ABSTRACT
The emerging growth of computer networks and the great influence of Internet on various aspects of trade made using users’ information to improve the performance of systems an ordinary task. Currently almost all the user modeling frameworks catch the required information from the application layer, thus just the information providers could use them. As a result, these frameworks have no benefit for the in-between nodes such as ISPs which may access these data in network layer. In this paper we propose a new multi-agent framework which can be used for modeling the users by network layer information. This framework can help the in-between nodes to arrange their service policies based on different user models. Although using intelligent agents and artificial intelligence techniques, the framework extracts the hidden relations between network layer behaviour components and application layer behaviour components, thus improving the modeling performance. Moreover we have presented a case study which models an ordinary ISP’s users and shows how the model can be used to improve the ISP’s quality of service.

KEY WORDS
User modeling, network layer, multi-agent systems, artificial intelligence

1. Introduction
The emerging growth of computer networks and the great influence of Internet on various aspects of trade made using users’ information to improve the performance of systems an ordinary task [1] [2]. Website personalization, targeted advertising, guided search, buy suggestion and a lot of other services currently suggested by websites are just some examples of what user modeling can bring us [3]. Generally three approaches exist in using users information, including user modeling, user clustering and user classification [4]. User modeling allows systems to properly adapt their behaviour with each user properties. The user specific system behaviour allows the system to catch the user requests behind what explicitly has been announced to the system [5]. Different methods such as hierarchical approaches and hidden markov models are used for user modeling [6]. Also we can use this information to cluster the users into some unknown classes. This can be done using methods such as ART neural networks, K-Means, or any other clustering algorithm [7]. These clusters have been used in different applications such as service priority assessment and data prefetching [7]. In user classification, users are classified regarding some predefined classes. A wide variety of algorithms including neural networks, learning classifiers and fuzzy classifiers have been used for this purpose [8].

Generally almost all the proposed methods use application layer information, so they can’t be used by in-between nodes which just have access to the information in network layer and see the data flow as some external traffic. Although a lot of works have been done in the field of network layer information analysis, but these works normally aim modeling some network traffic parameters [9-11], such as estimating a stochastic model for packet flow [12][13]. Also some works use user models to simulate the network traffic generation [14][15], but again the aim of these works is to model the network, not the users. Generally in these works the user models are predefined, i.e. we know exactly how specific user types generate network traffic, and then the simulation runs.

In this paper we propose a new multi-agent framework which can be used for modeling the users by network layer information. This framework can help the in-between nodes to arrange their service policies based on different user models. Although using intelligent agents and artificial intelligence techniques, the framework extracts the hidden relations between network layer behaviour
components and application layer behavior components, thus improving the modeling performance.

In the remaining of this paper, Section 2 presents the related works and tries to show where the proposed work stands among others; Section 3 describes the proposed multi-agent framework and explains how the framework works; a case study comes in Section 4; finally, Section 5 contains an overall conclusion and suggests further works.

2. Related Works

This section gives a brief review of general user modeling systems and also presents a deeper look at some user modeling works that aim modeling some network traffic parameters. In the remaining, the first part describes different viewpoints of user modeling and also enumerates the available general user modeling systems. After that, the second part discusses the previous works in the field of using network layer information for user modeling and network traffic generation.

2.1 User Modeling

User modeling can be investigated from three main approaches: modeling user knowledge, modeling user plans, and modeling user preferences [16].

User knowledge modeling is one of the first approaches to user modeling. It can be used in application areas in which a quick but not necessarily very accurate estimation of the user’s background knowledge is required. In this approach, generally the system developer must perform three tasks: user subgroup identification, in which the developer must define subgroups with homogeneous relevant characteristics; identification of key characteristics, in which the developer must choose some key characteristics that can be used to associate users to subgroups; and representation in hierarchically ordered structure, which brings a formal representation system to show the probable relations between subgroups [17].

User knowledge modeling has been used for information giving and suggestion making purposes [17-19]. As the system presumptions about user models is error prone, it has been suggested to let users correct their model themselves [19][20]. Evidently it’s a hard work to do for large systems, besides the fact that representing the presumptions in a user friendly way is very hard [21].

Modeling user plans investigates to find what sequence of actions is taken by each user to achieve an aim. In this approach, the system tries to guess the user goal based on the actions taken till now. As the user approaches are always subject to change, this approach is hard to be done [22]. Two general methods have been used for modeling user plans, plan library and plan construction. The plan library method is not applicable to large problems, and the computational complexity of plan construction method is very high when we can’t put a restriction on the plan length [16]. Thus, recently heuristic approaches which use the developer domain knowledge have gained more attention [23][24].

Modeling user preferences has been investigated from mid 90’s. Generally the proposed methods are heuristic and use the developer domain knowledge. This approach has been used in different application areas such as intelligent information retrieval [25], query enrichment [26], information filtering [27], and adaptive hypertext systems [28].

Also there are some user modeling shell systems which provide integrated representation, reasoning, and revision tools that form an “empty” user modeling mechanism [16]. Most of these systems use hierarchical user representation models. UMT [29] uses a hierarchical structure of user classes. In this system a rule base is used to store inference rules. The rules are applied till no applicable rule can be found and the user model is created. BGP-MS [30] is another sample of shell systems which uses hierarchical representation. It uses first order logic to state the system presumptions for user groups and inference about them. TAGUS [31] is similar to BGP-MS with the advantage that can track the user behavior changes and associate a combined model to the user. Doppelganger [32] is another sample of shell systems which uses markov models and clustering algorithms. All these systems use the user information in application layer.

2.2 User Modeling and Network Layer Information

Many researches have been done in the field of network layer information analysis and network traffic simulation. In most cases the modeling is based on network resources, such as finding a stochastic process which indicates packet arrival time or packet size [33]. In spite of these researches, user-based traffic generators try to model the user action sequence in a higher level. Because each higher level action results in a sequence of network layer events, typically these models use a hierarchical representation scheme [34][35]. Using markov models is another approach to model the user action sequence.
In these models user behaviour is inspected regarding time, data flow size, and traffic transfer port [36][37]. Also in some researches where network traffic model was investigated to analyse data transfer protocols, the protocol used for data transfer is considered [34]. It is worthy to note that in these researches the main goal is to model the network traffic, not the network users. In other words, in spite of considering what users do or how their actions are interrelated with the user model, the model of traffic which each user (or user group) generates is considered.

3. Proposed Multi-Agent Framework

In this section we describe our proposed multi-agent framework. The first part describes the overall model structure and how main components are cooperate. In the second part the intelligent agents and their components are described. At last the third part presents the modeling engine and its components and describes how the modeling procedure proceeds.

3.1 Framework

The proposed framework contains two main components: application layer intelligent agents and the network layer user modeling engine. Figure 1 depicts the framework structure and the relation between intelligent agents and the engine. Application layer intelligent agents are software agents which can access to users’ information in application layer. These agents can be used to facilitate the train phase of user modeling engine. The procedure is as follows: first, we run just the agents and train them using application layer information and one of known methods; then we run the engine and let it to take the users network layer information from the network and their model from the agents which at the same time are running. This lets the engine to create a data set which is composed of user models in different times and associated network layer information. This information could be of a wide range of forms, from some statistical indicators calculated on packet properties to some packet sequences. Besides this, using a supervisor can enhance the engine train set quality more.

The user modeling engine analyses network layer information and creates user models regarding the analysis results. As a result of low computational power or encrypted sessions, the engine may be not capable to track all the user behaviour components and being restricted to some information such as number of sessions, size of packets, etc. But because the intelligent agents have access to the application layer information, virtually they can grab all the user behaviour components and create a more exact model than the engine. So in the training phase the engine requires the agents or a prepared training set to proceed.

After the training phase, the engine will be capable to do the user modeling task without using any aid from the agents. Of course even in this phase if the agents were running, they can help the engine to improve its performance and create more exact models.

3.2 Intelligent Agents

As it can be seen in Figure 2, application layer intelligent agents must deal with two separate environments. The first is the application area in which user applications are running, and the second is the network area. These agents always spy running programs and at the same time try to analyse the transferred information to detect what the user is doing.

The agents’ knowledge of running programs (for example peer to peer programs) can be assumed as a supervisory mechanism which lets the agents to learn how to create an exact user model by just using the transferred information. Whether they succeed or not, a report must be sent to the network layer user modeling engine.

The agent receives the transferred information from its Network Interface, which sends the packets to the Feature Extraction component. This component extracts each packet properties which could be used in modeling procedure and sends them to History and User Modeling Core. History archives the provided features so the sequence of features always is ready for analysis.

User Modeling Core investigates the features received from Feature Extractor, probably a history of previous seen features, and also the current user
model which is brought by Model Library component and then builds the new user model. This model propagates to the components System Trainer, System False Detector, and Server Interface. System Trainer and System False Detector use the Application Watchdog hints to justify the estimated user model. If the model is wrong, System Trainer tries to modify the modeling procedure so the model could be correctly estimated in next occurrence and System False Detector sends an error report to the Server Interface. The Server Interface is the communication port which is used to communicate with the network layer user modeling engine.

3.3 Network Layer User Modeling Engine

If we separate the learning phase from the utilization phase, the user modeling engine components can be divided into two groups: the general components which are always required, and the learning-specific components which are just required in the learning phase.

Learning specific components include Agent Interface and System False Detector. All the remaining components are always required. The engine structure is depicted in Figure 3.

As it can be seen by comparing figures 2 and 3, the engine has some common components with intelligent agents. So we will describe just the components which are new or have a different application.

The Agent Interface receives the model suggestions from intelligent agents. As said before the component can be ignored during the utilization phase. The Policy Maker is responsible to enforce the network policies based on each user model. The User ID Extractor is responsible for associating a unique ID to each user. This may be done using some hardware identifiers. The ID will be used entirely to refer to the user. After User Modeling Core selected the appropriate model for the user, it sends it to the Model Library component.

4. Case Study: ISP

In this section a case study is examined to investigate the framework usability. The case study discusses the framework usage in modeling users for an internet service provider. We study this problem in three steps respectively: user subgroup identification, identification of key characteristics, and learning the engine.

4.1 User Subgroup Identification

Before identifying user subgroups, we must specify why we want to model our users. In this case the user modeling is intended to let the ISP manager to provide a higher quality of service (or a cheaper service) for some users such as academians and researchers and prevent some usages such as peer to peer. Due to this scenario, first we must specify the
user classes and our policy against each. Consequently, our user modeling problem is a user classification one in which we like to use user models to associate each user to one or more predefined classes. Regarding the ISP policy making preferences, we use these user behavior models: hack, pornography, academic, chats, file sharing, download, remote procedure call, and news reading.

4.2 Key Characteristics Identification

Now we must continue with identification of each user model key characteristics. The characteristics may be defined as some primitives or may be calculated using statistical methods or other feature extraction routines. In this problem, we used some heuristics to specify each class characteristics. It must be mentioned that the characteristics must be network layer characteristics, otherwise they couldn’t be used by the network layer user modeling engine.

**Hack:** Typically different protocols such as ICMP, HTTP, SMTP and FTP are used; the user sends information to some specific addresses (victims); the send and receive bandwidths are nearly equal; the user behavior against others isn’t a normal behavior, i.e. sometimes it sends a lot of requests in a little time (such as DoS) and sometimes keeps a connection for a nearly long time (for example when the victim has been hacked). Besides these general characteristics, different key characteristics may be defined for different attacks.

**Pornography:** Usually uses just HTTP; a lot of images transferred during each session, and some large images are downloaded after; the request text is full of suspicious or adult words; the number of requests in a specific time interval is very high; user isn’t bounded to a specific web server and moves frequently between different servers; servers don’t belong to well known organizations; download bandwidth is very higher than upload bandwidth; encrypted sessions may be used often.

**Academic:** Usually uses a lot of HTTP requests; a noticeable part of the request go to famous search engines; many files downloaded, generally documents (.pdf, .ps), source codes (.c, .java), archives (.zip, .gz), and executables (.bin, .exe); some times the response is a long text including some pictures; requests may include some suspicious words, but it isn’t a general behaviour; rarely uses encrypted sessions; users may refer often to some specific sites; a lot of requests may be sent to academic sites (.edu); usually the download bandwidth is very higher than the upload bandwidth.

**Chats:** Generally known protocols and ports are used; transferred packets are small; download and upload bandwidths are nearly equal and low; connections are long time; the packet source or destination belong to big companies such as Yahoo or Microsoft.

**File Sharing:** Generally known protocols and ports are used; the used bandwidth is high; although the ratio of download and upload bandwidth isn’t fixed, it can be calculated for each user; a lot of connections are used; typically the destinations are unknown and nameless.

**Download:** Typically uses FTP or HTTP; a lot of requests sent to specific destination addresses; generally small responses; low bandwidth usage; fully known and standard headers.

**Remote Procedure Call:** Usually uses HTTP; a lot of requests sent to specific destination addresses; generally small responses; low bandwidth usage; usually long time; the packet source or destination belong to big companies such as Yahoo or Microsoft.

**News Reading:** Generally uses HTTP; nearly small responses, generally in HTML format; low bandwidth in average, but may use a high bandwidth for a short time; usually some pictures transferred.

4.3 Learning the Engine

After identifying the key characteristics, we must create a dataset. To do this, we have used information from users which their usage model was pre-known. The users requested to choose a usage model and then do what they want. For example we asked several researches to do their usual task within a predetermined time. We have logged the network layer information and then labeled the data as belong to academic dataset. This procedure repeated for the other user groups.

After making the dataset ready, we must specify the engine learning system. In this case study we have used a decision tree classifier as the user classification engine. At first we have converted heuristic rules mentioned in part 4.2 to an ordinary classification engine. At first we have converted heuristic rules mentioned in part 4.2 to an ordinary decision tree. After that some parts of the dataset have been selected as the train set and have been used to adjust the decision nodes weights. The remaining of the data set have been used as the test set which the network layer user modeling engine performance over them is reported in Table 1.

As Table 1 show, the engine performance for detecting Pornography, Academic, Chats, File Sharing, Download and Remote Procedure Call is acceptable, but it has not an acceptable rate for Hack and News Reading. This can be described regarding the facts that hackers’ behavior is complicated and it is hard to be detected just using network layer information. Also news reading is very similar to a normal web surfing behavior, so the true positive ratio is low. Again this similarity to general behavior causes the false positive ratio to rise.
5. Conclusion and Future Works

In this paper we have proposed a new multi-agent framework for user modeling via network layer information. As it has been shown in the case study, the proposed framework can be used efficiently to model the users.

Future works shall include investigating more artificial intelligence techniques. Also employing fuzzy clustering to cover the multi-class user models is another subject to research.

References


<table>
<thead>
<tr>
<th>User Group</th>
<th>True Positive</th>
<th>False Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hack</td>
<td>36.7%</td>
<td>8.9%</td>
</tr>
<tr>
<td>Pornography</td>
<td>87.6%</td>
<td>6.2%</td>
</tr>
<tr>
<td>Academic</td>
<td>91.2%</td>
<td>3.6%</td>
</tr>
<tr>
<td>Chats</td>
<td>85.4%</td>
<td>5.4%</td>
</tr>
<tr>
<td>File Sharing</td>
<td>79.7%</td>
<td>11.3%</td>
</tr>
<tr>
<td>Download</td>
<td>81.2%</td>
<td>18.7%</td>
</tr>
<tr>
<td>Remote Procedure Call</td>
<td>77.2%</td>
<td>13.2%</td>
</tr>
<tr>
<td>News Reading</td>
<td>55.2%</td>
<td>25.7%</td>
</tr>
</tbody>
</table>

Tab. 1 User modeling engine performance