Abstract—Mobile agents are a powerful approach to develop distributed systems since they migrate to hosts on which they have the resources to execute individual tasks. In a dynamic environment like a peer-to-peer network, Agents have to be generated frequently and dispatched to the network. Thus they will certainly consume a certain amount of bandwidth of each link in the network if there are too many agents migration through one or several links at the same time, they will introduce too much transferring overhead to the links eventually, these links will be busy and indirectly block the network traffic, therefore, there is a need of developing routing algorithms that consider about traffic load. Mobile agents are autonomous, because the mobile agent is learning about the network as it progresses through it [1]. The mobile agent will visit nodes that were unknown when it was originally dispatched. At each node agent can make decisions such as invoke service, leave peer, or destroy itself based on its history of visited nodes and the current node. Information is being disseminated at every node that the mobile agent visits. Every node benefits from accepting a visiting mobile agent, because the mobile agent will have either new or more recent information about resources. Also, every mobile agent benefits from visiting a node because it will learn of either new or updated resources. If the mobile agents do not contain any new information they may be destroyed. In this paper, we propose an agent-based routing algorithm (ABRA), we investigate how agents should behave if they have to base their decisions on possibly known traffic information. In our model, we present agents migration processes based on decision tree on possibly extremist of interest of known traffic information, by analyzing, classifying network traffic attributes, and applying decision tree learning algorithm classified network flow data to get the best path that agent should migrate from node to node. The rest of this paper is organized as follows: We will first present background of network traffic analysis and classifying using naïve bayes technique, and overview of decision tree learning using ID3 algorithm in Section 2. In Section 3 related work will be introduced. The method is elaborated in Sections 4; section 5 will deal with simulation methods and experimental results. Conclusion and future work are presented in Section 6.

II. BACKGROUND

A. Network traffic analysis and classifying

Large quantities of network traffic flow data are generated on Peer-to-peer networks. These data contain information on the sources and destinations of individual flows encoded as IP addresses. The analysis of such data can reveal useful knowledge for traffic load, traffic cost, network situation, and network resource management. In order to perform analysis of network traffic flow data, naïve bayes technique [2] can be used, appropriate input data is needed, such as TCP flow, the classification process of input data requires parameterization of each object (dataset) to be classified. Using these parameters, classifier allocates an object to a class.
B. Naive Bayes Classifier Overview

The Naive Bayes Classifier technique is based on the so-called Bayesian theorem [3] and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.

To demonstrate the concept of Naïve Bayes Classification for network traffic attributes, consider the example displayed in the figure [1]. As indicated, the instance raw data of response time can be classified as either normal response time represented as a green color or low response time represented as red color. Our task is to classify new cases of response time value as they arrive, i.e., decide to which class label they belong, based on the currently exiting data. Since there are twice as many green color values as red color, it is reasonable to believe that a new case (which hasn't been observed yet) is twice as likely to have membership green color rather than red color. In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience, in this case the percentage of green color and red color objects, and often used to predict outcomes before they actually happen.

Thus, we can write:

Prior Probability for Green Color (Normal Response Time)

\[
\propto \frac{\text{NumberOfGreenColor}}{\text{TotalNumberOfObjects}}
\]

Prior Probability for Red Color (Low Response Time)

\[
\propto \frac{\text{NumberOfRedColor}}{\text{TotalNumberOfObjects}}
\]

C. Decision tree learning using ID3

In decision theory and decision analysis, a decision tree is a graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It can be used to create a plan to reach a goal. Decision trees are constructed in order to help with making decisions [4]. A decision tree is a special form of tree structure. Another use of trees is as a descriptive means for calculating conditional probabilities.

In agent migration, on the other hand, a decision tree is a predictive model; that is, a mapping of observations about a network situations to conclusions about the agent's migration path. More descriptive names for such tree models are classification tree or reduction tree. In these tree structures, leaves represent classifications and branches represent classes of network traffic attributes that lead to those classifications.

ID3

An early technique by the influential Ross Quinlan [5] that influenced a large part of the research on Decision Trees is useful to look at in order to understand basic decision tree construction.

Entropy

A measure used from Information Theory [6] in the ID3 algorithm and many others used in decision tree construction is that of Entropy. Informally, the entropy of a dataset can be considered to be how disordered it is. It has been shown that entropy is related to information, in the sense that the higher the entropy, or uncertainty, of some data, then the more information is required in order to completely describe that data. In building a decision tree, we aim to decrease the entropy of the dataset until we reach leaf nodes at which point the subset that we are left with is pure, or has zero entropy and represents instances all of one class (all instances have the same value for the target attribute).

We measure the entropy of a dataset, \(H\), with respect to one attribute, in this case the target attribute, with the following calculation:

\[
H = \sum_{i=1}^{c} p_i \log_2 p_i
\]

where \(p_i\) is the proportion of instances in the dataset that take the \(i^{\text{th}}\) value of the target attribute. This probability measures give us an indication of how uncertain we are about the data. And we use a \(\log_2\) measure as this represents how many bits we would need to use in order to specify what the class (value of the target attribute) is of a random instance.

III. RELATED WORK

Ichiro Satoh [7] presents a framework for building network protocols for migrating mobile agents over the Internet. The framework allows network protocols for agent migration to be naturally implemented within mobile agents and then dynamically deployed at network hosts by migrating the agents that perform the protocols. It is built on a hierarchical mobile agent system, called MobileSpaces, and several protocols for agent migration are designed and implemented based on standard protocols in the Internet, for example agent migration protocols through plain TCP, HTTP, SMTP or SSL, and application-specific routing protocols for efficiently migrating agents among multiple hosts.

Torsten Illmann, Tilman Krueger, Frank Kargl, Michael Weber [8] examined a migration technique of mobile agents in Java technology; they classified different migration styles and presented possible solutions and related work. They proposed classification distinguishes between code migration, execution...
migration and data migration. The classification defines a partial order to compare different migration approaches. For realizing strong migration in Java, two solutions are proposed. On the one hand, a pre-processor adds all the necessary information for migration to the source code before compilation time. On the other hand, A JNI-based plug-in for any virtual machine provides mechanisms to captures the agent’s execution state. The restoration of the execution state is done by the plug-in in combination with a byte code modifier which slightly changes the byte code of the agent.

Xiliang Zhong, Cheng-Zhong Xu, and Haiying Shen [9] presented a reliable connection migration mechanism that provides exactly-once delivery for all transmitted data during agent migration. It integrates with an agent-based access control mechanism that controls the access to network ports. To avoid frequent agent authentication and permission checking due to agent migration, a secret session key is associated with each connection. They present the design and implementation of the mechanism, named NapletSocket in Naplet [10] mobile agent system. It is a pure middleware implementation, requiring no modification of Java virtual machines.

Yingyue Xu, Hairong Qi [11] present two itinerary planning algorithms with the goal of maximising the information extracted while keeping resource usage to a minimum. The ISMAP determines the itinerary before dispatching the mobile agent while the IDMAP algorithm selects the route on the fly. They design three metrics (energy consumption, network lifetime, and the number of hops) and they used simulation tools to quantitatively measure the performance of different itinerary planning algorithms. Simulation results show considerable improvement over the ISMAP using the IDMAP itinerary algorithm.

Tino Schlegel, Peter Braun, Ryszard Kowalczyk [12], they suggested to apply the programming paradigm of mobile agents to reduce the network overhead by allowing agents to meet at the same network node before commencing communication. A remote communication between two agents could then be replaced by one or two agent migrations, followed by local communication. Since only in trivial cases it is possible to decide at design time whether remote communication or agent migration with subsequent local communication would perform better, this decision has to be made at run-time based on environmental parameters and agents’ past experience. Also they presented an adaptive approach, which is inspired by a solution of the El Farol problem [13]. Every agent forecasts the network load of the next communication step and applies a simple mathematical model to decide between the two alternatives at run-time.

IV. METHOD

A peer-to-peer system is a distributed system whose component nodes participate in similar roles, and are therefore peers to each other. Peer-to-peer can be viewed as decentralized network architecture.

Our algorithm focuses on how the mobile agents will behave while they roam around the network and can be sketched as follows:

1. A user accesses the system and places service requests specifying some criteria.
2. Mobile agents are periodically generated by its peer group; they room inside its peer group, collect and disseminate information
3. The data state contains the information carried by the MA during migrations obtained when each peer periodically sends a mobile agent, by broadcasting replicas of it to each neighbor node, and giving a path and an extremist of interest, the path identifies a subtree, e.g. path "abcd", path "abcf", or path "abece" as seen in Figure 2. The extremity of interest obtained by analysis, classify the network flow data attribute for each subtree.

Fig. 2 Mobile agent created by its peer

4. The main approach that is used to classify network traffic flow data is the Naïve Bayesian technique. Consider a collection of flows x = (x1,……, xn), where each flow xi is described by m discriminators{ d1(i),……, dm(i)} that can take either numeric or discrete values. In the context of the network traffic c, dj(i) is a discriminate of flow xi, for example it may represent the mean interval time of packets in the flow xi. In this paper, flows xi belongs to exactly one of the mutually-exclusive classes described in Section 2. The supervised Bayesian classification problem deals with building a statistical model that describes each class based on some training data, and where each new flow y receives a probability of getting classified into a particular class according to the Bayes rule below,

\[
p(cj \mid y) = \frac{p(cj)f(y \mid cj)}{\sum_{cj} p(cj)f(y \mid cj)}
\]

where \( p(cj) \) denotes the probability of obtaining class \( cj \) independently of the observed data, \( f(y \mid cj) \) is the distribution function (or the probability of \( y \) given \( cj \) ) and the denominator acts as a normalising constant,
Table 1 demonstrates the classification accuracy of sample data representing classes of network attributes such as: Round Trip Times, Throughput and Packet Loss, they are very well classified.

5. After classifying network traffic flow data, decision tree learning using ID3 algorithm will be applied to define the interest of extremist, that illustrate the best path, which agent should migrate from node to node.

First, we calculate the entropy for each class:

**Round Trip Time**:

$$H = \frac{1}{6}(-\left(\frac{1}{1}\log_2 1\right) + 4\left(-\frac{2}{4}\log_2 4\right) - \frac{1}{4}\log_2 2) = 0.462$$

**Throughput**:

$$H = \frac{3}{6}((-\frac{2}{3}\log_3 3) - \left(-\frac{1}{3}\log_3 1\right) + \left(-\frac{1}{3}\log_3 1\right)) = 0.318$$

**Packet Loss**:

$$H = \frac{5}{6}((-\frac{3}{5}\log_5 5) - \left(-\frac{2}{5}\log_5 2\right) + \left(-\frac{1}{5}\log_5 1\right)) = 0.56$$

Thus we select the class throughput as the first decision node since it is associated with the minimum entropy. This node has two branches normal and low. Under the branch low, only path A-B-C-D fall, and hence no further discrimination is needed. Under the branch normal, we need another attribute to make further distinctions. So we calculate the entropy for the other two attributes under this branch:

**Round Trip Time**

$$H = \frac{1}{3}(-\left(\frac{1}{1}\log_3 1\right) + \left(-\frac{2}{3}\log_2 2\right)) = 0$$

**Throughput**:

$$H = \frac{3}{3}(-\frac{2}{3}\log_3 2 - \frac{1}{3}\log_3 1) = 0.636$$

Thus the decision tree path that explains the extremist of network traffic flow data class is configured as illustrated in figure [3]. Figure [4] illustrate the best path that agent should migrate.

6. Each mobile agent while traveling collects, and carry path information, once an agent reaches a peer, the peer tells it whether the request can fulfilled in its peer group, if so the mobile agent goes back to the source peer along the same route it has explored, updates the routing table along its route. Thus each peer in the network maintains current routing information for reaching nodes outside its neighborhood.

Table 2 gives important notations of the agent-based routing algorithm, and it can be sketched as seen in algorithm [1].
### TABLE II

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ξ(path)</td>
<td>Returns a set of all classes managed by a peer with path path</td>
</tr>
<tr>
<td>π(p)</td>
<td>Returns the path of peer P</td>
</tr>
<tr>
<td>B</td>
<td>Set containing all paths between the higher and the lower bound of information contentment</td>
</tr>
<tr>
<td>AVG_ENTROPY</td>
<td>Return -∑p(v)log2(p(v)) , where 0.log2(0)=0</td>
</tr>
<tr>
<td>CommonPrefix</td>
<td>Returns a set of all classes managed by a peer</td>
</tr>
<tr>
<td>Index</td>
<td>Number of common data</td>
</tr>
<tr>
<td>ζ</td>
<td>Common prefix of interest</td>
</tr>
</tbody>
</table>

### Algorithm 1. Agent Based Routing Algorithm

```java
//Part 1, input data, classification attribute
Return decision tree
For each attribute compute AVG_ENTROPY is minimal
Part 2
1 if compath < mA.getIndex() then
2 5 sendBadRequest;
3 7 Return;
4 8 endif
5 // part 3
6 10 if compath < localPath.length AND compath < path.length then
7 12 setIndex(mAgent, compath);
8 13 peers = randomize(getPeersAtLevel(compath));
9 14 for peer = peers.getFirst(); peer ≠ localPeer; peer = peers.getNext() do
15 16 if NOT online(peer) then Continue;
16 17 correct = Send(peer, mAgen);
18 19 else if correct then return;
20 18 else Continue;
21 19 endif
22 20 reply = new PeerNotFound();
23 21 Send(mAgen.getInitialPeer(), reply);
24 22 Return;
25 23 else if compath < localPath.length AND compath == path.length then
26 24 for level = mAgen.getIndex(); level < maxLevel; level++ do
27 25 peers = randomize(getPeersAtLevel(compath));
28 26 for peer = peers.getFirst(); peer ≠ localPeer; peer = peers.getNext() do
29 27 if NOT online(peer) then Continue;
30 28 else if (isOnTheLeftOf(peer, localPeer) AND mode == LEFT) OR (isOnTheRightOf(peer, localPeer) AND mode == RIGHT) then
31 29 setIndex(mAgen, level);
32 30 correct = Send(peer, mAgen);
33 31 if correct then return;
34 32 else Continue;
35 33 endif
36 36 endif
37 37 Break; 38 endif
39 40 endfor
40 // The local peer should be the one we are looking for
41 reply = new Reply(localPeer);
42 Send(mAgent.getInitialPeer(), reply);
```

V. SIMULATION STRUCTURE

In order to validate our algorithm, we present some simulation results from omnet++ discrete event simulation systems [14], we compared it with di caro and dorigo’s antnet [15] that use similar forward agents but with more complex back-propagating agents and a different routing table updates procedure. Their antnet system has been shown to outperform OSPF [16]. In this section we use a simulated network that is composed of two cases, the first one is when no of agents are greater then no of agents, and the case two is when no of agents are greater than no of nodes. Table [3] summaries network parameters used to evaluate a simulation method.

![Time delay at no of nodes from 3 to 20 and 5 agents](image-url)

To evaluate the result of simulation we used two measurements: average delay on the network and throughput. The average delay experienced by all the packets that arrived during a given interval of time gave another view of the general status of the network. The throughput is the number of packets that went through the network during one unit of time. Figure [4] compares the average packet delay and throughput for agent based routing algorithm (ABRA) and OSPF. In figure [4,a] it is clearly show that ABRA algorithm is much better than OSPF when the number of nodes is increased the time delay in ABRA is lower than OSPF. Also in figure [4,b] throughput in ABRA consumed little time than OSPF. Case two represented in figure [4,c,d] time delay and throughput in ABRA consume much time than OSPF when numbers of agents and nodes small, when numbers of agents and nodes increased ABRA gives better performance than OSPF.
VI. CONCLUSION AND FUTURE WORK

In this paper we have introduced a mobile agent migration technique based on decision tree learning for P2P networks. The proposed solution in this work has the objective of a agent based routing algorithm (ABRA). We defined a most extremist path (subtree) for each link based decision tree learning algorithm, it was applied after analyze and classify network traffic flow data for each peer by using naïve baye technique, and this approach gives the best path, that agent should migrate from node to node. Figure 5 is a block diagram which illustrates the main processes of our technique.

**Fig. 5 Main Process of ABRA**

Future work will focuses on enhanced mobile agent performance in distributed system environment, while web services and peer-to-peer are considered the ideal distributed system technology [17]; we will study interaction of mobile agent with web services in semantic approaches [18]. Based on information theory in capacity channel we propose a layered communication protocol in which the agents gradually build towards a semantically integrated system by establishing minimal and effective shared ontologies [19].

**REFERENCES**


[7] Ichiro Satoh "Network Processing of Mobile Agents, by Mobile Agents, for Mobile Agents" presented at Mobile agents for telecommunication


[13] Ann M. Bell, William A. Sethares" The El Farol Problem and the Internet: Congestion and Coordination Failure"


[16] Uwe Röttgermann "Decentralized Throughput Optimization in Industrial" presented at Institute of Information and technology, Munchen University, Germany.
http://deposit.ddb.de/cgi-bin/dokserv?idn=978930878&dok_var=d1&dok_ext=pdf&filename=978930878.pdf

