PrefSIENA

In the context of the AEOLUS project, we implemented a publish/subscribe functionality for the overlay computing platform enhanced with ranking capabilities.

Generally, in publish/subscribe systems, users describe their interests via *subscriptions* and are *notified* whenever new interesting events become available. Typically, in such systems, all subscriptions are considered equally important. However, due to the abundance of information, users may receive overwhelming amounts of events. In our work, we propose using a *ranking mechanism* based on *user preferences*, so that only *top-ranked* events are delivered to each user. Since many times top-ranked events are similar to each other, we also propose increasing the *diversity* of delivered events.

Concerning the ranking of events, we have implemented a number of different delivering policies for forwarding ranked events to users, namely a *periodic*, a *sliding-window* and a *history-based* one. Our implementation is based on the SIENA notification Service [1, 2] and is available for download [3].

Using PrefSIENA

The notification service consists of a number of servers, organized in a hierarchical manner. The publishers, that provide events to the system, and the subscribers, that enter subscriptions and consume events, connect to some server in the hierarchy and submit their publications and subscriptions respectively.

Servers

A PrefSIENA Server can be started as follows:

```
StartServer -id <hostname>
            -<protocol>
            -port <port>
            -k <k>
            -mode <mode>
            -period_length <length>
            -window_size <size>
            -diversify <true|false>
```

where <hostname> is a name used to identify each server, <protocol> is the protocol used to contact the server, i.e. tcp, udp, or ka (a “keep alive” mechanism), <port> is the port the server listens at, <k> is the number of top-results delivered to the subscribers of this server, <mode> is a parameter determining the delivery policy (1-sliding-window, 2-periodic or 4-history-based), <length> and <size> determine the length and size of each period a window respectively and diversify is a boolean argument determining whether result diversification is applied or not. For example:

```
start java –cp prefsiena.jar siena.StartServer –id myHost –ka
           -port 2345 –k 10 –mode 1 –window_size 100
           -diversify false
```

Publishers

Publishers connect to a PrefSIENA Server and upload their publications which are stored in a local file. They keep running as long as they have more publications to submit. A publisher can be called as follows:

```
Publisher   <protocol>:<hostname>:<port>
            <publications filename>
            <intra-publication time>
```

Where `<protocol>:<hostname>:<port>` are the contact details of a PrefSIENA Server, `<publications filename>` is the file containing the publisher's publications and `<intra-publication time>` is the time between two consequent publications in milliseconds. For example:

```
java –cp prefsiena.jar Publisher ka:myHost:2345 publications.txt 50
```

Events are sets of typed attributes. Each event consists of an arbitrary number of attributes and each attribute has a type, a name and a value. Attribute types belong to a predefined set of primitive types, such as "integer" or "string". Attribute names are character strings that take values according to their type. Formally, an event *e* is a set of typed *attributes* $\{a_1, \ldots, a_p\}$, where each a_i, $1 \leq i \leq p$, is of the form ($a_i.type$ $a_i.name$ | $a_i.value$). Publication files should be formatted as follows:

```
genre|Comedy
length|103
mpaa|R
rating|6.1
title|Everlasting Piece, An
votes|475
year|2000

genre|Comedy
length|94
rating|3.4
title|Fantasm Comes Again
votes|13
year|1977
.
.
.
```

Subscribers

Subscribers connect to a PrefSIENA Server and upload subscriptions stored in a local file. Then, they keep running, waiting to be notified about their top-ranked events. A subscriber can be called as follows:

```
Subscriber  <protocol>:<hostname>:<port>
            <subscriptions filename>
            <result filename>
            <waiting time>
            <sorted>
            <skyline>
```

where <protocol>:<hostname>:<port> are the contact details of a PrefSIENA Server, <subscriptions filename> is the file containing the subscriber's subscriptions, <result filename> is the filename where the received results and relative statistics are stored and <waiting time> is the time the subscriber will remain alive listening for notifications in milliseconds. <sorted> is an optional argument that, when used, orders the results in <result filename> according to their importance instead of their time of arrival. <skyline> is an optional argument that, when present, ignores received notifications that are dominated by past ones. For example:

```
start java –cp prefsiena.jar Subscriber ka:myHost:2345
subscriptions.txt result.txt 15000 sorted
```

Concerning the format of subscriptions, each subscription consists of a set of constraints on the values of specific attributes and a numeric score indicating user preference. Each attribute constraint has a type, a name, a binary operator and a value. Types, names and values have the same form as in events. Binary operators include common operators, such as, =, <=, <, > and * (substring). Formally: A *subscription s* is a set of attribute constraints $\{b_1, \ldots, b_q\}$, where each b_i, $1 \leq i \leq q$, is of the form ($b_i.type\ b_i.name\ \vartheta b_i\ b_i.value$), $\vartheta b_i \in \{=, <, >, \leq, \geq, !=, *, *<, *>\}$. Subscription files should be formatted as follows (when no operator appears, the operator "=" is used):

```
year > 1980
genre * Comedy
score 0.87

year < 2000
length > 100
mpaa R
genre * Drama
rating > 7
score 0.78
.
.
.
```

The received notifications for each subscriber are presented to the screen and also stored in <result filename>. The form of the received notifications is the following. All attributes of the corresponding event are presented to the user along with a publication timestamp and the notification's score according to the user's preferences [4]:

```
genre="Drama" length=121 mpaa="PG-13"
rating=7.1 title="Luther" votes=1567
year=2003} "2009.02.26 12:40:55 EET"
"2009.02.26 13:40:55 EET" 0.92
```

References

[1] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. *Design and evaluation of a wide-area event notification service*. ACM Trans. On Computer Syst., 19:332–383, 2001.

[2] SIENA. http://serl.cs.colorado.edu/~serl/dot/siena.html.

[3] PrefSIENA. http://dmod.cs.uoi.gr/aeolus_site.

[4] M. Drosou, K. Stefanidis, and E. Pitoura. *Preference-Aware Publish/Subscribe Delivery with Diversity*. In DEBS, 2009.