Building a Scalable Interactive Video Solution using Peer Assisted Networking in Flash Builder

MAX 2009 Hands-On Lab

Lab files: `Desktop/exercises/`

Instructors: Michael Thornburgh, Kevin Towes

Twitter: #adobemax104 (Monday), #adobemax82 (Tuesday), #adobemax375 (Wednesday)

EXERCISE 1: Make an RTMFP NetConnection to a local RTMFP server

This exercise will step you through the process of establishing a connection to the RTMFP rendezvous server (Stratus) using the NetConnection Class. At the end of this exercise, you will request a connection to an RTMFP server and receive status events.

Set up the Development Environment

1. Launch Flash Builder from the Dock
2. Open the "viewer" application
3. Open `"src/viewer.mxml"` in the editor

Add the Connection Logic

1. Locate the variables section at the top of the file
2. Add a string constant called `SERVER` to list the server name `rtmfp://stratus.local/`

```
private const SERVER:String = "rtmfp://stratus.local/";
```

3. Locate the empty `DoConnect()` function. Add the highlighted lines to the function to output a status message, create a new `NetConnection`, attach the pre-built event listener function, and connect to the server.

```
private function DoConnect():void
{
    StatusMessage("Connecting to \"" + SERVER + "\" ...\n");
    netConnection = new NetConnection();
    netConnection.addEventListener(NetStatusEvent.NET_STATUS, NetStatusHandler);
    netConnection.connect(SERVER);
}
```

Test your Application

1. Save `viewer.mxml`. Flash Builder will build your application
2. In Flash Builder, click the Play button. This will launch the application in a web browser
3. In the application, click the Connect button
4. Notice the status message. It should output "Connecting to..." and "Disconnected".

You have successfully requested a connection to the RTMFP server, but the connection was not accepted.

EXERCISE 2: Using a developer key to successfully connect

This exercise will step you through using a developer key to connect to Adobe Stratus, a free RTMFP service hosted by Adobe. At the end of this exercise, you will successfully establish a connection to the local Stratus server.

Set up the Development Environment

Open the source file for your application from the previous exercise. If you are still testing your application, close your browser. If you did not complete the previous exercise, you can replace your project's `viewer.mxml` with `Desktop/exercises/02-add-devkey/viewer.mxml` .

Set the Developer Key

1. In Flash Builder, go to the variables section at the top of viewer.mxml
2. Add a new string constant called `DEVKEY` below the `SERVER` string constant, set to `"MAX2009-04893b3a8a06"`

```
private const DEVKEY:String = "MAX2009-04893b3a8a06";
```

Update the Connection Logic

Locate the `DoConnect()` function. Change the highlighted line to add the developer key to the connect URI so that the function looks like

```
private function DoConnect():void
{
    StatusMessage("Connecting to \"" + SERVER + "\" ...\n");
    netConnection = new NetConnection();
    netConnection.addEventListener(NetStatusEvent.NET_STATUS, NetStatusHandler);
    netConnection.connect(SERVER + DEVKEY);
}
```

Test your Application

1. Save `viewer.mxml`. Flash Builder will build your application
2. In Flash Builder, click the Play button. This will launch the application in a web browser
3. In the application, click the Connect button
4. Notice the status message. It should output "Connecting to..." and "Connected"
   (If you see "Disconnected") : close your application, check the `DEVKEY` value, and retry

You have successfully requested a connection to the RTMFP server, and the connection was accepted.

EXERCISE 3: Use the GroupSpecifier class to specify a Group

This exercise will show you how to use the `GroupSpecifier` class to construct a *Group Specification* string, used to join Flash Peer-to-Peer groups. At the end of this exercise, you will have constructed a Group Specification string encoding the name and capabilities of a group to be used in future exercises, and output it to the status log.

Set up the Development Environment

Open the source file for your application from the previous exercise. If you are still testing your application, close your browser. If you did not complete the previous exercise, you can replace your project's `viewer.mxml` with `Desktop/exercises/03-add-groupspec/viewer.mxml` .

Make the Group Specification

1. Locate the OnConnect() function
2. Add a definition for a new variable for the GroupSpecifier

```
var groupSpecifier:GroupSpecifier;
```

3. Make a new instance of GroupSpecifier, naming it `"max2009lab/"` concatenated with the value of the `groupName` input field

```
groupSpecifier = new GroupSpecifier("max2009lab/" + groupName.text);
```

4. Enable multicast

```
groupSpecifier.multicastEnabled = true;
```

5. Enable the server channel so that Stratus can bootstrap your application into the peer-to-peer group

```
groupSpecifier.serverChannelEnabled = true;
```

6. Output the Group Specification string for this group

```
StatusMessage("Join \"" + groupSpecifier.groupspecWithAuthorizations() + "\"\n");
```

7. Check the new OnConnect() function

```
private function OnConnect():void
{
    var groupSpecifier:GroupSpecifier;

    StatusMessage("Connected\n");
    connected = true;

    groupSpecifier = new GroupSpecifier("max2009lab/" + groupName.text);
    groupSpecifier.multicastEnabled = true;
    groupSpecifier.serverChannelEnabled = true;

    StatusMessage("Join \"" + groupSpecifier.groupspecWithAuthorizations() + "\"\n");
}
```

Test your Application

1. Save `viewer.mxml`. Flash Builder will build your application
2. In Flash Builder, click the Play button. This will launch the application in a web browser
3. In the application, click the Connect button
4. Notice the status message. It should output "Connecting to...", "Connected", and then `Join "G:0101010c130e6d6178323030396c61622f64656661756c7400".`
   (If you do not see the "Join" message): close your application and check that the `OnConnect()` function matches the above.

EXERCISE 4: Construct a NetGroup and Group NetStream

This exercise will show you how to make a `NetGroup` and a P2P Group `NetStream` using the Group Specification from the previous exercise.

Set up the Development Environment

Open the source file for your application from the previous exercise. If you are still testing your application, close your browser. If you did not complete the previous exercise, you can replace your project's `viewer.mxml` with `Desktop/exercises/04-add-join/viewer.mxml` .

Make the P2P Group NetStream and NetGroup Objects

1. Locate the `OnConnect()` function (same as previous exercise)
2. Add the highlighted lines to make a new `NetStream` for the previously specified group, a new `NetGroup` for the group, and to add `NetStatusHandler()` to each of them to handle their events (note: global variables `netStream` and `netGroup` have already been defined at the top of the code)

```
private function OnConnect():void
{
    var groupSpecifier:GroupSpecifier;

    StatusMessage("Connected\n");
    connected = true;

    groupSpecifier = new GroupSpecifier("max2009lab/" + groupName.text);
    groupSpecifier.multicastEnabled = true;
    groupSpecifier.serverChannelEnabled = true;

    netStream = new NetStream(netConnection, groupSpecifier.groupspecWithAuthorizations());
    netStream.addEventListener(NetStatusEvent.NET_STATUS, NetStatusHandler);

    netGroup = new NetGroup(netConnection, groupSpecifier.groupspecWithAuthorizations());
    netGroup.addEventListener(NetStatusEvent.NET_STATUS, NetStatusHandler);

    StatusMessage("Join \"" + groupSpecifier.groupspecWithAuthorizations() + "\"\n");
}
```

Test your Application

1. Save `viewer.mxml`. Flash Builder will build your application
2. In Flash Builder, click the Play button to launch the application in a web browser
3. In the application, click the Connect button
4. The output in the status log should be the same as in the previous exercise. However, the Peer Assisted Networking dialog should appear
5. Click Allow in the dialog box
6. Notice that no messages are displayed in the status log for this case yet
7. Close the application and re-launch it from Flash Builder
8. In the application, click the Connect button
9. Click the Deny button in the Peer Assisted Networking dialog
10. Notice that "Disconnected" appears in the status log

EXERCISE 5: Subscribe to a video broadcast

In this exercise, you will use your RTMFP connection to play a live video broadcast application-layer (peer-to-peer) multicast.

Set up the Development Environment

Open the source file for your application from the previous exercise. If you are still testing your application, close your browser. If you did not complete the previous exercise, you can replace your project's `viewer.mxml` with `Desktop/exercises/05-add-play/viewer.mxml` .

Handle the Peer Assisted Networking "Allow" Case

1. Locate the `NetStatusHandler()` function
2. In the `switch` statement, locate the case for `NetStream.Connect.Success`
3. Insert a call to `OnNetStreamConnect()` in this case handler before the `break` statement
4. Locate the case for `NetGroup.Connect.Success`
5. Insert a call to `OnNetGroupConnect()` in this case handler before the break statement

```
private function NetStatusHandler(e:NetStatusEvent):void
{
    switch(e.info.code)
    {

    ...

    case "NetStream.Connect.Success": // e.info.stream
        OnNetStreamConnect();
        break;

    ...

    case "NetGroup.Connect.Success": // e.info.group
        OnNetGroupConnect();
        break;

    ...
    }
}
```

Subscribe to the Multicast Stream

1. Locate the `OnNetStreamConnect()` function
2. Play the stream called `"stream"` and set the playback buffer to one second by inserting the highlighted lines

```
private function OnNetStreamConnect():void
{
    netStream.client = this;

    if(null != video)
    {
        videoDisplay.removeChild(video);
        video = null;
    }

    video = new Video();
    video.smoothing = true;
    video.attachNetStream(netStream);
    videoDisplay.addChild(video);

    netStream.play("stream");
    netStream.bufferTime = 1.0;
}
```

Enable the User Interface

1. Locate the `OnNetGroupConnect()` function
2. Enable user interface components by setting the bound variable `joinedGroup` to `true`

```
private function OnNetGroupConnect():void
{
    joinedGroup = true;
}
```

Test your Application

1. Save `viewer.mxml`. Flash Builder will build your application
2. In Flash Builder, click the Play button to launch the application in a web browser
3. In the application, click the Connect button
4. Click Allow in the dialog box
5. After a few seconds, video should appear in the window. If it doesn't, review your code.

EXERCISE 6: Using Posting to build a Chat

In this exercise, you will add a simple text chat to the video viewer using NetGroup *Posting*.

Set up the Development Environment

Open the source file for your application from the previous exercise. If you are still testing your application, close your browser. If you did not complete the previous exercise, you can replace your project's `viewer.mxml` with `Desktop/exercises/06-add-chat/viewer.mxml` .

Enable Posting in the Group

Add the highlighted line to the `OnConnect()` function to enable posting in the Group:

```
private function OnConnect():void
{
    var groupSpecifier:GroupSpecifier;

    StatusMessage("Connected\n");
    connected = true;

    groupSpecifier = new GroupSpecifier("max2009lab/" + groupName.text);
    groupSpecifier.multicastEnabled = true;
    groupSpecifier.postingEnabled = true;
    groupSpecifier.serverChannelEnabled = true;

    netStream = new NetStream(netConnection, groupSpecifier.groupspecWithAuthorizations());
    netStream.addEventListener(NetStatusEvent.NET_STATUS, NetStatusHandler);

    netGroup = new NetGroup(netConnection, groupSpecifier.groupspecWithAuthorizations());
    netGroup.addEventListener(NetStatusEvent.NET_STATUS, NetStatusHandler);

    StatusMessage("Join \"" + groupSpecifier.groupspecWithAuthorizations() + "\"\n");
}
```

Post a Message to the Group

Add the highlighted lines to the `DoPost()` function to post a message to the Group. The posted object contains your username as the `user` property of the object, the message itself as the `text` property of the object, and a sequence number and sender ID as the `sequence` and `sender` properties, respectively, to ensure that each posted object is unique (as required by the posting system).

```
private function DoPost():void
{
    var message:Object = new Object;

    message.user = userName.text;
    message.text = chatText.text;
    message.sequence = sequenceNumber++;
    message.sender = netConnection.nearID;

    netGroup.post(message);

    StatusMessage("==> " + chatText.text + "\n");

    chatText.callLater(ClearChatText);
}
```

Display Incoming Postings

1. Locate the `NetStatusHandler()` function
2. In the `switch` statement, locate the case for `NetGroup.Posting.Notify`
3. Insert a call to `OnPosting(e.info.message)` in this case handler before the `break` statement

```
private function NetStatusHandler(e:NetStatusEvent):void
{
    switch(e.info.code)
    {

    ...

    case "NetGroup.Posting.Notify": // e.info.message, e.info.messageID
        OnPosting(e.info.message);
        break;

    ...

    }
}
```

4. Locate the `OnPosting()` function and see that it uses the user and text properties of the received object to output a line to the status log.

Test your Application

1. Save `viewer.mxml`. Flash Builder will build your application
2. In Flash Builder, click the Play button to launch the application in a web browser
3. In the application, click the Connect button
4. Click Allow in the dialog box
5. After a few seconds, video should appear in the window
6. Change your username in the lower left box to your own name
7. Type a message in the chat input field in the lower right, and hit Return. The outgoing message should appear in the status log
8. Observe that messages from other participants appear in the status log. Chat amongst yourselves.

ADDITIONAL MATERIAL: Changes to Publish a Multicast Stream

This section lists the differences between the final viewer application that you developed in the previous exercises and a pre-built publishing application. The "publisher" application is already in Flash Builder. Its main source file is `src/publisher.mxml`, and its functions should already look like the below.

```
    private function OnApplicationComplete():void
    {
        userName.text = "user " + int(Math.random() * 65536);

        groupName.text = "channel" + (int(Math.random() * 899) + 101);

        resizeTimer = new Timer(2000.0);
        resizeTimer.addEventListener(TimerEvent.TIMER, DoResizeVideo);
        resizeTimer.start();
    }

...

    private function OnNetStreamConnect():void
    {
        netStream.client = this;

        var mic:Microphone = Microphone.getMicrophone();
        if(mic)
        {
            mic.codec = SoundCodec.SPEEX;
            mic.setSilenceLevel(0);

            netStream.attachAudio(mic);

            StatusMessage("got microphone\n");
        }

        var camera:Camera = Camera.getCamera();
        if(camera)
        {
            camera.setMode(320, 240, 10);
            camera.setQuality(30000, 0);
            camera.setKeyFrameInterval(15);

            videoDisplay.attachCamera(camera);
            videoDisplay.maintainAspectRatio = true;

            netStream.attachCamera(camera);

            StatusMessage("got camera\n");
        }

        netStream.publish("stream");
    }

...

    private function DoDisconnect():void
    {
        if(netConnection)
            netConnection.close();
        videoDisplay.attachCamera(null);
    }

...

    <mx:VideoDisplay id="videoDisplay" width="320" height="240" resize="DoResizeVideo()"/>
    <mx:TextArea id="statusLog" width="100%" height="100%"/>

...
```