process one
INSTANT MESSAGES | CREATIVE BUSINESSES

OneTeam Media Server

2009, 30th april

Mickaël Rémond <mremond@process-one.net>

process one
INSTANT MESSAGES | CREATIVE BUSINESSES

An Erlang company

ProcessOne delivers scalable and robust systems to support creative Instant Messaging applications www.process-one.net

ProcessOne: An Erlang company

- 20 people, most of them Erlang developers
- Lots of Erlang Veterans (Two «Erlang User of the Year !»)
- Focusing on software development and technology for which Erlang is a good fit.
- World wide customer base: We are spreading Erlang use all around the world.
- Main customers base:
  - Social networks
  - Carrier / Telco

process one
INSTANT MESSAGES | CREATIVE BUSINESSES

~50 major customers
~45 millions of users

Software products: ejabberd

- Instant messaging platform based on the open XMPP protocol.
- Open source software. Lots of example of Erlang code.
- Largely used in the world and usefull to convert developers to Erlang and gain mind share.
- About 40% market share of the public XMPP known domains.
- Known for its clustering, scalability and hackability (easy to extend, live code update)
- Very large deployment in production
- Require some know-how to reach large scale, but strongly supported solution by ProcessOne
- Very large development community

Software products: Tsung

- Highly scalable benchmark tool: Can easily simulate lots of users on a cluster (hundreds of thousands)
- Many supported protocols:
  - HTTP/HTTPS
  - XMPP
  - Postgres
  - LDAP
  - ...
- Extensible: You can add new protocols.
- Strong support by ProcessOne.

process one
INSTANT MESSAGES | CREATIVE BUSINESSES

OneTeam Media Server

ProcessOne delivers scalable and robust systems to support creative Instant Messaging applications www.process-one.net

OneTeam Media Server

- This is a Flash server, designed to develop connected Adobe Flash client applications.
- It can supports:
  - Networked web applications (like text chat system)
  - Voice
  - Video playback
  - Video recording
  - Video chat
  - Event distribution with publish and subscribe system
  - ...
- Perfect server for rich web real time applications.

OneTeam Media Server: Why Erlang ?

- For scalability and flexibility
- For clustering features
- For the ability to integrate the application with other popular Erlang software:
  - ejabberd
  - Couchdb
  - Yaws
  - Mochiweb
  - RabbitMQ

- All those software are getting mindshare in their area. OMS will both add value and gain from them.

Rich real time web applications in Erlang

- OneTeam Media Server gives the opportunity to build Adobe Flash or Adobe Air connected / real time applications in Erlang
- OneTeam Media Server is an application server. It is used as a building brick for your own applications.

- Example applications:
  - Low latency multiplayer games.
  - Video voicemail (Seesmic-like features).
  - Video conferencing system.
  - Video distribution system.
  - Event distribution system for financial dashboards.
  - Push base news distribution systems.
  - ...

OneTeam Media Server perspective

- OneTeam Media server along with ejabberd can help building the most versatile web application server platform.
- Both components can play a different role but can become a key element in the real time web.
- Can bring new web developers / startup to Erlang.
- All in Erlang !

process one
INSTANT MESSAGES | CREATIVE BUSINESSES

Using OneTeam Media Server

Technical overview

- Several Adobe protocols are at the core of the platform:
  - RTMP: Real Time Messaging Protocol (port 1935)
    - Originally used for Flash persistence.
    - Now used for Flash RPC, streaming, messaging.
    - AMF: Action Message Format. Protocol to define the RPC and message form. This is a binary serialized representation of Flash objects.

- Technically RTMP is the main transport protocol, but actually it is AMF over RTMP.

- All other features are build upon those foundations:
  - Shared objects
  - Publish and subscribe
  - Video streaming

Technical overview

- RTMP exist in various flavour:
  - RMTP: Standard TCP/IP (port 1935).
  - RTMPS: Secured RTMP. This is basically RTMP inside TLS.
  - RMTPT: Tunnelled over HTTP (port 80). Fallback mechanism used to pass firewalls. It can work over HTTPS as well.
    - Yes, it works with video streaming as well

- Currently OMS implements only RTMP
  - This is the bigger part
  - Adding other variations should be straigthforward

Flash client point of view

- The first thing to do to work with OMS on the Flash client is to connect to OMS.
  - This is done with the Flash connection object:
    - flash.net.NetConnection
- Once you have done that you have a bidirectional stream to perform RPC from client to server or from server to client.
- If you want to support media publishing (Voice or Video), you have to use another Flash object:
  - flash.net.NetStream

OMS application point of view

- You need a module to handle the connection from your application.
  - You do that by creating an adaptor for the Flash NetConnection.
  - This is an arbitrary module implementing connect/2 method and link to oms_netconnection object.
- If you want to stream video or voice from webcam / microphone you need to implement specific feature in a netstream module and link it to oms_netstream object
  - Example of function to implement: publish/2,closeStream/2

process one
INSTANT MESSAGES | CREATIVE BUSINESSES

OneTeam Media Server: Example applications

Basic application

- First basic application will only open the RTMP connection between Flash client and module.
  - Client code (Flex)
  - Server code (Erlang)

Basic application: client

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute" initialize="init()">
  <mx:Script>
<![CDATA[
import flash.net.NetConnection;      import flash.net.SharedObject;      import mx.controls.Alert
private var myconn:NetConnection;

public function init():void {
   myconn = new NetConnection();
   myconn.addEventListener("netStatus", onNCStatus);
   myconn.connect("rtmp://127.0.0.1/demo");
}

public function onNCStatus(event:NetStatusEvent):void {
        if(event.info.code == "NetConnection.Connect.Success")
          { Alert.show("Connected to OMS"); }
        else { Alert.show("Failed to connect to OMS. Reason:"+ event.info.code); }
       }
]]>
  </mx:Script>
</mx:Application>
```

Basic application: Server

```
-module(mynetconnection).

-export([connect/2]).

connect(_CallC,_Msg) ->
    io:format("my netconnection received call"),
{object,Obj} = lists:nth(1,_Msg),
case dict:find("objectEncoding",Obj) of
   {ok,{number,Encoding}} ->
       io:format("~nEncoding is :~w",[Encoding]),
       {ok,[{result,[{mixed_array,{0,[{"objectEncoding",{number,Encoding}},{"application",{null,0}},{"level",
            {string,"status"}},{"description",{string,"Connection succeeded"}},
            {"code",{string,"NetConnection.Connect.Success"}}]}}]}]};
   error  ->
        {ok,[{result,[{object,dict:from_list([{"level",{string,"status"}},
              {"code",{string,"NetConnection.Connect.Failed"}},
              {"description",{string,"Connection failed"}}])}]}]}
end.
```

This is raw data structure, but we provide helpers / wrappers

Application XML configuration

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<omsapp>
<display-name>OMS Demo Application</display-name>
<short-name>demo</short-name>
<apppath>server/ebin</apppath>
<adapter obj="oms_netconnection" adapter="mynetconnection" />
</omsapp>
```

You can have several adapter objects

Chat application

- The goal of the chat test application is to show you can do push from the server side by calling functions that are running on the client.

Chat application: Interesting client code

```
public function post():void {
  resp = new Responder(glamresult,glamerror);
  myconn.call("post",resp,nick.text,saisie.text);
   input.text = "";
}

public function newline(s:String):void {
    chatarea.text += s + "\n";
}
```

Chat application: Interesting server code

```
post(CallC, Msg) ->
        {string,From} = lists:nth(2,Msg),
        {string,Text} = lists:nth(3,Msg),

 gen_server:cast(CallC#callcontext.rtpid,
                        {clients_call,"newline",[{string,binary_to_list(Text)}],0}),

 Arraydata = lists:map( fun(X) ->  {string,X} end,[]),
 AnsAMF3 = {array,Arraydata},
{ok,[{result,[AnsAMF3]}]}.
```

clients_call: call on all clients
client_call: use to call on only one client

Chat application: Demo

Video player / recorder: Interesting server code

```
getlistofavmedia(_CallC,_Msg) ->
    io:format("~ngetlistofavmediacalled"),
    Filelist = filelib:wildcard("../medias/*.flv"),
    Arraydata = lists:map( fun(X) ->  {string,X} end,Filelist),
AnsAMF3 = {array,Arraydata},
{ok,[{result,[AnsAMF3]}]}.
```

Video player / recorder: Interesting client code

```
public function get_list_of_av_medias():void {
    var resp:Responder;
    resp = new Responder(glamresult,glamerror);
    myconn.call("getlistofavmedia",resp);
}
 private function glamresult(result:Array):void {
          var length:int = result.length;
          var newtab:Array;
          newtab = new Array();
          for (var i:int=0;i<length;i++) {
                    var temp:String;
                    temp = result[i];
                     var ta:Array;
                     ta = temp.split("/");
                     var med:String = ta[ta.length - 1];
                     newtab[i] = {Name:med};
          }
         medlist.dataProvider = newtab;
}
```

Video player / recorder: Interesting client code

```
public var ns1:NetStream;
public function startrecording(e:MouseEvent):void {
      bustop.visible = true;
      burec.visible = false;
      var temp1:String = medrecname.text.replace("/","");
      var temp2:String = temp1.replace("..","");
      if (ns1 != null) {
          ns1.close();
          ns1.attachCamera(camera);
          ns1.publish(temp2,"RECORD");
       }
}
```

Video player: Demo

Video chat: Demo

process one
INSTANT MESSAGES | CREATIVE BUSINESSES

OneTeam Media Server future

What's next ?

- Open Source release
  - When ?
    - 2009, May 15th (in alpha version)
    - Release from ProcessOne web site
- Planned features
  - Remote shared objects
  - RTMPT: Adobe connected protocol over HTTP
    - This is needed in some context to work over firewalls.
  - Adobe Livecycle publish and subscribe like features
    - Ability to support push to client over RTMP / RTMPT
    - BlazeDS (Java exist in open source but does not support more efficient protocol RTMP).
  - More example applications

process one
INSTANT MESSAGES | CREATIVE BUSINESSES

OneTeam Media Server

http://www.process-one.net/en/oms/