Streaming Media Division

WMT Eclipse Functional Specification

Intelligent Buffering Specification

Version 1.0

Stability: Closed

Authors: Troy Batterberry

File Location: \\showtime\docs\Platform\V5\Specifications\Network\Intelligent Buffering Specification.doc

Core Team

Development	David Stewart, Sanjay Bhatt, Anders Klemets, Scott Colville, Dawson Dean
User Education	TBD

1 AUDIENCE AND DISCLAIMER 1
4 OVERVIEW 1
4.1 On-Demand Startup Latency Problem Statement 2
4.2 Broadcast Startup Latency Problem Statement 2
4.3 Network Congestion/Reliability Problem Statement 2
5 CUSTOMER SCENARIOS 2
5.1 Corporate Network Presentation Scenario 2
5.2 Broadcast Video Content Surfing Scenario 3
5.3 Internet Radio Content Surfing Scenario 4
5.4 Cable Modem Streaming Scenario 4
5.5 Wireless Network Streaming Scenario 5
5.6 Satellite Network Scenario 5
6 REQUIREMENTS AND FEATURES 6
7 FUNCTIONAL SCENARIOS 6
7.1 Fast startup 6
7.1.1 On-demand content 6
7.1.2 Broadcast content 8
7.2 Improved ability to withstand occasional brownouts 8
8 FUNCTIONAL OVERVIEW 9
8.1 Configurable Settings 9
8.1.1 Server Options 9
8.1.2 WMF SDK Options 10
8.1.3 Client Options 10
9 ECLIPSE DELIVERABLES 10
9.1 WMSDK Support 10
9.2 Server Support 10
9.3 Encoder 11
9.4 Client 11
10 COMPATIBILITY 11
11 OPEN ISSUES 11
11.1 Dealing with the 1 Mbit/s B/W detection limit. 11
12 EDIT HISTORY 11

1 AUDIENCE AND DISCLAIMER

This document is MICROSOFT CONFIDENTIAL– appropriate care should be taken when distributing copies. The intended audience is members of the DMD development group. Others may read this document at their own risk.

2 DOCUMENT SCOPE

This document details the intelligent buffering feature that allows clients to buffer streamed data at rates proportional to the available network bandwidth. This technique is used to reduce latency and also to improve the ability to withstand temporary network congestion.

It should be noted that some of the functionality discussed in this document already exists in the Artemis/Cyprus and Hercules code bases. For instance, accelerated streaming of buffered data is currently implemented for Artemis-based clients streaming from on-demand publishing points located on Hercules servers. This specification attempts to address improvements in the current limited implementation in order to fully address the various customer requirements related to intelligent buffering.

3 GLOSSARY OF TERMS

End-to-end latency: The time from when the input data (raw media) enters the Windows Media platform and the time the data is actually rendered on the computer.

Latency (generic definition): The time that elapses between a stimulus and the response to it.

Multi-Bitrate (MBR): A file that contains a program encoded with multiple streams each with a different bandwidth. When used with a streaming media server, the client automatically selects the highest bandwidth streams that will successfully traverse the network link between the server and the client. The client can also temporarily downshift to lower bandwidth streams when congestion is encountered.

Progressive Download: A feature that allows the player to download data and play it simultaneously without the use of a streaming media server.

Round Trip Time (RTT): The time taken for a packet of data to travel from the source computer to a destination computer and back.

Startup Latency: The time that elapses between the user request for an URL and when rendering actually starts.

4 OVERVIEW

The current implementation of buffering logic is inadequate for a variety of markets. The following section details areas that would benefit from intelligent buffering algorithm enhancements.

4.1 On-Demand Startup Latency Problem Statement

Startup latency continues to be a significant deterrent to the deployment of streaming media in Intranet presentation scenarios. Startup latency is considerably longer with existing streaming media products than with the use of conventional delivery mechanisms such as the network file redirector or the web server (and "progressive download" delivery). Customers are therefore choosing not to deploy streaming media services in their corporate networks for on-demand presentations. Solving this problem will help increase the customer base of streaming media servers in the corporate presentation market.

4.2 Broadcast Startup Latency Problem Statement

Startup latency for video broadcast sources is generally worse than video on-demand sources. A client must wait for a key frame to start buffering useful data. Depending on when the client connects to a broadcast source and also the frequency of key frames in the encoded media, startup latency can often exceed 10 seconds even on LANs and broadband networks. The concept of surfing the Internet to browse broadcast video content is severely hampered by this problem.

4.3 Network Congestion/Reliability Problem Statement

Many emerging network topologies have significantly higher error rates and outage problems. For instance, the cable modem networks deployed by AT&T and Time Warner reportedly suffer from frequent temporary outages. These outages can last from less than one second to 15 seconds or longer.

Wireless network topologies also result in problems when attempting to stream content. Temporary outages often drain the small client data buffer and ultimately result in frequent rebuffering. Because these networks often have a rather long RTT, the ability to successfully satisfy a resend request in time is often unachievable in the current platform implementation.

5 CUSTOMER SCENARIOS

The following scenarios will be supported. Many supported derivative scenarios also exist.

5.1 Corporate Network Presentation Scenario

The "helpdesk presentation" created for the marketing team clearly demonstrates the current problems associated with startup latency. The presentation is designed to run on from the following sources:

1. CD.
2. Hard drive.
3. Network drive.
4. Network UNC path.
5. Web server. (ASF content is progressively downloaded).
6. Web server with WMS. (ASF content is streamed).

Unfortunately, latency is significantly worse with a WM Server than any other source type using a LAN or broadband topology. This is contrary to conventional wisdom that suggests one of the major benefits of streaming media is the file does not need to be downloaded and therefore has "faster" startup times. Startup latency delays of 5-7 seconds are common in this presentation. Startup latencies of less than 2 seconds on LANs are required for reasonable viewing. Depending on caching hits, startup latency for network file shares and web servers are anywhere from less than one second to a few seconds on an LAN.

Use of a WM Server also offers other benefits for presentations. It is currently the only scenario that fully handles MBR content. It can therefore stream the appropriate content bandwidth to regional offices over WAN links or remote users through Remote Access Software (RAS). Network file shares and web servers do not currently offer this capability.

5.2 Broadcast Video Content Surfing Scenario

WindowsMedia.com has assisted the user in finding compelling content to watch and enjoy. With the increasing deployments of broadband networks and the more rich media available to watch, consumers are much more likely to spend time simply browsing (i.e. surfing) multimedia content.

Surfing is an extremely common paradigm used by consumers to enjoy multiple television channels, radio stations, and even web sites. As the latency associated with changing media sites increases, customers become less interested in the activity. Less interest obviously translates into less usage and subsequently less advertising revenue. Furthermore, the time spent by the consumer waiting for the player to buffer could instead by filled with advertisements. While some creative solutions mask buffering time through the use of animations, etc, initial buffering time is extremely wasteful for true multimedia content.

The following is an approximate breakdown of the causes of startup latency from a video broadcast:

Open File (initial handshake, transfer headers, etc)	0.4 - 2 seconds
Wait for key frame (key frame every 8 s)	0 - 8 seconds
Default buffering time	5 - 6 seconds
Total	**6 - 16 seconds**

In order to make broadcast video content surfing much more enjoyable, the startup latency must be driven lower. Startup latencies of less than 1 second are typical in television scenarios such as DirecTV and other MPEG-2 delivery schemes. While this is desirable, it is not practical given the current constraints of bandwidth, CODECs, and other existing platform infrastructure.

It is reasonable and achievable to target startup video broadcast latencies of 2 seconds or less on LAN and broadband environments for many scenarios. (These scenarios include 100-

300kbit/s video streams). This improvement would make a dramatic difference to the end user. It can be accomplished in the existing Eclipse deliverables with minor design revisions.

5.3 Internet Radio Content Surfing Scenario

This scenario is similar to the user who uses preset stations on a conventional radio to surf through audio programming. Unfortunately, unlike conventional radios, the Windows Media platform components produce average startup latencies that vary from 5-10 seconds. Again, this is lost opportunity for a variety of usage statistics and potential revenue. Users will simply have more time to stream from more radio stations if startup latencies are reduced. This translates into more potential unique Windows Media streams and therefore increases some of the relative statistics associated with the Windows Media platform.

Internet radio programs are generally built using a broadcast publishing point. (There are notable exceptions, but the majority of our big customers still employ a broadcast model). Reducing the startup latency for the audio broadcast scenario therefore has significant relevance. Driving the startup latency down to values less than 2 seconds on LAN and broadband networks would noticeably affect the user in a positive manner. It would also provide a compelling reason for customers to upgrade to the Eclipse platform.

5.4 Cable Modem Streaming Scenario

Cable modem networks reportedly suffer from occasional brief network outages. The current default buffering time for the WMP is set to 5 seconds. If the outage lasts longer than a few seconds, the user will experience rebuffering. Lengthening the buffering time on the client has the potential benefit of increasing the resilience of the WMP to handle longer outages without rebuffering. However, without other algorithm changes, simply lengthening the buffer duration will proportionally increase the startup latency.

Cable modem networks are not easily detectable at run time. A majority of cable modem users utilize a standard Ethernet network card to connect their computer to the cable modem. Therefore, whatever solution is employed, it must meet the needs of cable modem users without adversely impacting users located on other network topologies. Requiring the user to enable manual settings in order to make the WMP function properly on a cable modem network should be avoided whenever possible.

Cable modems generally have high average throughput values. The additional available bandwidth can be used to reduce the time required to send the data needed to fill the client buffers by accelerating the initial transmission rates. By buffering more data and by using the additional bandwidth often available on cable modems, the WMP will be able to withstand longer temporary outages with no impact to the startup latency.

The SDK networking code shall be designed to request the first 10 seconds of the stream to be sent at an accelerated rate determined by bandwidth detection logic. However, the WMP will still start rendering after 5 seconds of data (default player value) are present in the buffer. Assuming the difference between the encoded bitrate of the stream and the actual

bandwidth available have a distinct difference, the player will benefit from receiving the buffered data at an accelerated rate without incurring additional startup latency. The startup latency will not degrade as a result of extending the player buffer duration even if accelerated transmission does not occur. The player will still be able to start rendering with the same duration of data present in the buffer as previous versions of the client. However, in such cases, the player will be less resilient to network outages.

For instance, a typical cable modem network user attempting to stream a 300 kbits/s file may obtain a bandwidth detection of 1 mbits/s. Assuming accelerated transmission is employed, the player will have sufficient data to begin rendering 1.5 seconds after the data transmission begins. After 3.0 seconds of data transmission, the player will have 8.5 seconds worth of data present in the buffer. (Since the player began draining the buffer at 1.5 seconds, it will have consumed 1.5 seconds worth of data). Note that at 3.0 seconds, the server will cease accelerated transmission and resort to normal transmission that matches

If the UDP protocol is employed for data delivery, other existing changes must be made to the streaming logic in order to improve the ability to withstand longer network outages. The server currently defaults to maintaining only a few seconds worth of data available for retransmission requests. As the outage durations lengthen, the server shall be required to keep more data present in order to successfully satisfy retransmission requests.

5.5 Wireless Network Streaming Scenario

Wireless users suffer from similar occasional outages experienced by cable modem users as well as increased bit error rates. Wireless users are also often plagued with a large RTT. This means the client buffer time needs to be increased and the server needs to be able to successfully satisfy older resend requests in order to reduce the impact of temporary outages or bit errors.

Many of the smaller wireless appliances being designed and manufactured today will not be able to increase the buffering time due to severe memory constraints. Given all the other constraints on these devices such limited additional available bandwidth, little can be done to solve problems associated with longer outages. Temporary outages on such devices will simply result in rebuffering.

5.6 Satellite Network Scenario

Satellite networks suffer from similar conditions experienced by cable modem users and also wireless users. Satellite networks are also notoriously latent. Customers have reported RTTs in excess of 3 seconds. Such excessive RTTs require the server to increase the current amount of data that is maintained for the purpose of satisfying resend requests from a few seconds to approximately 8 seconds.

6 Requirements and Features

1. The WM Server shall maintain 8 seconds worth of data for satisfying UDP resend requests in both on-demand and broadcast scenarios in a default configuration. This duration value shall be configurable in the server namespace.
2. The WM Server shall support the ability to stream data to the client at a rate and duration requested by the client. For instance, the client shall be able to select the 300 kbits/s stream and request the first 10 seconds of the stream to be sent at 1 mbit/s.
3. The server shall govern the total bandwidth used (at both a publishing point level as well as a server level). It shall prevent any limits being exceeded by client requests. In the case where a client requests a 300 kbit/s stream to be sent at 2 mbits/s, and only 300 kbits/s of total server bandwidth are remaining, the server shall simply stream the file at 300 kbits/s. Buffering will not occur at an accelerated rate in this situation.
4. Performance counters used to track the current bandwidth shall accurately reflect true transmission rates even when intelligent buffering requests are employed.
5. The server shall have a configurable option to disable the transmission of data at rates greater than the encoded bitrate. It shall be enabled by default.
6. The server shall maintain a buffer of data on broadcast publishing points that is available for accelerated transmission when new clients connect. The default size of the buffer shall be at least 10 seconds. The duration of the buffer shall be configurable in the namespace and the server administration UI.
7. The accelerated rate requested by the client shall be determined by bandwidth detection logic. The rate requested by the client is configurable by using the existing client configuration options available to override automatic bandwidth detection.
8. The amount of buffered data required before the client rendering initially starts shall be configurable on the client. (Note this functionality already exists in both the v6.4 and v7 clients).
9. Data shall not be sent at an accelerated rate when a stream switch occurs. Stream switches are representative of network congestion changes. Therefore, additional bandwidth should not be used in such scenarios.
10. By default, the client shall request the data be accelerated at a rate equal to 85% of the detected link bandwidth. In the case where the streams selected exceed 85% of the link bandwidth, accelerated buffering shall not be used.
11. Intelligent buffering shall be employed during the following conditions:
    1. Initial open request.
    2. A start request and the client buffer are not full.
    3. A seek request.
    4. A stride request.
12. Intelligent buffering requests shall not be employed when the aggregate limit exceeds

7 Functional Scenarios

The following functional scenarios describe the impact of intelligent buffering.

7.1 Fast startup

7.1.1 On-demand content

When a user requests an on-demand file, the first 10 seconds worth of data shall be fetched by the server and transmitted at the rate requested by the client. After the first 10 seconds are sent, subsequent data shall be transmitted at rates equivalent to the encoded rate of the file.

The client shall start rendering content when 5 seconds worth of data are present in the client buffer. (Note this value is dependent upon the client buffer setting and also the preroll value present in the ASF file). In the case of a broadband user opening a 56 kbits/s WMA file, the following examples demonstrates the behavior that shall take place:

a.) Example where intelligent buffering reduces startup latency:

00.00	-SDK attempts to stream the file mms://server1/song.wma. -The request included an accelerated duration of 10 s -The request included a bandwidth value of 700 kbits/s. -Note RTT is assumed to be zero in this example. -Note the file is already opened, and headers have already been obtained.
00.10	-Data transmission begins at 700 kbits/s. -The delta between the request and the start of transmission is purely an estimate. (Actual values depend on a variety of factors).
00.50	-Client buffer contains 5 seconds worth of data. -Client rendering begins.
00.90	-The server has transmitted 10 seconds worth of data. -The client has rendered 0.4 seconds worth of data. -The client buffer contains 9.6 seconds worth of data.

Since the client buffer now has considerable time to recover from a short outage (3-6 seconds), the end user may not experience any rebuffering. Much of this assumption depends on the ability of the network to fully recover following the outage and allow additional data through the connection before the client buffer is exhausted.

b.) Example where intelligent buffering does not affect startup latency:

00.00	-SDK attempts to stream the file mms://server1/song.wma. -The request included an accelerated duration of 10 s -The request included a bandwidth value of 56 kbits/s. -Note RTT is assumed to be zero in this example. -Note the file is already opened, and headers have already been obtained.
00.10	-Data transmission begins at 56 kbits/s. -The delta between the request and the start of transmission is purely an estimate. (Actual values depend on a variety of factors).
05.10	-Client buffer contains 5 seconds worth of data. -Client rendering begins.
10.10	-The server has transmitted 10 seconds worth of data. -The client has rendered 5.0 seconds worth of data. -The client buffer contains 5.0 seconds worth of data.

7.1.2 Broadcast content

The broadcast scenario has many similarities to the on-demand scenario. The main difference is the adverse impact to end-to-end latency if content is buffered on the server for accelerated transmission. The server must buffer data in order to send it at an accelerated rate. This results in delaying the rendering of data further from when it actually transpired.

To understand this issue better, imagine a broadcast publishing point being fed from an encoder capturing a video feed. In addition to all the other latencies caused by the Windows Media platform, buffering on the server for the purpose of accelerated streaming will only make end-to-end latency worse.

Different customers have different requirements. Some customers are very concerned about overall end-to-end latency. For example, customers that desire near real-time feedback from a broadcast event through a mechanism such as chat may find the incremental additional latency unacceptable. Another instance where end-to-end latencies are very undesirable is the case involving web simulcasts that customers wish to loosely synchronize to other broadcast mediums such as live television. In these limited cases, the customers shall reduce the amount of buffering that occurs on a publishing point to a minimum at the cost of startup latency.

A majority of broadcast events today are not concerned with a reasonably small (5-10 s) incremental end-to-end latency. The benefits offered by greatly reduced startup latencies on high bandwidth networks are worth the costs to many customers such as MSNBC and most Internet radio stations. Therefore, the default behavior of the server shall include buffering data at the broadcast publishing point to enable accelerated transmission.

It should be noted that users who connect over network links that closely match the bitrate of the content will not receive any benefit from accelerated streaming. Broadcast publishing point buffering will actually cause harm in this case, since end-to-end latency is increased and no corresponding benefits are achieved.

7.2 Improved ability to withstand occasional brownouts

Intelligent buffering allows clients to maintain a larger buffer without increasing the startup latency. Since a larger buffer is available, clients can theoretically withstand larger network outages with little or no perceptible impact to the user. For instance, imagine a client that suffers from a 3 second network outage. When the network resumes, nearly 7 seconds of content will still be present in the client buffer. The client will continue to submit UDP resend requests to the server until the specific packet is actually needed by the rendering engine. Assuming the server maintains the packet for resending, and the client is able to submit the request after the network has recovered, intelligent buffering greatly increases the chances the recovery of more packets.

8.1 Configurable Settings

8.1.1 Server Options

8.1.1.1 UDP Resend Data

The amount of data kept for the purposes of satisfying UDP resend requests shall be configurable with the following namespace keys. (Note this behavior has already been implemented).

"Other\\PacketPump\\MinResendBufferSizeInMSecs"
"Other\\PacketPump\\MaxResendBufferSizeInMSecs"

The default value for both keys shall be 8,000.

8.1.1.2 Broadcast Publishing Point Buffer Duration

The amount of data buffered on a broadcast publishing point shall be configurable with the following namespace key:

"Other\\FanOut\\BCastBufferQueueSize"

8.1.1.3 Server-level Accelerated Streaming Throttle Mechanism

The server uses an aggregate bandwidth limit to throttle fast startup requests. For example, a nearly simultaneous surge of 100 fast startup clients requesting a low-b/w stream would theoretically saturate the Fast Ethernet NIC and adversely affect all streaming clients if a throttling mechanism isn't in place. To make matters worse, if the administrator employs limits to "protect" the system from becoming saturated, the limits will quickly be hit (in the bursty scenario previously mentioned) and new clients will be denied the ability to connect. While it is prudent to deny new connections if absolutely necessary to protect the existing streaming clients, it is also desirable to increase the likelihood a request will succeed when the server is under load.

The following namespace value sets a limit that disables fast startup requests once the aggregate server output reaches this value. The default value is 30 mbits/s.

"Server\\Limits\\AcceleratedStreamingAggregateLimit"

8.1.1.4 Maximum Acceleration Rate Setting

The maximum transmission rate of accelerated streaming shall be configurable in a server namespace value. Client requests that exceed this rate shall be serviced with the maximum setting on the server. This value is in kbits/s. The default value is 1,024 kbits/s. Setting this value to 0 essentially disables the accelerated buffering feature in the server for all content bitrates.

"PublishingPoints*PubPointName*\\AcceleratedStreamingBandwidth"

8.1.2 WMF SDK Options

The SDK shall support the ability to perform the following options.

8.1.2.1 Rate of acceleration.

The rate of acceleration is determined by the LinkBw function. The linkbw function determines the value through the use of TAPI, packet pair measurements, or explicit values set through the SDK interface listed below:

IWMReaderNetworkConfig::SetConnectionBandwidth (DWORD)

Setting this value to 0 causes the networking code to automatically detect the bandwidth through packet pair and TAPI algorithms.

8.1.2.2 Percentage of LinkBw measurement to use for accelerated buffering

The SDK namespace value is used to degrade the amount of bandwidth utilized for accelerated buffering in order to prevent link saturation and also to allow some overhead for UDP resends, network jitter, etc.

Location:	**NetworkSource\\Object Store\\WMS Network Source\\Properties**
Value Name:	**PercentBWUsageForAccelStreaming**
Default value:	**85**

8.1.3 Client Options

8.1.3.1 Configuration of the acceleration duration.

The user can currently set the minimum buffers present before rendering can begin through the player UI. The client networking code shall request twice the amount of content needed for rendering to be accelerated. For instance, if the client buffer size is set to 5 seconds, the client shall request the first 10 seconds to be accelerated.

9 Eclipse Deliverables

9.1 WMSDK Support

The Zeus WMF SDK will contain the implementation necessary to meet the intelligent buffering logic specifications.

9.2 Server Support

The Hercules server will contain the implementation necessary to meet the intelligent buffering logic specifications.

9.3 Encoder

The encoder will not be affected by the implementation of this new feature. No specific changes shall be made to the encoder code for this feature.

9.4 Client

The Eclipse client code will utilize the intelligent buffering logic contained in the Zeus SDK.

10 COMPATIBILITY

This feature shall be supported and tested using combinations of the following components in various streaming and distribution scenarios:

-v7 encoder and later versions
-v4.1 server and later versions
-v6.4 clients and later versions

11 POSTPONED ISSUES

11.1 Dealing with the 1 mbit/s b/w detection limit.

The Artemis and Zeus clients currently limit the fast startup request to 1 mbit/s. If the content bitrate exceeds 1 mbit/s and the link_bw reports 1 mbit/s, the client will select the highest b/w stream available, but it will not employ fast startup. Also, on networks where sufficient b/w exists, fast startup will only be employed in a limited manner for higher bandwidth content less than 1 mbits/s. For example, a 700 kbit/s stream on an enterprise network will experience minimal benefits from fast startup. Siddeb is to investigate possibly increasing this 1 mbit/s restriction on the client and propose a solution. Discussion regarding potential solutions included raising the 1 mbit/s limit to a higher value. Other proposals included overloading the meaning of wm_bitrate.

12 OPEN ISSUES

13 DD HISTORY

10/6/00 Troyba Document Created.