithm sequentially. Each peer is assigned a numeric identifier. In each iteration the simulator goes through the peers in an increasing order of identifiers. An iteration is counted as one time unit. The peers execute the manageTopology algorithm in each iteration.

Operation	Details
newPeer(*type*)	Create a new hub or normal peer.
select(*n*)	Selects and returns *n* random peers chosen from the overlay network.
random()	returns a random number between 0 and 1

TABLE IV

THE TABLE SHOWS SOME OPERATIONS WHICH CAN BE EXECUTED BY THE SIMULATOR. THE OPERATIONS ARE USED TO DESCRIBE THE ALGORITHM USED BY THE SIMULATOR TO CREATE THE RANDOM NETWORK'S ON WHICH THE SIMULATIONS ARE PERFORMED.

```
step 1:
  Peer p = newPeer(random() =< 0.9 ? "peer" : "hub");
step 2:
  p.maxConnections = p.type == "hub" ? 20 : 5;
step 3:
   p.add(select(3));
```

TABLE V

THE ALGORITHM ABOVE IS USED BY THE SIMULATOR TO CREATE THE RANDOM NETWORK'S ON WHICH THE SIMULATIONS TO TEST THE HUB ALGORITHM ARE PERFORMED.

III. CASE STUDY: CREATING AN ADAPTIVE NETWORK OF HUBS

This section presents a case study that demonstrates how the abstract Schelling's algorithm can be utilized in a P2P network. Hubs are peers that have high availability and capacity. Hubs are used in the overlay network to perform resource-intensive tasks like maintaining a directory of resources on the network, which can be used to process search requests. A peer examines its capacity to decide whether it will act as a peer or a hub. The case study shows that an algorithm (called the *hub algorithm*) developed using the abstract Schelling's algorithm can be used

Peer	Satisfaction Criteria	Topology Adaptation Steps

Hub	$H_{max} >$ **count**(*hub*)
	and
	count(*hub*) ! $= 0$
	
	where, H_{max} is the maximum number of hubs desired as neighbors.
		step 1:
		if (**count**(*hub*) $> H_{max}$)
		drop(**neighbor**(*hub*))
		step 2:
		if (**count**(*hub*) $== 0$)
		add(**search**(*hub*))

Normal	**count**(*hubs*) > 0	*step 1:* if (**count**(*all*) $== maxNeighbours$) **drop**(**neighbor**(*any*)) *step 2* **add**(**search**(*hub*))

TABLE III

SATISFACTION CRITERIA AND TOPOLOGY ADAPTATION STEPS THAT WILL BE EXECUTED BY THE HUBS AND THE ORDINARY PEERS TO CREATE A BACKBONE NETWORK OF HUBS WITHIN THE OVERLAY NETWORK.

Seed / Peers	1	2	3	4	5	6	7
100	5	1	TD	3	3	2	3
1000	4	4	TD	5	TD	TD	5

TABLE VI

THE $H_{maxCritical}$ VALUE FOR THE DIFFERENT STATIC RANDOM OVERLAY NETWORKS ON WHICH THE SIMULTIONS WERE PERFORMED. THE SEED IS USED TO GENERATE A NEW SEQUENCE OF PSEUDO-RANDOM INTEGERS. TD STANDS FOR DISCONNECTED TOPOLOGY.

to create an adaptive network of hubs within a pure P2P network. The hub-based topology can be created using the TAS and SC shown in Table III.

Simulations have been performed using the simulator described in Section II to study the effect of applying the hub algorithm on a static overlay network and on an dynamic overlay network which has a constant inflow of peers. All the simulations have been done on random networks created using the algorithm described in Table V. In the random networks generated using this algorithm, 90 percent of the peers chosen randomly are assigned the role of ordinary peers and the rest are assigned the role of hubs. The maximum number of connections that a peer can have is called maxConnections. The maxConnections value is 5 for an ordinary peer and 20 for a hub. Each peer is initially connected to 3 randomly chosen peers from the overlay network. It is ensured that the generated random topology is connected. The **search** operation is performed using a Depth First Search (DFS) on the overlay network. The simulations and their results are discussed below.

A. Static Random Overlay Network

The first set of simulations have been done on a static random network created from scratch using the algorithm in Table V. Simulations have been done on four different random networks (created by using different seeds values for the random network generator on Linux) of 100, and 1,000 peers each using H_{max} values (H_{max} is the maximum number of hubs desired as neighbors by a hub) from 1 to 10. The simulations go on till all the peers are satisfied or 1,000 simulator iterations are reached. For a random network of 100 peers typically less than 400 messages are exchanged for the hub algorithm to converge. For a random network of 1,000 peers typically less than 10,000 messages are exchanged for the hub algorithm to converge.

A critical value of H_{max} (called $H_{maxCritical}$) was observed below which all the peers were not satisfied even after 1,000 simulator iterations. Table VI shows the $H_{maxCritical}$ value for the different random networks. The value of $H_{maxCritical}$ is different for different random networks. The authors were not able to find any corelation between the random network and the $H_{maxCritical}$ value. For some seed values (e.g., using seed value 3 for creating a random network of 100 peers) the initial topology generated was disconnected and so no simulations were performed on these topologies. When H_{max} is below $H_{maxCritical}$ the simulations do not converge because some of the hubs are not satisfied as they are not able to find another hub to establish a connection. When H_{max} is greater than or equal to $H_{maxCritical}$ all the peers are satisfied and the simulations converge within 5 simulator iterations. The simulations converge when all the peers are satisfied which means that each hubs on the overlay network is connected to at least one other hub and at most to H_{max} hubs and all the ordinary peers are connected to at least one hub.

B. Dynamic Overlay Network

The second set of simulations have been performed on a random network where new peers join the system every simulator iteration to demonstrate that the approach can be used in dynamic environments. The simulations start with a small random network of 100 peers, and 5 new peers are added every iteration till the number of nodes reaches 5,000. The simulations have been done using a H_{max} value of 5, as this was a typical $H_{maxCritical}$ value. The simulations go on till there are 5,000 peers on the overlay network and all the peers are satisfied or 1,500 simulator iterations are reached. The simulations were repeated for four different random networks generated by using different seed values for the random number generator on Linux.

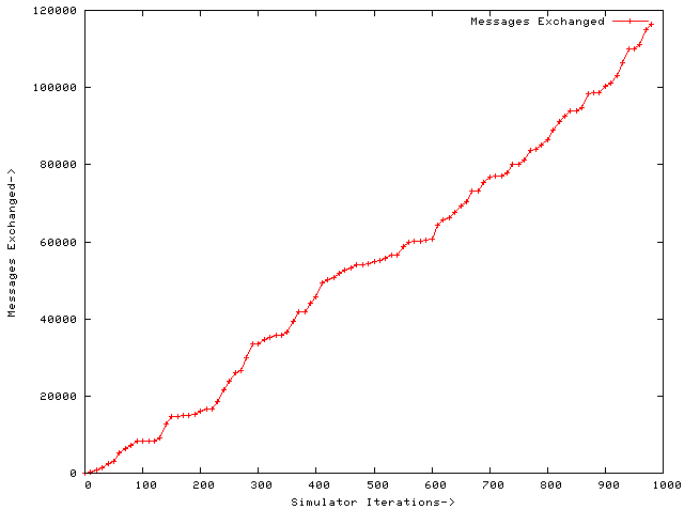

Fig. 3. A plot of total number of messages exchanged against simulator iterations, for simulations on a random network with 100 nodes initially and 5 nodes added every iteration. The simulations were done using a H_{max} value of 5. The simulations go on till there are 5000 peers in the overlay network and all the peers or satisfied or 1500 simulator iterations are reached.

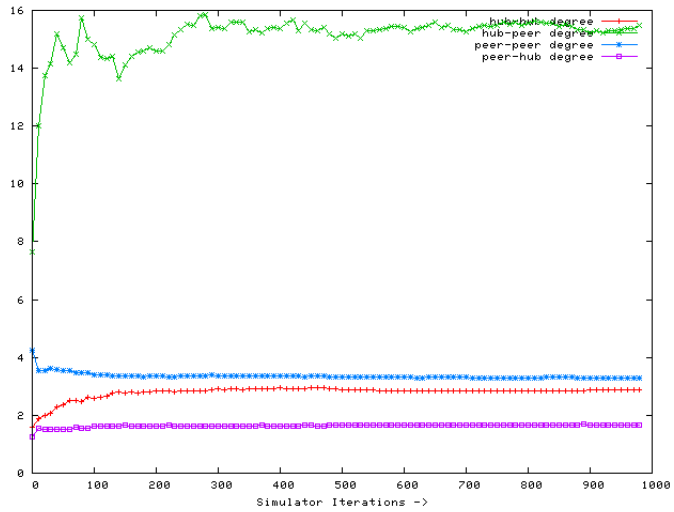

Fig. 4. A plot of hub-hub, hub-peer, peer-peer and peer-hub degree against simulator iterations, for simulations on a random network with 100 nodes initially and 5 nodes added every iteration. The simulations were done using a H_{max} value of 5. The simulations go on till there are 5000 peers in the overlay network and all the peers or satisfied or 1500 simulator iterations are reached.

Let **H** be the set of hubs and **P** be the set of peers on the overlay network. Let $\mathbf{E}_{H,H}$ be the set of connections connecting two hubs, $\mathbf{E}_{P,P}$ be the set of connections connecting two peers and $\mathbf{E}_{H,P}$ be the set of connections connecting a hub to a peer on the overlay network. In terms of these sets: hub-hub degree is defined as $\frac{|E_{H,H}|}{|H|}$, hub-peer degree is defined as $\frac{|E_{H,P}|}{|H|}$, peer-peer degree is defined as $\frac{|E_{P,P}|}{|P|}$ and peer-hub degree is defined as $\frac{|E_{H,P}|}{|P|}$.

Figure 4 shows a plot of hub-hub, hub-peer, peer-peer and peer-hub degree against time (simulator iterations) for simulations performed on one of the random networks. Throughout the simulations on an average each peer is connected to approximately 2 other hubs. When the simulations start each hub is connected on an average to approximately 2 other hubs. However within 100 simulator iterations the hub-hub degree changes to 3 and it stays at that value throughout the simulations, because of the hub algorithm.

Figure 3 shows a plot of the total number of messages exchanged to perform topology adaptation against time (simulator iterations) for the random network of Figure 4. The total number of messages exchanged (120,000) to perform topology adaptation may seem to be on the higher side. However for an overlay network with a long life it may be advisable to create an adaptive network of hubs by exchanging lots of messages. In the simulations the **search** operation was implemented without any caching. Caching can be used to improve the efficiency of the **search** operation thereby reducing the total number of messages exchanged.

The simulations on all the random networks converged within 1,000 simulator iterations. When the simulations converge all the peers are satisfied and the generated topology satisfies the constraints that we imposed in the satisfaction criteria. The simulations show that the hub algorithm can maintain an adaptive network of hubs even when there is a continuous arrival of new peers on the network.

IV. Conclusion and Future Work

The paper has demonstrated that Schelling's algorithm can be used for adapting P2P network topology. The abstract algorithm and the simulator presented can be used to develop and evaluate different variations of Schelling's algorithm. The paper presented a case study that demonstrates that an adaptive network of hubs can be created within a pure P2P network using a variation of Schelling's algorithm.

Currently a peer decides whether it should act as a hub or an ordinary peer. Future work could involve investigating the possibility of creating an algorithm based on abstract Schelling's algorithm, in which the peers mutually decide whether a peer should act as a hub.

References

[1] D. S. e. a. Milojicic, "Peer-to-peer computing," HP Labs, Tech. Rep., 2002.
[2] [Online]. Available: http://www.gnutella.com
[3] S. C. T., *Micromotives and Macrobehaviour*. Nortan and Company: W. W. Norton,, 1978.
[4] Schelling, "Dynamic models of segregation," *Journal of Mathematical Sociology*, 1971.
[5] U. Wilensky, "Netlogo segregation model." Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. [Online]. Available: http://ccl.northwestern.edu/netlogo/models/Segregation
[6] J. Rauch, "Seeing around corners," *The Atlantic Monthly*, April 2002.
[7] P. Adamic, Lukose and Huberman, "Search in power-law networks," *Physical Review Vol 64*, 2003.
[8] "Kaza," http://www.kazaa.com/us/index.htm.
[9] "Project jxta 20 superpeer virtual network," http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf.

[10] H. R. J. Gamma E. and R. V. J., *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.
[11] A. Singh and M. Haahr, "Topology adaptation in p2p networks using schellings model," workshop on Emergent Behaviour and Distributed Computing, PPSN 2004.