Experimentation aware peer to peer client (P2NER) for live video streaming

P2ner is a P2P client that is used for transmission of live video. P2ner can act both as a producer of a video (publisher) and as a consumer (subscriber). The architecture of our client is developed in such a way to facilitate experimentation. In P2ner a user can become a publisher and exploit the following capabilities:

1. Selection of the desired stream. This stream can be a live stream captured by a camera, any video file or any third party live stream transmitted by any streaming protocol (i.e. rtp, http, udp).

2. Adjustment of the quality of the transmitted stream.

3. Selection of the input component which is responsible for trans-coding the stream to the proper format and quality. Right now there are two input components available. One based on gstreamer and one based on Vlc.

4. Set of the P2P parameters which will be used for the transmission relevant with scheduling and media distribution graph architecture.

5. Register the stream and its parameters to a P2ner Server. This server is responsible for storing the available streams along with the addresses of the publishers and acting as an entry point for the P2ner clients.

In case that a user is a consumer is able to: browse the available streams stored in various P2ner Servers and subscribe to the desired one, select the appropriate output component which is responsible for playing the stream. Again, there is an output component based on gstreamer and one based in Vlc, while there is also the option to just discard the downloaded stream. Based on the parameters of the stream, the appropriate scheduler and overlay components are loaded automatically by the client, given that they are installed, requiring no intervention by the user.

Figure 1: P2ner user interface (left) and experimentation interface (right).

P2ner is written in python using the twisted framework. Its architecture, depicted in the previous figure, is component based, where each component has a well defined API, making it extremely modular, as every component can be developed independently and then just be installed in the client in order to be used. This is accomplished by using the setuptools module of python which allows for the definition of entry points for each component and for the seamlessly installation of them in the main program.

The core of the P2ner client is the P2P Engine component. This component is responsible for loading and connecting the initial components required for the client to run, and for dynamically loading the
appropriate components for the producing or consuming of a stream.

In order to facilitate experimentation, P2ner client can be controlled either locally or remotely. On the first case, the P2P engine loads the local control component along with the client (command line interface) or the GUI component with its local interface, according to the user preferences. The GUI component is written in GTK2. In the latter case, the P2P engine loads just the XMLRPC control component, while in the remote PC the user can either launch the same GUI component but with the XMLRPC interface or use the WEB interface. In either case, the remote communication is based on remote XML RPC calls.

![P2ner software architecture](image.png)

**Figure 2: P2ner software architecture**

The other components that the P2P engine loads initially are the network components and the various plugins. The network component is divided in the control channel, used for the transmission of control messages, and the data channel used for data transmissions. Both are using UDP as the underlying transmission protocol. In the control channel there is a sub-component which handles the acknowledgement and retransmission of critical control messages, while in the data channel there is a flow control sub-component for the best utilization of the available bandwidth.

When a user publishes a stream the P2P engine constructs on the fly the appropriate published stream component. This component consists of the sub-components that were described previously. Analogously, when the user subscribes to a stream the P2P engine constructs the subscribed stream component according to the stream's parameters. The same network components are used for every subscribed or published stream.

For more information and communication with us please visit - [p2ner wiki][1]

```javascript
var _gaq = _gaq || []; _gaq.push(['_setAccount', 'UA-35307132-1']);
_gaq.push(['_trackPageview']); (function() { var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true; ga.src = ('https:' == document.location.protocol ? 'https://' : 'http://www') + '.google-analytics.com/ga.js'; var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s); })();
```


**Links:**