

# Media Service using Red5 and Adobe AIR

by Olav Nistad

---

## I. Abstract

As the resources of client devices accessing the web as well as the bandwidth of Internet connections have improved greatly in the later years, richer and more resource-demanding services can be provided to the users. Heavy data such as multimedia are now not only limited to expensive and powerful computers, but can now be processed by average equipped and affordable computers, laptops and even smaller devices such as iPhones. You can now rent movies online, and even watch them online. Such streaming services often require specialized streaming servers. One of the most known enterprise media servers is the Flash Media Server (FMS) by Adobe Systems. This server works together with Flash Player to create media driven multiuser RIA's (Rich Internet Applications). Osflash, that is a community for open source flash developers, has developed an open source version of FMS written in Java which is called Red5. I have chosen to use the latter in my project. On the client side I have developed a Flash application in Adobe Flex that is running through Adobe AIR. Adobe AIR is the latest interesting development by Adobe, and removes several of the limitations that other web applications suffer from. In this paper I will start with an insight in how media can be delivered on the Internet today. Then I will give an explanation of both the server implementation and the client implementation of my project.

## II. Streaming vs progressive download

When delivering media over the Internet you have two choices; streaming and progressive download. [1], [2] It is common by many to use the term «streaming» about any technique of

watching media online, but in fact, most often you are not streaming, but using progressive downloading. In both techniques the media content is kept external to the client application (swf-file). The benefits with this approach is mainly that it is easy to make changes to the media files without needing to republish the swf-file. Keeping the media content separate from the player also enables the developer to keep the size of the swf-file as small as possible to enforce fast page loads.

### Progressive download

Progressive download is a popular method of delivering media over the Internet today. Both YouTube and Break.com uses this method. Progressive download allows developers to load external video files into a Flash or Flex interface and play them back during runtime. When the video is played, the video file first begins to download to the client's hard drive (buffer). When enough of the data is downloaded the playback begins. If the network bandwidth is poor, you can pause the playback while the rest of the file is being downloaded in the background. The file is served from a standard web server through an HTTP request, just like a normal web page or any other downloadable document.

In comparison to streaming video, one consistent benefit to progressive download is that you don't need a streaming server to deliver the video, since progressive download video can be served from any normal web server.

### Streaming

Like with progressive download the streaming method also keeps the video content external to the client application. Developers can use ActionScript code to load the media into the player and play them back at runtime. In fact, the ActionScript code needed for streaming is almost the same as the code for progressive download. The main difference lies in the server. A streaming method requires a special streaming server in contrast to a regular HTTP web server as is the case for progressive download.

When watching video from a streaming server,

each client opens a persistent connection to the server and the server streams the video bits to the client. Those bits are displayed and then immediately discarded.

This tight connection between the media server and the client, and the server's ability to precisely control and deliver any portion of a stream as requested, enables the developer to take advantage of some advanced capabilities. Some of these are:

- determining the client bandwidth and serving a stream with an appropriate bitrate
- measuring and tracking the stream's quality of delivery and switching to a lower, or higher bitrate stream if needed – if network congestion increases, for example.

- Automatically generating thumbnails or play short previews of your video without having to create separate image or video clips.

- Automatically creating «chapters» so you can jump forward in longer videofiles, without having to cut up the video file into smaller pieces.

While progressive downloading would be the easiest choice if you have a lot of small video clips (<10 min), where the video content is not subject to copyright restrictions, the streaming method has a lot of advantages in the other case. Some of these are:

#### **Advanced video control**

A streaming server offers bandwidth detection, QoS monitoring, server-side playlists and more.

#### **Efficient use of network resources**

Only the bits that the client can actually view are transferred

#### **Secure, protected multimedia delivery**

Since the media data is not saved in the client's cache when streamed, the users cannot save a copy of the media.

#### **Tracking, reporting and logging capabilities**

With streaming you can log statistics of how the clients «behave» while watching the video. Such as when he or she navigates forward or backward, how long they have watched the file, how many times the user has watched the file and other things.

#### **Seek and navigation**

Viewers can seek to any point in the video file and start to stream the content from that point immediately. This is in contrast to progressive download where you have to download all the content up to the point you are seeking to before you can watch.

#### **Interactivity**

The precise control found in streaming enables developers to create extensive interaction in their video applications. One example is the ability to switch camera angles, have one video spawn another video, or seamlessly switch to alternate endings,

#### **Live video and multiuser capabilities**

Streaming provides the ability to deliver live video and audio from the client's webcams. This enables the possibility of video conferencing.

### **III. Red5 Standalone server**

#### **Spring Framework**

Red5 is a standalone server that includes a Jetty Embeddable Http Server- and Servlet container, managed by the Spring Framework [3]. The Spring Framework is a Java Framework where configuration of metadata is in the form of either XML bean definitions, Java properties file or by calling the configurable items of the Spring API from within your Java object [5]. From this configuration you can control the behaviour of various Java applications, objects or frameworks (eg Jetty, or Hibernate).

This allows you to use a broad range of Java technologies from within your Java object without having to face the challenge of coding to interface to all these Java objects yourself while still ensuring you adhere to the common Java Design Patterns.

The XML configuration metadata is used by Spring to create a fully configured system or application. This configuration metadata tells the Spring container how to instantiate, configure, and assemble the objects in your application.

#### **Communication Protocol**

The rules of communication between the server and the client is set by the Real Time Messaging Protocol (RTMP) [6],[7]. This protocol was developed by Adobe Systems and is used for delivery of real time objects. The protocol is a container for data packets that are either AMF (see next section) or raw audio/video objects such as Flash Video (FLV). RTMP comes in three variations:

- RTMP : Default communication protocol using port 1935
- RTMPT: Tunnel through HTTP, default port of 80
- RTMPS: Tunnel through HTTPS, default port of 443

As default the client tries to communicate with the server using the default RTMP protocol. Sometimes, however, the client is located behind a firewall which prevents communication to non-standard ports. Thus, if the first connection-attempt fails the client tunnels its RTMP packets through HTTP running at port 80. When tunneling the client actually sends POST requests to the server. Because of the non-persistent nature of HTTP, RTMPT requires the clients to poll for updates periodically in order to get notified about events that are generated by the server or other clients. RTMP are preferred over RTMPT as the latter approach can lead to a downgrade of performance. If security is an issue, RTMPS can be used so that SSL encryption is enforced on the data stream.

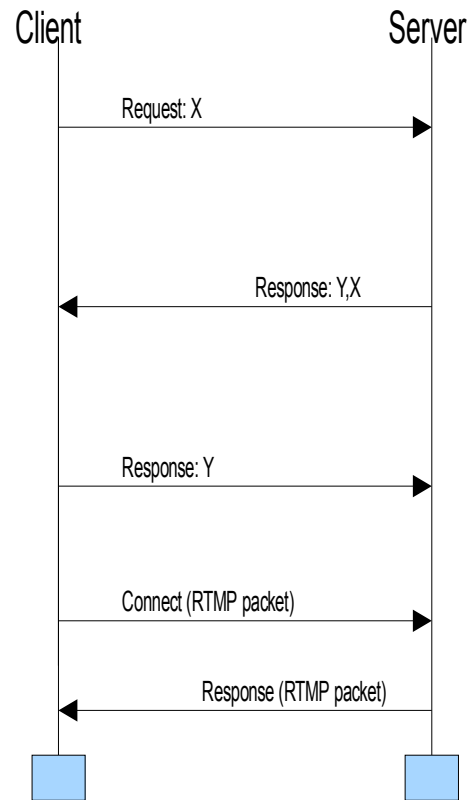
Raw RTMP uses a binary TCP connection which allows a persistent connection to be set up. A single connection is capable of multiplexing many net streams using different channels. Within these channels packets are split up into fixed size body chunks.

Since RTMP runs on top of TCP a handshaking process must be initiated before any streaming can find place. The handshaking procedure looks like follows:

When the client first wants to talk to the server it Each application instance has its own unique

sends a request message where it includes a single byte (0x03) followed by a 1536 bytes with a random content (X in figure). The server the responds with a single byte (0x03) followed by 1536 bytes of random content (Y), in addition to the clients 1536 bytes chunk (X). The client the responds by sending back the server generated

Figure 1: Handshake process



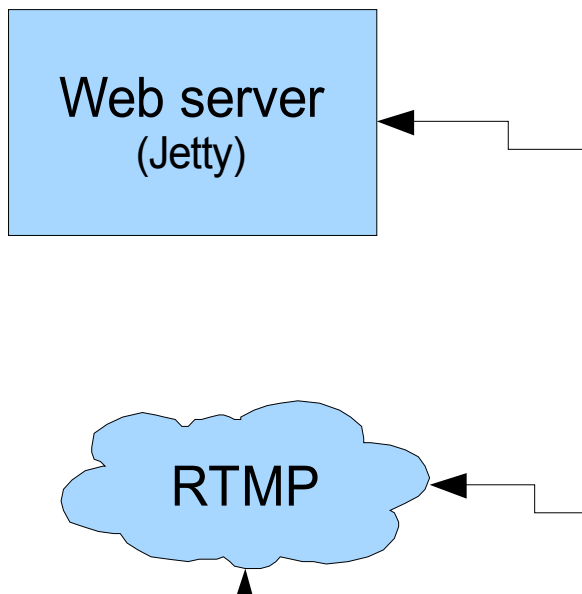
bytes (Y). The handshake process is now done, and the client and server can start to talk to each other using RTMP. It is first now, in the latter stage, that the client can access the services offered by the media server. (See figure 2) Once the client is connected to the media server it can communicate with both the server and with other clients. Clients connects to instances of applications: for example a chat application has many rooms. Each room is an instance of the same chat application. Multiple instances of an application can be running simultaneously.

name and provides unique resources to its

connected clients.

Due to the streaming nature of RTMP, the performance will be affected by the connection established between the server and the client. Since RTMP uses TCP/IP for packet transmission, it will have information on which packets that are sent successfully, and which have not. The packets sent are broken down in three different types. These are data, audio and video. During the connection the server may decide to drop some packets, meaning that it drops queued packets that are ment for the client to receive. The reason for this is normally that the server has to accomodate for a slow connection. When it decides to drop it first starts with the video packets, then audio at at last data packets. The reason for this order is based on the critical information needed for the client application to work properly. A human watching a video can more easily accept some framelosses during a video playback, that a choppy sound. Data is most critical since it is often needed to perform necessary logic or operations.

Figure 2: Full server/client architecture. I have not included the use of a database in my project, but I included the entity in the drawing to show that it can be easily implemented.



## IV. Client

For the client part of my project I have created a media player in Adobe Flex [8] running through Adobe AIR [10]. Below is a brief explanation of the most important concepts involved.

### Adobe Flex

HyperText Markup Language is the predominant markup language for web pages. This language undeniably provides heavy restrictions for your applications and most developers therefore needs to add some JavaScript and DHTML to make the application more dynamic and intuitive. Adobe Flex takes you to a new level of web application development with the concept of Rich Internet Applications (RIAs). A lot of web pages are developed natively in Flash itself and provides a much more robust experience for the user where productivity is bolstered by ease of use, streamlined workflow and real time interactivity that is impossible to accomplish in HTML itself. To develop flash applications from the bottom for developers that aren't core flash developers, can be a time consuming affair. The Flash tools are created for designers so working with a timeline can be strange and unintuitive for others. Adobe Flex removes this barrier to entry by providing a programmatic way for deloping thes RIAs.

### Rich Internet Applications

HTML applications are completely state-less. The application server keeps track of the state by using cookies- and session variables, but all the logic is done on the server side. So when a user are looking at a web page, nothing new will happen untl the user performs some action, and commits that action to the server. The server will then process the incoming data and generate a new HTML page that the browser loads back in. JSP, PHP, ASP and Coldfusion are common languages used to create such dynamic web pages.

In Flex, your application does not need to send data to the server everytime the user makes a

change. Instead of the logic lying purely on the server side, you can develop an application where the location of the logic is more divided between the two (client and server). For small applications with few users this won't make a big difference, but if your application is serving thousands of users a solution like this will undoubtedly save your web server, application server, database and network for a lot of unnecessary load.

Flex does not replace server side logic, it's more of a companion to it. While your server logic provides the middle/business tier, your client logic takes care of the presentation tier. It lets you focus on the client workflow, and not waste time building HTML tables and endless cosmetic logic. Flash is not unique in this context, that a lot of the logic resides at the client side. But with a penetration above 96% the Flash Player helps you provide a rich real-time experience on Unix, Windows and MAC. Alternatively any richness you try to accomplish in IE using DCOM, ActiveX and DHTML locks you down to particular platforms.

There is a close connection between the functionalities offered by Flex and Ajax. While the first is a RIA in that you are building application that are desktop like but use the Internet for deployment ease and connectivity to services, the latter can be called a Rich Web Application (RWA) in that it increases functionality and usage efficiency of web pages. HTML provides means to describe the structure of text-based information in a document, and it vastly improved how content can be deployed. It is, however, severely limited on UI functionality since it is a stateless mechanism. A web application gives a much more dynamic experience where the choices you make generate new pages on the fly. Although there is an overlap in how Flex and AJAX can be used to solve a problem there are still some differences. If you want to build a photo editor, for example. HTML is optimized for document distribution, so although it could be done in AJAX, it would be too time consuming. Flex on the other hand has desktop like capabilities, so making such a graphically oriented application would be pretty

straight forward. Another example is an application where you want to use your webcam, for web conferencing for example. This is very easy in Flex, and impossible in HTML/AJAX. A third example is an application who provides you a live update of the soccer results. With an AJAX solution you would have to periodically perform polling to check for updates, even if there haven't been any. In a Flex solution a backend server would simply push the updates to the client as they occur.

Another drawback with AJAX is that because JavaScript differs from browser to browser and OS to OS, a massive amount of QA needs to be done to ensure that it works across all platforms. There is however, a big effort to provide tools and API's to abstract that complexity, and streamline the development of AJAX based application. Most notably the Open AJAX Initiative, which is supported by such as IBM, BEA, Borland, Google, Mozilla, Oracle, RedHat and Yahoo.

### **Developing in Flex**

You use two languages to write Flex applications: MXML and ActionScript. MXML is an XML markup language that you use to lay out user interface components. You also use MXML to declaratively define nonvisual aspects of an application, such as access to data sources on the server and data bindings between user interface components and data sources on the server. MXML will seem very familiar if you have worked with HTML as the syntax is based on tags. However, MXML is more structured than HTML, and it provides a much richer tag set. You can also extend MXML with custom components that you reference as MXML tags.

You build your main files in .mxml, and you can build custom components, objects and logic in the ActionScript (.as). The latter language will be very easy to learn for developers who are familiar with Java, and other OO languages.

The MXML and ActionScript code are compiled into SWF files and rendered by Adobe Flash Player (in web browser), or Adobe AIR (on desktop).

## Limitations

Since Flex applications are Flash applications, they are limited by what the Flash Player can do.

The Flash Player is not a web browser. So creating rich documentation is challenging. This is what HTML was originally developed for.

Flash popup windows are limited to being displayed inside the dimensions of the player that created it. So unlike a regular popup window in your browser, that you can move around everywhere you want, a flash popup cannot move outside the area of the player.

Drag and Drop from Desktop to Player is not supported. (Browser based Flash Player only)

The Flash SWF format is a published specification, so it is possible for other vendors to make tools that generate Flash. Because Flex applications are thin clients, and not web pages, they are not suitable for where you want to use a web page. For example, there is an initial load and initialization process that occurs when running a Flex application, so the user would have to wait for some time (depends on the application how long) as the application loads.

## Adobe AIR

Adobe AIR stands for Adobe Integrated Runtime [10] and is one of the latest interesting development from Adobe. AIR applications run on the desktop and are half way between a web based application (which runs in a browser) and a full blown desktop application such as Word or Photoshop. Developers can create AIR applications using HTML, CSS, Javascript, Flash/Flex or PDF, and AIR acts as a wrapper that allows them to run on the desktop and go beyond the limitations they would normally have if they were running in a web browser. The latter fact is what is really separating AIR applications from regular web applications; because they run as desktop applications developers can implement

features such as full drag and drop between applications, save files to the local hard drive or network, and store data locally in a database. There is not really that much new in AIR from a technology perspective. Its strength is that web developers can use existing web technologies to create applications that run on the desktop. AIR can work in tandem with your existing web sites and web applications and can be built by web developers using the skills they already have. AIR also matters because it provides companies and organizations the possibility to have a presence not only on a web page, but on the desktop as well. There is also the look-and-feel aspect of it, meaning that often it looks better, and is more practical to access a web service from your desktop, instead of having to always open your web browser.

## V. Interaction between the server and client

### Remote Procedure Call

A Remote Procedure Call (RPC) is a technology that allows a program to invoke methods/functions/subroutines on another program running on another computer. It is a very common paradigm in a any client-server model. The invoke is initiated just as a normal procedure call from the client where the call is sent as a message together with parameters to the remote server. The client is then blocked while it waits for the response.

There exists different RPC protocols and the main difference lies in the format of the messages sent. Macromedia (now Adobe) has developed a message format called AMF (Action Message Format), and the remote interactivity between Flex/Flash and Red5 is also based on this format. AMF is a particular message structure that needs to be understood by both the client and the server. Once the initial handshake is done, all messages sent uses this particular structure.

The first conversation using the AMF structure between the Flash Player and Red5 is the

ActionScript call `NetConnection.connect()`  
Below is a code example written in ActionScript3  
to connect to the Red5 media server.

```
var myvideo:Video = new Video();
var nc:NetConnection= new NetConnection();
var connected:Boolean = nc.connect(«rtmp://[mediaserver-address]/myapp»);
if(connected){
    var ns:NetStream = new NetStream(nc);
    myvideo.attachVideo(ns);
    ns.play(«flvFile»);
}
```

The `NetConnection.connect()` method creates a bidirectional connection between the Flex application and the Red5 server.

The `NetStream` class opens a one-way streaming connection between the two. While the `NetConnection` object works like a pipe between the client and the server, the `NetStream` object is used to send streams through the pipe.

To invoke functions on the server from the client, one can use the `NetConnection` object together with the function call() like this:

`NetConnection.call('funcname', Respond-object,...arguments).`

On the server side one can use the class `IConnection` to invoke function residing at the client: `IConnection.invoke('funcname', ..arguments).`

## VI. Conclusion

As the bandwidth and CPU capacity only will continue to improve, we will most likely witness more and more multimedia services being deployed on the web. To watch video online is now not only limited to short movie clips, but it is now possible to stream full length videos in high definition quality. This requires specialized streaming servers that can preserve copyright restrictions and so that QoS (Quality of Service) can be employed. Red5 is such a server, and it is a really exciting tool to «play around» with. Since this is a non commercial project that is relatively new, it does not fully provide you with all the features that FMS does. The team behind Red5 are, however, continuously developing it and making it better. Through their mailing list they

are creating a largely growing community that aids and helps with the development, it will therefore be exciting to follow the progress and see what happens.

The new development from Adobe, AIR, is also a very interesting and an exciting tool. It provides a big jump in the development of RIAs because of the way it provides the users a different way of accessing and interacting with web services. By being a desktop application that accesses web services, it can interact more with the local operative system and offer new features that earlier was only possible for regular desktop applications. AIR is still pretty young of age, but the interest around it has become very large and by searching the web you can find a lot of great and impressive applications made with AIR.

## References

- [1] Microsoft: «Web Server vs Streaming server»,  
[<http://www.microsoft.com/windows/windowsmedia/compare/webservvstreamserv.aspx>]
- [2] Larry Bouthillier: “Sreaming vs Downloading Video: Understanding the Differences”, Streamingmedia.com, July 22 2003,  
[<http://www.streamingmedia.com/article.asp?id=8456&page=1>].
- [3] Red5 Open Source Flash Server  
[<http://osflash.org/red5>]
- [4] OSFlash.org  
[<http://osflash.org/>”]
- [5] Java Spring Framework  
[<http://www.springframework.com/documentation>]
- [6] OSFlash.org: «RTMP Protocol [DRAFT]»  
[<http://osflash.org/documentation/rtmp>]
- [7] Overview of streaming with with Flash Media Server 3”, Adobe Systems Inc, February 25 2008:  
[[http://www.adobe.com/devnet/flashmediaser/ver/articles/overview\\_streaming\\_fms3.html](http://www.adobe.com/devnet/flashmediaser/ver/articles/overview_streaming_fms3.html)]
- [8] Adobe Systems Inc, “Flex overview”:  
[<http://www.adobe.com/products/flex/overview/>]
- [9] Adobe Systems Inc, “Overview of the ActionScript 3.0 Language Reference”:  
[<http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/>]
- [10] Adobe Systems, «Adobe Air Developing Center»:  
[<http://www.adobe.com/devnet/air/>]



