

“AMF XML LoadVars RTMP HTTP”

-Flashklientens kommunikation i ett virtuellt community eller multiplayer online spel.

Sammanfattning

Denna uppsats är en uppgift som är gjord i kursen MEC703 – Virtuella miljöer.

Syftet med uppsatsen är att jag djupgående fördjupar mig i vad för olika alternativ det finns för flera flashapplikationer att kunna kommunicera med varandra. Detta för att göra det möjligt att bygga upp en virtuell miljö, virtuellt samhälle, en chatt eller ett multiplayer online spel.

I uppsatsen kommer jag att presentera de olika vägarna som går att ta för att bygga upp en flera användarbaserad flash applikation. Jag kommer inte att gå igenom de vanliga funktionerna i flash eller beskriva programmet genomgående.

I uppsatsen väljer jag skriva Flash istället för Macromedia Flash.

Innehållförteckning

1. Inledning	4
1.1. Syfte.....	4
1.2. Målgrupp	4
2. Olika kommunikationssätt.....	5
2.1. ”Turn based” kommunikation	5
2.2. Socketanslutning.....	7
2.2.1. Flash Communication Server	7
2.2.2. Unity 2, Moocks server	9
2.2.3. Flash now.....	9
2.2.4. Jabber.....	10
2.2.5. Skriv en egen server	10
2.3. XMLSocket i flash	12
2.4. Web services.....	13
3. XML	14
4. Dela upp ett stort community eller spel på flera servrar.....	15
4.1. Flash Communication Server	17
5. Sammanfattning och slutdiskussion	18
Referenser	20

1. Inledning

1.1. Syfte

Syftet med uppsatsen är att jag ska fördjupa mig för att göra det möjligt att göra en flash applikation som flera användare kan använda och kommunicera igenom. Efter denna fördjupning ska jag ha bra kunskaper om olika tekniker som gör detta möjligt. Jag ska även veta vilken teknik som lämpar sig bäst till en viss applikation.

1.2. Målgrupp

Målgruppen för detta arbete är jag själv, läraren, andra studenter i kursen samt de som vill veta mer om detta.

Jag gör denna uppsats framförallt till mig själv och andra som ska utveckla en Flash applikation för flera användare. Jag inriktar mig till folk som är insatta i området.

2. Olika kommunikationssätt

Idag blir det allt vanligare med multiplayer online spel, där två eller flera personer kan spela mot varandra. För att göra detta möjligt krävs anslutning och kommunikationsmöjligheter. Jag kommer nu att gå igenom olika lösningar för nätverksprogrammering i programmet Flash. För att kunna ta del av denna information bör man vara relativt insatt i programmet och ha intresse för nätverksprogrammering.

Shockwave filer kan inte kommunicera direkt med varandra eller med en databas utan de måste ha en tredjepart som sköter kommunikationen.

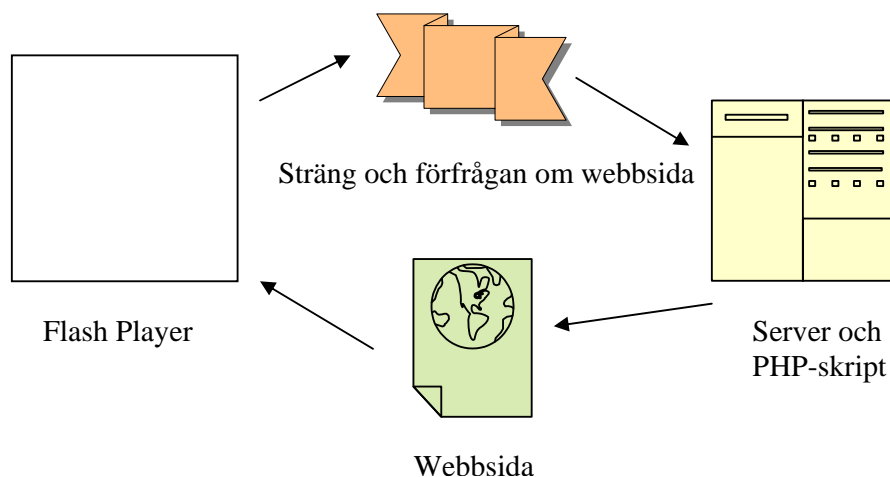
2.1. "Turn based" kommunikation

Enligt Thörning bygger de flesta onlinespel som är gjorda med Flash på så kallad "turn based" –tekniken. Alltså att det är en användare i taget som sköter kommunikationen och den eller de andra användarna väntar på sin tur.

Att skapa en Flash applikation på detta sätt kan vara relativt enkelt. Du kan skapa ett utomstående skript som skickar och tar emot XML eller LoadVars objekt. Sen i sin tur kan skriptet kommunicera med en databas. (Thörning, 2005).

LoadVars gör det möjligt att skicka och ta emot data mellan Flash och en webserver. Det skapar en tillbaka skicknings mekanism som webbservern skickar tillbaka. Det finns två olika sätt som du kan skicka datan med. Antingen är det GET som gör att datan syns i URLen, det är denna metod som jag kommer att använda i texten som kommer även kallad query string. Den andra metoden är POST som skickar data gömt, men POST fungerar bara när sockwavefilen körs i en webbläsare. (Triolo, 2005).

Om vi ska göra ett luffarschack i Flash där vi vill att två spelare ska kunna spela mot varandra så kan vi använda oss av LoadVars och den omgångsbaserade modellen. Nätverkskommunikationen sker då med hjälp av query string eller POST metoden. Ett exempel på det är att shockwavefilen skickar en liten textsträng till ett PHP-skript och sedan läser shockwavefilen av den informationen som den får tillbaka.



Flash kod kan se ut på följande sett.

```

myData = new LoadVars()
myData.load("File.php?Variabel1=" + "text_i_variabel_1" + "&Variebel2=" +
"Text_i_variabel_2")
myData.ref = this
myData.onLoad = function(succes){
    if(succes){
        if (this["Status"] == "OK") {
            //Din Data
        }
        else trace("Error loading data")
    }
}
}

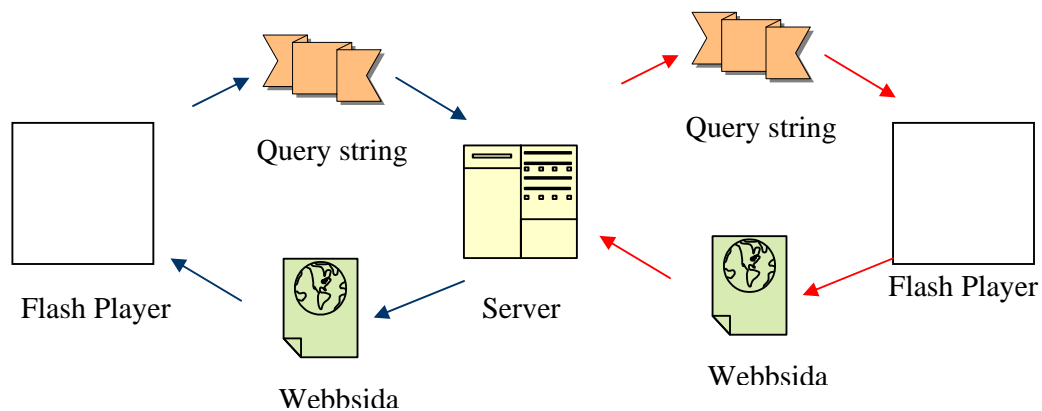
```

Då anropas PHPfilen på följande sätt

"File.php?Variabel1=text_i_variabel_1&Variebel2=Text_i_variabel_2". Denna strängen kallas för query string, där en variabel med namnet "Variabel1" och har värdet "text_i_variabel1".

När PHP-skriptet behandlat variablerna skickar den tillbaka en webbsida som sedan Flash läser av. När denna metoden används kan inte PHP-skriptet anropa Flashfilen själv utan PHP-skriptet måste bli anropat av Flashfilen.

Om man ska göra luffarschacket så måste en flashfil alltid stå och vänta och loopa tills att den får rätt information tillbaka från PHP skriptet.



Enligt Thörning skickas datan på detta sätt över http protokollet. En fördel med denna tekniken är att det externa skriptet kan kommunicera som jag redan har nämnt med en extern källa så som en databas och gör att Flashfilen kan bli påverkad utifrån och flashfilen kan i sin tur påverka en extern källa.

Det skapas en ny anslutning varje gång som Flashfilen gör skickar data och den avsluta direkt efter att datan har skickats eller mottagits. (Thörning, 2005).

Detta gör att det ibland behövs andra sätt att kommunicera på. Den tekniken anpassar sig mest bara för applikationer där latensen inte har någon större roll och ingen jätte stor

data ska överföras. Vissa applikationer till exempel chatt och multiplayer spel kräver en konstant anslutning och då kan detta vara en dålig lösning.

2.2. Socketanslutning

En lösning på problemet som förra lösningen inte gav är socketanslutning. Det ger låg latens, ständig anslutning mellan två eller flera anslutningar.

Flash gör det möjligt att kunna kommunicera över två olika protokoll, TCP, HTTP. Om man räknar med RTMP som ett fristående protokoll så kan Flash kommunicera över tre protokoll.

HTTP står för Hypertext Transport Protocol.

Som tidigare är beskrivet fungerar HTTP på så sätt att en klient ansluter till en server och ber om data. Servern returnerar sedan datan och anslutningen stängs. Till exempel en webbläsare som begär data i form av HTML sida.

Detta är ett effektivt sätt i vissa fall eftersom kommunikationskanalen endast är öppen när data skickas eller tas emot. Men för nätverksspel eller nätverksapplikationer fungerar det inte bra. Detta är för vi behöver göra massor med anrop som till exempel uppdatera spelares positioner vilket kan påverka svarstiderna från servern. Vi har inte "råd" med för hög latens eftersom det i spel kan göra väldigt stor skillnad för varje spelare. Utan det är mer lämpat för "turn based" spel såsom luffarschacket som jag beskrev. (Thörning, 2005)

TCP då, det står för Transmission Control Protocol. Detta protokoll ligger på lägre nivå i ISO modellen över nätverkslagren än HTTP. HTTP är uppbyggt ovanpå TCP lagret och många andra protokoll är också byggda ovanpå det till exempel FTP.

Med TCP kan vi skapa en bestående anslutning och själva säga till när den ska avslutas. Man kan beskriva det med ett telefonsamtal. I ena ändan ringer en upp en i andra ändan och de kan prata fritt tills någon av dem lägger på. Detta sätt passar mycket bättre med en nätverksapplikation.

För att återupprätta detta behöver vi något som heter socket. En socket representerar den punkt där själva nätverkskommunikationen sker rent fysiskt, så att vi kan påverka den med kod. Kommunikation sker över två sådana punkter.

Det finns flera olika typer av sockets men Flash språk ActionScript klarar endast av "stream" socket. Det innebär att man måste skapa en anslutning innan data kan skickas och tas emot. Anslutningen existerar tills du tar bort den.

För att kunna använda XML data måste man använda en klass i Flash som heter XMLSocket. Problemet är att shockwavefiler inte kan kommunicera med varandra själva utan de behöver en server.

Det går inte att skapa en server i Flash, men det finns andra vägar att ta.

- Det går antingen att skriva en server själv i ett språk som klarar av socketanslutningar.
- Alternativ två är att använda en server som redan finns.

Det finns några olika servrar ut på marknaden för tillfället jag har undersökt några av dem.

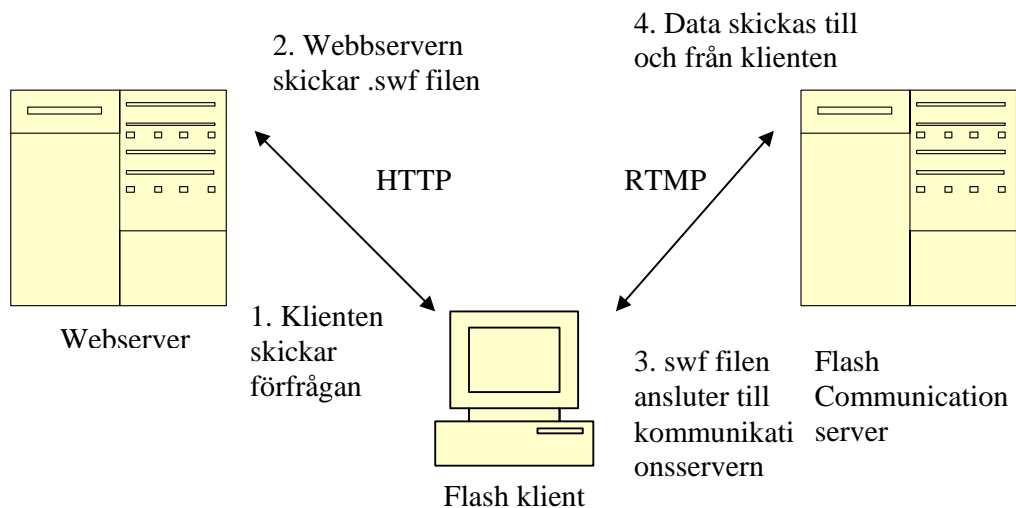
2.2.1. Flash Communication Server

Flash Communication Server är Macromedias egna server, det finns en ny upplaga av den som heter Flash Media Server som hanterar strömmande video bättre, men jag kommer inte att gå in på det något mer.

Med Flash Communication Server kan du kommunicera med två eller fler shockwavefiler i realtid med text, ljud och video. Du kan använda den till att skapa multiplayeronlinespel, communities, instant messaging applikationer, osv. Med Flash Communication Server kan du strömma (stream) data över nätverk till datorer vid Internet, handdatorer och interaktiv TV.

Flash Communication Server använder sig av protokollet RTMP (Real-Time Messaging Protocol). När en klient i detta fall en shockwavefil ansluter sig till servern kommer det en ström av data från och till klienten, även kallad network stream. Denna information kan andra klienter komma åt genom att ansluta sig till servern.

Servern fungerar på det sättet att den skickar vidare all information den får till klienterna. Alltså om du sitter vid en chatt som är en shockwavefil så skickar du ett meddelande till servern som i sin tur skickar iväg samma meddelande till alla anslutna servrar. (Yang, 2003).



Med hjälp av tekniken som Flash Communication Server erbjuder öppnas det möjligheter för att flash applikationer kan kommunicera med .NET och J2EE applikationer. Detta gör att man kan bygga relativt avancerad kommunikations applikationer. Flash Communication Server gör det enkelt att använda databasen ColdFusion i realtid. (Balfour, 2004).

Den senaste versionen av Flash Communication Server heter Macromedia Flash Server 2 och kostar i dagsläget 4759 euro exklusive moms. (Macromedia, 2006).

Jag själv installerade trial versionen av Flash Communication Server men fick det inte att fungera ordentligt, men det var förmodligen min brandvägg som gjorde att den inte gick att använda.

2.2.2. Unity 2, Moocks server

Unity 2 är den första socketservern till Flash som jag kom i kontakt med. Unity 2 är skriven i Java och väldigt enkel att sätta upp och ställa in. Jag testade trial versionen och den är begränsad så att man bara kan ansluta 5 klienter.

Unity 2 är uppbyggd genom klienter, rum, namnrymder, rumattribut, klientattribut fjärranrop. Klienter ansluter sig till servern, Rum är uppbyggt av klienter, Namnrymden är uppbyggd av rum, rumattributen är delad data till/om rummen, klientattributen är delad data till klienterna, fjärranrop kan göras för att kontakta användare eller server.

När det är uppbyggt på detta sätt är det lätt att dela upp kommunikationen så att inte alla klienter behöver belastas av trafiken.

Moocks server gör det lätt att bygga upp applikationer med flera användare trots att jag personligen tycker att deras dokumentation är svår att sätta sig in i och förstå. Men deras gränssnitt av administrationen är enkel och de har flera guider om hur man bygger upp en chatt i flash med hjälp av servern.

Enligt Moock har Unity 2 stöd för följande.

- Hantera upp till 2000 användare beroende på begränsningar av CPU, minnet, nätverket och maskinens tråd begränsningar.
- Grupphantering av användare i olika rum.
- Automatisk delning av data mellan användare.
- Släng ut, porta användare med administrationsverktyg.
- Automatiskt släng ut användare efter att de har varit inaktiva för länge.
- Se anslutna användare och rum i en webbaserad server.

Skillnaden mellan Unity 2 och Flash Communication Server är.

- Flash Communication Server tillåter ljud och video konferenser.
- Unity 2 använder sig av XML för att kommunicera med Flash medan Flash Communication Server använder sig utav protokollet RTMP.
- Unity 2 använder sig utav Java på server sidan med Flash Communication Server använder sig utav server ActionScript.
- Flash Communication Server kan delas upp kluster på flera servrar medan Unity 2 kan enbart köras från en dator.
- Unity 2 tillåter vilken applikation som helst att ansluta som har stöd för TCP/IP socketanslutning medan Flash Communicationsserver är skapt för att kommunicera med Flash enbart.
- Unity 2 kostar 140 US dollar och Flash Communication Server kostar 4759 euro.
- Unity 2 och XMLSocket kan inte hantera krypterad data. (*Moock, 2005*).

2.2.3. Flash now

Flash now är XML socket server som är skriven i C för att linux. Men standardversionen som har stöd på upp till 64 användare kan också köras på en Windows maskin. Enterprise versionen har stöd för upp till ca 1000 användare per maskin men har stöd för att kunna köras på flera datorer och då med flera tusen användare.

Enterprise versionen kostar 495 US dollar men en skola eller akademiska institutioner kan få den gratis. (Flash now, 2002)

Flash now har blivit ett open source projekt som nu alla kan ta del av. Det går att ladda ner det från deras hemsida på <http://www.nowcentral.com/>, där står även

användarnamn och registreringskod. Jag antar att det servern är gratis att använda nu tack vare det har blivit en open source version av det.

2.2.4. Jabber

Jabber är ett annat alternativ som server. Bland annat Disney med <http://www.disneyblast.com> använder sig utav denna server.

Jabber använder sig utav protokollet XMPP (Extensible Messaging and Presence Protocol). Detta är ett öppet protokoll som bygger på XML i realtids kommunikation. Det som Jabber används mest till idag är IM (Instant messaging) så det är inte special designat för Flash.

Prestandan är hög och klarar över 2000 användare samtidigt, men det kan vara svårt att få Jabber att fungera med Flash. Men Disney som är lite av pionjärer av kombinationen Jabber och Flash har sparat in på utvecklingstid när de började använda Jabber som server. En stor fördel är att servern klarar av att kommunicera med många andra medier såsom SMS, WAP, ICQ, MSN, AIM osv. (Lee, 2004).

I en uppsats som Newton Lee (2004) har publicerat står det hur de har gått tillväga för att bygga upp en Jabber server som kommunicerar med Flash, där står det också hur man bygger upp ett tic tac toe spel med två och tre användare.

Det finns lite olika versioner av servern och vissa har öppen källkod och är gratis medan vissa är kommersiella och kostar pengar.

2.2.5. Skriv en egen server

Om man vill ha full kontroll över servern och inte har tillräckligt med pengar över kan man skriva sin egen XML socketserver.

Det ända man behöver är ett programmeringsspråk som klarar av socketanslutningar och det är de flesta, exempel på sådana är Java, C, Visual C++, C# och så vidare. Jag kommer att gå igenom lite enkelt och inte helt fullständigt hur man skriver en server i Java, för det finns mycket att lära och denna uppsats handlar inte om hur man skriver en socketserver.

Jag väljer att dela upp servern i fyra olika klasser.

- Server.Java – Ser till att allt körs och ligger i lyssningsläge
- XMLhandler.Java – Skapas för varje klient som ansluter sig
- Ping.Java – Kollar om anslutningen mellan klienten och servern ”lever”, annars avsluta anslutningen.
- Logg.Java – Loggar trafiken.

Jag använder mig också av XML klasser som finns i jar biblioteker XMLtest.

Före jag börjar med koden är det några viktiga funktioner som ska vara med.

- broadcast – sänder meddelande till alla användare.
- sendToOther – skickar meddelande till alla som inte behandlas av den aktuella tråden.
- reply – skickar tillbaka meddelande till den användaren som använder den aktuella tråden.
- sendToId – skickar meddelande till en användare med ett specifikt id.

Ett exempel på ett XML-element kan vara `<broadcast a="varde" />`. Då ska elementet tas hand om broadcast och skickas ut till alla användare. a är ett attribut.

Ett exempel på Server.Java kan vara

```

import java.net.*;
import java.io.*;
import java.util.*;

public class Server {
public Server (int port, String logfile) throws IOException {
    ServerSocket server = new ServerSocket (port);
    Logg logg = new Logg (logfile);
    while (true) {
        Socket client = server.accept ();
        System.out.println ("Inkommande anslutning från " +
client.getInetAddress ());
        XmlHandler c=new XmlHandler (client, logg);
        c.start ();
    }
}

public static void main (String args[]) throws IOException {
    if (args.length !=2) {
        throw new RuntimeException ("För få argument, Starta
servern med: Server (portnummer, loggfil)");
    }
    new Server (Integer.parseInt (args[0]), args[1]);
}
}

```

Det som händer är att servern får en förfrågan av en klient (den loggas) och då skapas det en ny tråd av XMLHandler. Och om det är för få argument i uppstarten så skrivs ett felmeddelande ut.

I XMLHandler kan byggas upp med följande funktioner.

- kill – dödar tråden, och stänger anslutningen
- run – Pingar och kollar om anslutningen är öppen, läser av datan som skickas till servern (egentligen server bufferten), kontrollera datan, kontrollerar om anslutningen fortfarande lever.
- process – körs när data har kommit. Kontrollerar om elementet är en broadcast, sendToOthers, sendToId, eller setID. Starta sedan rätt funktion.
- broadcast – Skickar XML koden till alla användare.
- sendToOthers – Skickar XML koden till alla förutom den aktuella användaren.
- reply – Skickar XML koden bara till klienten i den aktuella tråden.
- sendToId – Skickar XML koden till ett specifikt Id. (Det kan vara bra att kontrollera om det är någon som försöker fejka och ”hacka” servern här.)
- setId – Skapar ett unikt Id och skickar tillbaka.
- newUser – tala om för den nya användaren att den är ansluten.

Logg skriver helt enkelt ner allt till en fil och loggar alla händelser.

Ping pingar klienten efter vissa förbestämda intervaller för att se att anslutningen fortfarande lever.

Jag hoppas att ni har fått en översiktlig bild på hur man kan implementera en server trots att jag inte har skrivit allt för mycket kod utan mer skrivit vad som bör vara med i en primitiv socketserver.

2.3. XMLSocket i flash

Som ni märker så finns det en hel uppsjö av servrar på marknaden och jag har bara tagit med dem stor kända serverna.

Som jag skrev innan så är själva shockwavefile(.swf) klienten och den måste vara inom samma domän som servern. Om detta inte är så går det ändå att fixa med hjälp av till exempel en policy fil. Stand alone filer fungerar också att köra utanför samma domän.

Klienten ansluter till servern med IP nummer eller domännamn och portnummer. XML socket kan bara ansluta med port 1024 och högre, detta är för att de andra portarna är reserverade för systemet.

I Flash är XMLSocket den klassen som används för att ansluta sig till en socket. För att kunna använda den måste vi skapa ett objekt och detta gör vi med enkelhet i ActionScript med följande kod.

```
var xConnection:XMLSocket = new XMLSocket();
```

Det vi gör här är att vi skapar en variabel som heter xConnection och den är av typen XMLSocket och den tilldelas XMLSocket. Det låter kanske lite oklart för den ovane men det går att göra på flera olika sett. Man får tänka objekt orienterat även om ActionScript inte är objekt orienterat i sig utan endast har stöd för det.

För att kunna ansluta oss till servern måste vi använda oss av metoden connect(), den ger antingen ifrån sig ett sant eller falskt värde (true eller false) beroende på om anslutningen har lyckats eller inte. connect() är asynkron, alltså resterande kod körs samtidigt som en anslutning försöker skapas. Vi får reda på om anslutningen har lyckats eller inte genom händelsehanteraren onConnect().

Koden kan därmed se ut så här.

```
var xCon:XMLSocket = new XMLSocket();
var connected:Boolean = xCon.connect("doman.se", 5001);
xCon.onConnect = function(success:Boolean):Void
{
    if(success)
    {
        trace("Ansluten");
    }
    else
    {
        trace("inte ansluten");
    }
}
```

När anslutningen har lyckats kan kommunikationen mellan klienten och servern starta. Med XMLSockets kan man både skicka och ta emot data. Skicka data görs genom metoden send(). Exempel på detta kan vara.

```
var xmlData:XML = new XML("<xmltest />");
xCon.send(xmlData);
```

Nu behöver vi en funktion som kan ta emot datan också som vi får ifrån servern. Här finns det två alternativ. Det första alternativet är onXML() och det är en gammal ActionScript metod för tidigare versioner av ActionScript. Det andra alternativet är onData() och det är detta alternativet som jag rekommenderar.

När data tas emot körs onData() automatiskt med datan som inparameter. Datan som tas emot är alltid en sträng. Vill man ha det i XML format krävs att ett XML objekt skapas och för att omvandla det kör man parseXML().

När vi inte vill ha en anslutning längre ska vi köra metoden close(). När anslutningen har stängts körs onClose() metoden automatiskt. Detta kan man utnyttja om man vill att något ska hända när anslutningen avslutas.

2.4. Web services

För att använda Web services i flash bör man veta vad web services är för något. Web services är en slags tjänst som kan fördela funktionalitet över Internet. En web services applikation kan till exempel tillåta andra applikationer att komma åt klasser och dess metoder via Internet. De behöver alltså inte ha dessa klasser lokalt på datorn.

I och med att web services fördelas och byggs upp i flera moduler finns det flera fördelar med detta.

- Enkelt att återanvända.
- Kan användas från flera system samtidigt.
- Behöver bara modifieras på ett ställe för att alla ska kunna få del av ändringen.
- Kan köras på flera plattformar.
- Kan köras på Internet.

(Thörning, 2005)

En nackdel med web services och Flash är storleken på datapaketet när de är i XML format de kan då bli ganska stora. Alternativt finns Flash Remoting där datapaketet är ungefär en fjärdedel. *(Thörning, 2005)*

I flash går det att använda web services på två olika sätt. Det ena sättet är med komponenten WebServiceConnector. Det är ett enkelt interface och det mesta sköts i ett grafiskt gränssnitt. Det andra sättet är att använda klassen WebServices där man får bättre kontroll för att man arbetar på en lägre nivå.

WebServiceConnector är som vilken annan komponent som helst. Man drar ut instanser på en keyframe och ge den instansnamn. Sedan går det att binda komponenterna för att kunna ta emot och skicka data. *(Thörning, 2005)*

Det senaste inom att skicka och ta emot data i Flash är Flash Remoting. I Flash Remoting skickas data med AMF paket (Action Messaging Format). I AMF paketen kan det innehålla vanliga datatyper såsom Boolean och Number men även referenstyper såsom Array. Datan är binär som skickas och tar därför mindre plats än XML. Det finns färdiga remoting komponenter att ladda ner från Macromedias webbplats, det finns inte med i programvaran från början.

Några av de nackdelar med Flash Remoting är att det krävs att en Flash remoting gateway är installerad på servern. Ligger man på ett webbhotell kan det vara svårt att få det installerat. Ytterligare en nackdel är att gatewayen kostar pengar.

I augusti 2005 startade det ett Open Source Projekt med AMF över RTMP protokollet. Projektet heter RED5 och finns på webbsidan <http://osflash.org/red5>, projektet ligger än så länge på prototypstadiet.

3. XML

XML är en teknik som man bör ha koll på när man ska kommunicera med XML socket i Flash, även om man inte behöver kunna allt är det bra att ha en uppfattning om vad XML är och kan använda det till.

XML står för Extensible Markup Language. Tidigare fanns ett stort behov av en standard för strukturering av data. 1998 kom XML och blev snabbt en utbredd standard för att utbyta data. XML är plattformsoberoende. Plattformarna behöver bara något som kan tolka XML. Med XML kan man strukturera data. XML kan man definiera obegränsat antal taggar eller element som Tindale i boken Flash XML StudioLab vill att man ska kalla det när man pratar om XML. Detta är en skillnad mot HTML där de har fördefinierat antalet taggar och ett begränsat antal, i XML bestämmer man namnen själv i elementen. I HTML kan det en rubrik se ut på följande sätt `<h1> Victor Wickström </h1>` medan i XML kan det se ut på följande sett `<namnrubrik> Detta är en rubrik </namnrubrik>`.

HTML kan bara presentera data i en webbläsare medan XML kan hanteras i många olika applikationer. XML är strikt, alltså det måste se ut på ett visst sett, till exempel måste det finnas en sluttagg. HTML är förlåtande och det går bra att missa lite här och där och därför kan det se lite olika ut i olika webbläsare.

XML slog igenom 1998 och används i dagsläget nästan överallt där data ska struktureras.

XML är uppbyggt på det sättet att den har rot element sedan har den barn och eventuellt barnbarn. Man kan likna detta med en trädstruktur.

```
<uppsats>
  <syfte>
    <rubrik> bla bla bla </rubrik>
    <brödtext> bla bla bla </brödtext>
  </syfte>
  <innehållsförteckning>
    <rubrik> bla bla bla </rubrik>
    <brödtext>bla bla bla </brödtext>
    <sidnummer> 12 </sidnummer>
    <brödtext>bla bla bla </brödtext>
    <sidnummer> 13 </sidnummer>
  </innehållsförteckning>
  <innehåll>
    <rubrik> bla bla bla </rubrik>
    <brödtext> bla bla bla </brödtext>
  </innehåll>
</uppsats>
```

Detta exempel kan vara strukturen på en uppsats, då lite förenklat. Elementet uppsats är då roten och har då 3 barn under sig (syfte, innehållsförteckning och innehåll) dessa har barn i sin tur. Till exempel har syfte en rubrik och en brödtext.

Nu har vi fått lite kött på benen i alla fall så nu börjar jag gå in på Flash och XML.

Flash har stöd till att både skicka och ta emot XML. Man kan också skapa och modifiera XML. I Flash finns det lite olika klasser och funktioner som man bör känna till när man jobbar med XML.

- XML - objekt av denna klass används för att fyllas med data. Denna klass kan även skicka och ladda data.
- XMLNode - klass som innehåller definitioner för alla element. Den kan inte skicka eller ladda data.
- load() – funktion för att ladda in XML från en extern källa. Dessa data laddas asynkront. Alltså övrig kod fortsätter att köras samtidigt som datan laddas. Detta kan leda till att applikationen hänger sig.
- send() – funktion som man kan skicka XML med.
- sandAndLoad() – kan ta hand om svar efter sändning.
- LoadVars – detta är en klass som jag har nämnt innan och den är lik XML men XML är specifik för just XML. Men det går även att skicka data med denna. LoadVars används främst vid data med namnvärde par till exempel namn=Victor%20Wickstrom&kurs=Virtuella%20Miljoer.
- getBytesLoaded() – detta är en funktion som både finns i XML och LoadVars och den talar om hur mycket data som har tagits emot.
- getBytesTotal() – denna funktion finns också med i både XML och LoadVars och den talar om hur mycket data som ska tas emot.

Som jag har skrivit innan i uppsatsen kan Flash bara ta emot och skicka data från samma domän som shockwavefilen (.swf). Detta är på grund av säkerhetsskäl. Men man kan komma runt detta på tre olika sätt.

- Skapa en policy fil, Denna policyfilen måste heta crossdomain.xml för att Flash ska kunna tolka och använda den. I denna fil anger man vilka domäner som man vill att servern ska ha åtkomst till. Policyfilen ska ligga i roten av servern som shockwavefilen ska ha åtkomst till.
- DNS alias, få en domän att verka vara en som är godkänd.
- Använda ett proxyscript som vidarebefodrar data. Proxyskriptet ska ligga på samma domän som shockwavefilen. Du anropar då proxyskriptet från shockwavefilen som ligger på samma domän och i sin tur tar det kontakt med det som ligger på den andra domänen.

Om du har en shockwavefil lokalt på din dator brukar det inte vara något problem, men om sockwavefilen körs från en server har flashfilen begränsad återkomst på grund av säkerhetsskäl.

4. Dela upp ett stort community eller spel på flera servrar

Bartle skriver om att det är en grej att ha 100,000 spelare på 50 servrar än att ha 100,000 spelare på en server är något helt annat. Det kostar mindre att köpa in åtta datorer med kraften X än att det är att köpa in en dator av kraften 8X. Detta sätt att tänka har kommit för att stanna för en virtuell värld är för stor att passa på en dator, därför måste det partitioneras ut på flera servrar. (Bartle, 2004)

Om man använder flera servrar och en spelare vill flytta dens karaktär från A till B, då måste subservern låsa placering B, kolla om den är fri, om den är det, flytta spelaren till placering B, då låsa upp den. Placering A måste också vara låst så att alla som vill göra något med karaktären tror att den är på placering A. (Bartle, 2004)

Detta är ett sätt hur man kan tänka när man ska implementera en virtuell värld på flera servrar. Vid flera serverar vill vi sprida spelarna jämt över alla subserverar. Detta för att öka prestandan över alla servrar. Det är onödigt om en server går på 5% och en annan går på 100% och kanske brakar ihop för att den subservern med som arbetar 100% inte hinner med. Man måste balansera ut allt på olika servrar, det finns två olika tillvägagångssätt för det. Det ena är fixed load balancing och det andra är dynamic load balancing. (Bartle, 2004)

Enligt Paul Willys så sker laddningen i dynamic load balancing i enligt serverns förmåga att handskas med det, snabba maskiner får mer att göra. Fixed load balancing skickas lika stor del av trafiken till varje maskin, slöa maskiner kan bli överladdade medan snabba maskiner sotar igen. (Willys, 1999)

Fixed load balancing

- Det är lätt att implementera
- Kan portionera ut det på flera subserverar för större effektivitet.
- Tillåter klienter att räkna ut vilken textur som behövs och förladdar dem till grafikortets minne. (Bartle, 2004)

Dynamic load balancing

- Har tät terräng, du kan se horisonten
- Har inga fysiska gränser, alltså monster fastnar inte i gränser
- Balanserar laddning bättre. (Bartle, 2004)

Bartle skriver om Jack the Ripper kunde gå över en gräns för att polisen inte kunde ta honom skulle han använda en annan teknik. Om man använder sig av fixed load balancing och poliserna fastnar i gränserna går det bara gå över gränsen och ens taktik förändras gent emot om man skulle använda sig av dynamic load balancing och poliserna skulle följa med över gränserna och bli jagad över allt. (Bartle, 2004)

Bartle skriver att man alltid får räkna med så kallad lag eller fördröjning. Även om man har en perfekt lina till servern så blir det en fördröjning. Detta är något man måste räkna med när man gör en virtuell värld. Kommer användare som sitter med perfekt lina i rummet jämte servern att få fördelar jämfört med någon som sitter på satellit på andra sidan jordklotet? Om det finns tekniska aspekter som gör att man kan få fördelar så kommer vissa spelare att dra nytta av det eller kanske utnyttja det. (Bartle, 2004)

Vilka begränsningar finns det på nätverkskommunikationen? Var går den kritiska gränsen på hur många användare som en singel spelvärld klarar av?

Implementeringen av multiplayer-nätverks-funktion på ett kortspel eller liknande spel där intervallet mellan de olika spelarna är långa är relativt enkel. Ju mer nätverksspelet bygger på snabba drag, spelarnas reaktioner och realtidshändelser är desto svårare att implementera och få timingen att fungera.

Det är därför viktigt att börja designa med det värsta scenariet i åtanke. Om spelet fungerar över normal Internetanslutning så blir det inte svårt att skapa ett "turn-base" variant för LAN. (Bartle, 2004)

När du designar ett multiplayer spel är den mest utmärkande aspekten att ta itu med nätverksverksprästandan. Det brukar definieras av två saker latens och bandbredd.

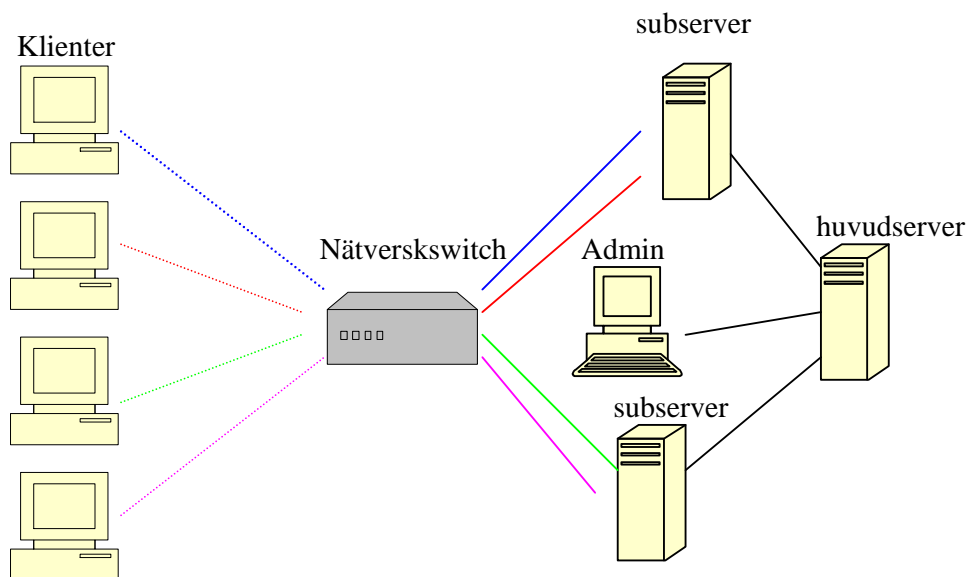
Latens menas med den tiden som ett paket färdas från sändaren till mottagaren, (klient – server, server – klient eller klient – klient). Med bandbredd menas hur mycket data som kan skickas via nätverkets kablar inom ett tidsintervall.

Ett problem är att alla som är anslutna inte har samma anslutning. De med bäst anslutning kan dra fördel med att få data snabbare. En lösning på detta problem är att skicka data till dem som har sämst anslutning först. (Bartle, 2004)

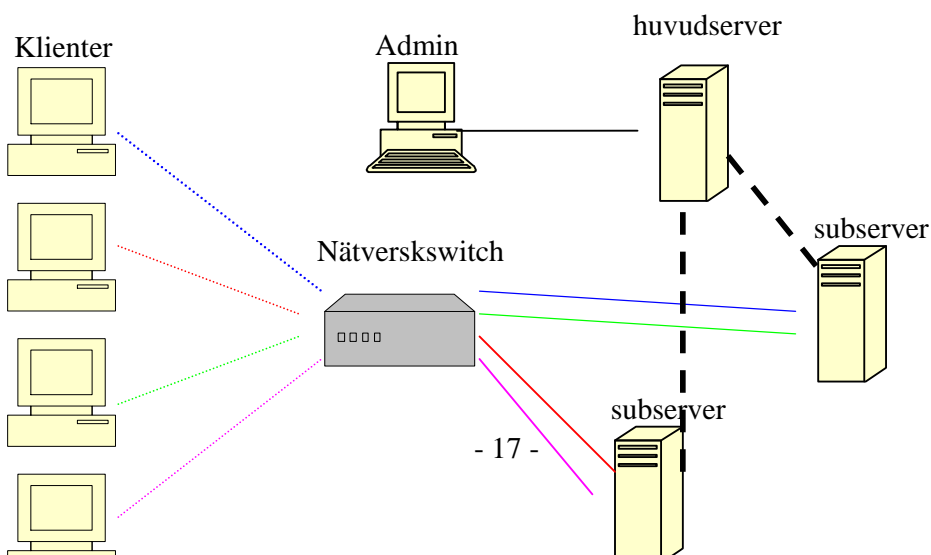
4.1. Flash Communication Server

Det finns flera olika metoder för att sätta upp Flash Communication Server. Antingen har man en server eller ha flera servrar. Det finns flera olika skolor hur man ska sätta upp Flash Communication Server på det bästa sättet. Jag kommer att beskriva tre olika möjligheter

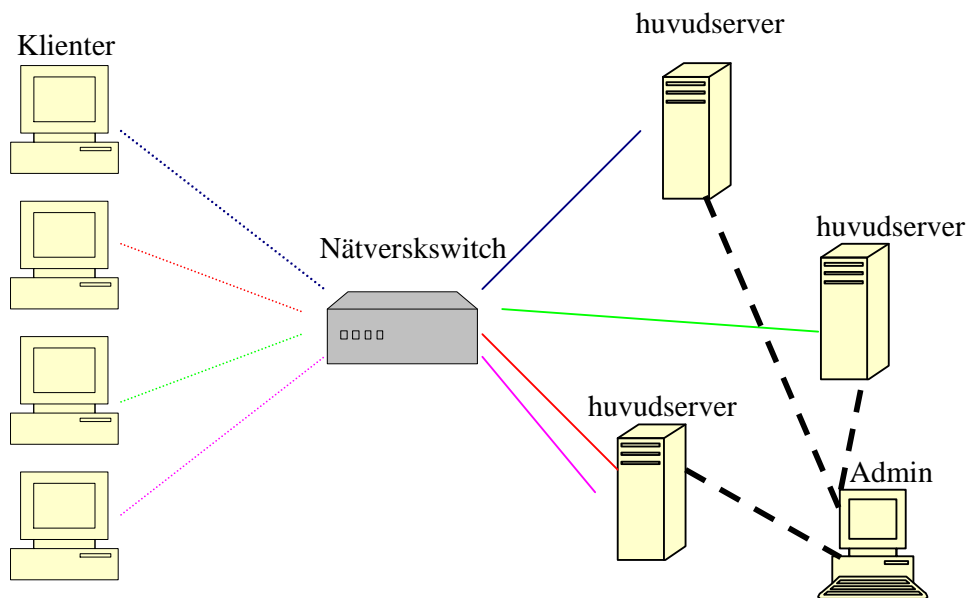
- Att ha en huvudserver som är ansluten till subserverar som i sin tur är anslutna till klienterna.



- Ha flera servrar som antingen kan uppföra sig som huvudserver eller subserver beroende på vem som återupprättade anslutningen först.



- Ha flera huvudservrar som inte är bunden till andra men har en anslutning till en admin.



De olika lösningarna är bra till olika applikationer. Men den sista lösningen kan lämpa sig bäst i några fall. Ett exempel är om en server kraschar då kan de andra servrarna ta hand om det. Nätverksswitch tar då hand om load balancing om den är rätt konfigurerad. Switchen i sin tur ska ligga och pinga servrarna och sen om de kraschar om de gör det så ska de automatiskt styra om trafiken till en annan server. (Hilton, 2005)

John Hilton har skrivit en tutorial hur man gör för att få detta att fungera i Flash på adressen http://www.macromedia.com/devnet/flashcom/articles/clustering_03.html.

Flash Communication Server delar ut kluster på andra servrar.

5. Sammanfattning och slutdiskussion

Vilken lösning av alla ska vi välja när vi ska skapa vårt multiplayer online spel eller i uppbyggnaden av vårt community?.

Svaret är inte det enklaste, men först får man tänka efter vad det är för spel som vi ska göra. Ska vi göra ett spel som det inte spelar någon roll vad vi har för latens till exempel ett luffarschack kan vi implementera det med LoadVars och ett skriptspråk. Men om man har 40 000 kr över och en server så kan man installera Flash Media Server på den. Men det känns lite överkill.

Om man ska bygga ett spel som kommer att beröra mindre än tusen användare samtidigt kan det vara en idé att titta på alternativen till Flash Communication Server /Media Server, just på grund av kostnadsfrågan. Det man bör ha i åtanke är vad för begränsningar de olika servrarna har och vad mitt spel kräver samt finns det någon server som jag kan installera detta på. Om det inte finns någon server som man kan installera det på kanske sista utvägen är att använda LoadVars ändå.

Det verkar som att Flash Media Server är överlägsen alla andra val om man bortser från priset. Därför tycker jag att man som en seriös aktör och ska göra något kommersiellt med många användare borde välja Flash Media Server. Detta på grund av att du inte vill att någon bugg eller något ska göra så att ditt spel ska stå still, om du har problem finns det många där ute som håller på Media Server och du kan säkert få bra

support av Macromedia. Sen använder sig Media Server sig utan AMF som gör att din bandbredd blir mindre för att de data som skickas tar bara en fjärdedel av storleken så i längden kan du kanske spara in de kostnaderna på det. Media Server har också stöd att dela upp din server på många olika servrar genom klusterfördelning. Sen är det säkerhetsaspekter också, det går att kryptera den data som du skickar och man kan köra över https. För du vill inte att personuppgifter ska komma ut eller att någon ska hacka ditt spel. Allt detta och mycket mer gör att Flash Media Server är överlägsen. Men det ska bli intressant och se vad open source versionen RED5 kommer att lyckas med. De ska redan ha gjort så att det går att sända video över den och det är bara fler och fler som ansluter sig till projektet.

Men om inte väljer att använda Flash Media Server eller Flash Communication Server så finns det flera vägar att ta.

Det som jag tycker att man ska tänka över om man ska göra en multiplayer applikation är:

- Vilka naturliga avgränsningar kan man göra för att kunna dela upp applikationen på flera olika servrar?
- Vilka möjligheter finns det att bygga ut?
- Vilken bandbredd finns kräver applikationen? Går det att bygga ut den?
- Finns det någon tillgång till en server, eller ska det läggas ut på ett webbhotell?
- Vad finns det för budget?
- Hur kommer användaren att se på allt? Hur kommer användaren och spelet få för konsekvenser för ditt val?
- Hur många användare förväntas det att bli? Går det att bygga ut iså fall?

Det går också att kombinera olika alternativ. Du kan kommunicera med LoadVars med ett PHP-skript med sån data som inte latensen har någon stor roll och sedan kan du ha skrivit din egen server i Java över XML socket som sköter realtidskommunikationen.

Du kan också dela in din värld i olika zoner och varje zon sköts av en server. Om det börjar bli för många användare kan du bygga ut världen med mer zoner. Något som man bör tänka på då är att det ska verka naturligt. Om ett berg som avgränsar världen helt plötsligt försvinner för att världen byggs ut tror jag inte att alla användare köper det och vissa till och med kanske tappar känslan och slutar spela.

Om du har delat upp din värld i många olika zoner och scener så kommer det kanske bli en liten laddningstid mellan serverbyte/zonbyte, hur ska du lösa det?

Lever har ett förslag om att du skriver ut information till användaren, det kan vara om den nya zonen eller om spelkontroller eller liknande, för en helt tom ruta är inte så roligt och väntetiderna känns mer outhärdliga då. (*Lever, 2004*)

Kommer spelare att utnyttja spelets svagheter om du delar in det i zoner? Kommer spelare att gömma sig i en zon för att sedan göra ett snabbt bakhåll? Hur ska du lösa uppdateringen av allas positioner när du kommer in i den nya zonen? Vad händer om en server går ner som representerar en zon?

Det finns många frågor som man måste svara på innan man har tänkt ut sin lösning. Jag vet ingen universallösning på problemen och det finns förmodligen ingen heller.

Referenser

AMFPHP (Elektroniskt)

Flash remoting for PHP: A responsive Client-Server Architecture for the Web (2005)

Tillgänglig: < <http://www.amfphp.org/> > (2006-01-06)

Balfour Julia Kathleen (Elektroniskt)

"SCRYTCHAT": a chat room and quarterly publication dedicated to collaborative creative writing within the third medium (2004)

dec12.pdf

Tillgänglig: < <http://a.parsons.edu/~julia/thesis/dec12.pdf> > (2006-01-06)

Bartle, Richard (2004). *Designing Virtual Worlds*. USA. New Riders Publishing

Castro, Elizabeth. *XML: visuell snabbguide* UK. Frank O'Donel

Chambers, Mike (Elektroniskt)

Macromedia Flash MX Security (2002)

Macromedia white paper

Tillgänglig: < <http://www.macromedia.com/devnet/flash/whitepapers/security.pdf> > (2006-01-06)

Comer, Douglas E.(2004) *Computer networks and Internets with Internet Applications*. 4th Edition. USA.

Pearson Education International

Deitel H.M, Deitel P.J, Gadzik J.P, Lomeli K, Santry S.E, Zhang S (2003) *Java Web services for experienced programmers*.

USA. Pearson education, Inc.

Ek Jesper, Ekman Rasmus (1998) *JAVA-programmering*. Stockholm. Pagina

Flash Now (Elektroniskt)

Flash Now (2002)

Tillgänglig: < <http://www.nowcentral.com/> > (2006-01-06)

Lee Newton, Disney Online (Elektroniskt)

Jabber for Multiplayer Flash Games (2004)

p13-lee.pdf

Tillgänglig: < <http://delivery.acm.org/10.1145/1040000/1037872/p13-lee.pdf?key1=1037872&key2=8317656311&coll=GUIDE&dl=GUIDE&CFID=64360898&CFTOKEN=90373362> > (2006-01-06)

Lever, Nik. (2004) *Flash MX 2004 Games Art to ActionScript*. Great Britain. Newgen Imaging System (P) Ltd.

Macromedia, Inc (Elektroniskt)

Macromedia (2006)

Tillgänglig < <http://www.macromedia.com/> > (2006-01-06)

Moock (Elektroniskt)

Moock >> unity (2005)

Tillgänglig: < <http://moock.org/unity/> > (2006-01-06)

Powazek, Derek (2002) *Design for Community*. USA. New Riders Publishing

Preece, J. (2000). *Online Communities: Designing Usability, Supporting Sociability*. West Sussex: John Wiley & Sons

Thörning, Kim (Elektroniskt)
Nätuniversitetet – Flash & ActionScript
Föreläsning 10
Kräver användarnamn
Tillgänglig < <http://www.nu.hik.se/hik> > (2005-08-05)

Tindale Ian, Macdonald Paul, Rowley James. (2001) *Flash XML StudioLab*. USA. Friends of ED.

Triolo, Helen (2005) *Using an XML file with Flash*
A Flash / ActionScript resource site
Tillgänglig: <http://flash-creations.com/notes/dynamic_xml.php> (2006-01-06)

Tupper Luke (2002): *XML FAQ*
Tillgänglig: <<http://www.tupps.com/flash/faq/xml.html>> (2006-01-06)

Willys Paul (1999): Load Balancing
Digital Point Solutions
Tillgänglig: <<http://www.digitalpoint.com/lists/19435.html>> (2005-12-15)

Wolfgang Bernatz, Wolfgang Finke (Elektroniskt)
Chapter 21: Flash Remoting
Tillgänglig: < [http://svrwinf02.bw.fh-jena.de/knowledge_base/winf_library.nsf/22a43014d99830a1c1256ce10067412a/581838b963e1f8ddc1256d20002ef61d/\\$FILE/chapter21.pdf](http://svrwinf02.bw.fh-jena.de/knowledge_base/winf_library.nsf/22a43014d99830a1c1256ce10067412a/581838b963e1f8ddc1256d20002ef61d/$FILE/chapter21.pdf) >
(2006-01-06)

Yang Te Ling (Elektroniskt)
TAGS (2003)
thesis1.pdf
Tillgänglig: < http://dt.parsons.edu/thesis_archive/m2003/Yang_Teling/thesis1.pdf > (2006-01-06)