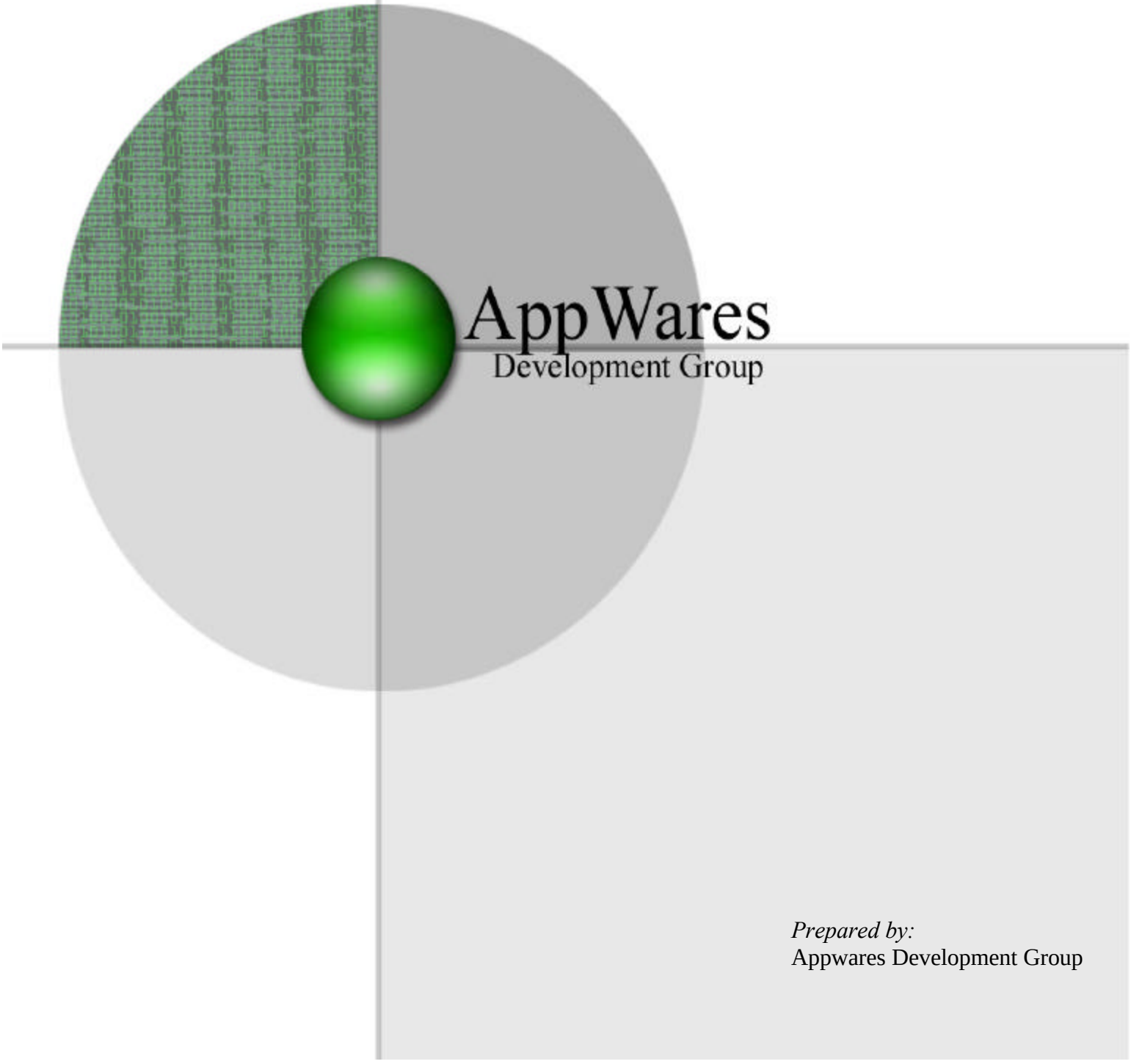
Technical White Paper:

Integrated Media Portal Back-End
Services and Messaging



Prepared by:
Appwares Development Group

Technical White Paper:
Integrated Media Portal Back-End Services and Messaging

Integrated Media Portal Back-End Services and Messaging

This document describes how an AppWares Integrated Media Portal (IMP) retrieves content and communicates with the back-end system to facilitate dynamic content delivery, broadcasting, messaging and website interaction. It serves as an overview and includes detailed descriptions of the capabilities of the AppWares server software as well as the IMP. Comments on scalability as well as security are included in the respective sections.

IMP Content Retrieval

The IMP typically displays content in an embedded browser window. The content is therefore any digital media file (or collection of files) that can be shown in a standalone or plugin-enhanced browser. The IMP facilitates the easy use and implementation of three methods for content delivery:

- local content storage
- direct web retrieval of content
- cached web retrieval of content

Any content can also be parsed or modified before it is being used by the IMP or shown in the embedded browser allowing for maximum flexibility. Should content require a specific browser capability provided by a plug-in, the plugin can either be automatically installed by the IMP executable installer or retrieved through the internet via the standard browser plug-in interface.

Typically content retrieval and display is initiated through a local event (i.e. mouse click), a network event (i.e. a broadcast), a network message (i.e. as a response from another IMP) or as part of the build-in functionality of the IMP (i.e. dynamic content).

All content is retrieved using standard HTTP or HTTPS connections. This is currently the most compatible method for most end users behind broadband routers, firewalls and proxy-firewalls. The IMP uses the internet connection settings of the embedded browser for its connections transparently - i.e. if the end-user can surf the web, the IMP can too.

Local Content Storage

The embedded browser displays a locally stored media file (HTML, images, video, audio, etc.) which is included in the IMP Archive (.AWI). The information is either extracted from the archive once when it is first opened or can be created on the fly by the IMP. Security can be achieved by encrypting the file and requiring the IMP to decrypt content on the fly before it is viewed - this ties the media to the specific IMP (i.e. it becomes a viewer for the encrypted data). Scalability is not an issue for this case of content as each IMP has all information available at the time of view.

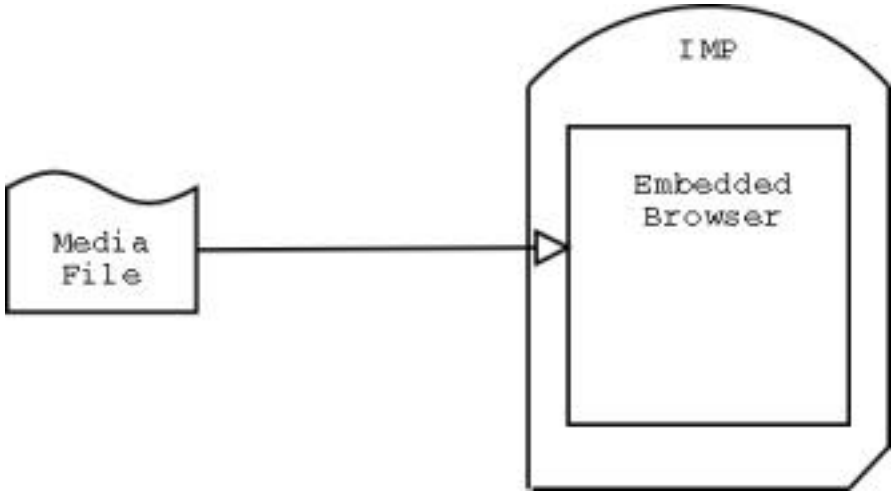


Fig. 1

Shown in the diagram of Fig. 1 is the *Embedded Browser* displaying or the *IMP* loading a local *Media File*. Each IMP has a designated storage path for content so that multiple IMPs don't overwrite each others files.

Direct Web Retrieval

The embedded browser displays a webpage or streaming media file retrieved through the internet. Similarly the IMP can retrieve content the same way, but as a background operation. This is the most common method an end-users makes use of internet resources, i.e. when the user is surfing the web. It allows for versatile content creation and the most complete media format support. This mode of retrieval relies on the browser to load and the show the content and would work if the end-user has an internet connection. It offers data security through SSL encrypted connections as well as the possibility to embedded streaming media content such as live-audio or live-video feeds. Scalability can be addressed similar to any high volume website: a virtual server setup with a farm of physical webservers pushing mostly static content can easily service the needs of million of clients.

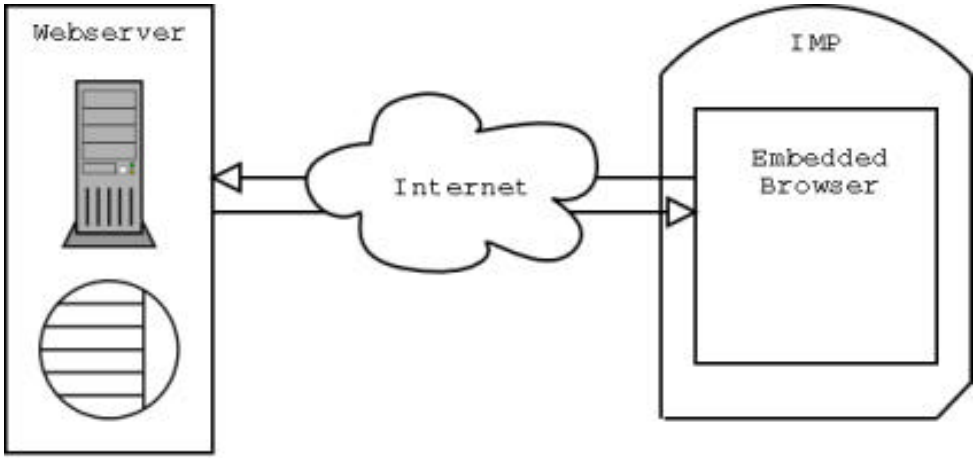


Fig. 2

Shown in the diagram of Fig. 2 is a *Webserver* serving files to the *Embedded Browser* or the *IMP* through the *Internet*. An IMP can retrieve information from any number of different servers The webserver could consist of virtualized farm of servers to satisfy a high volume of traffic. Specialized media servers can also be used for streaming formats and content.

Cached Web Retrieval

The combination of local content storage and web retrieval can be used to cache files and make them available once they are completely downloaded to the end-users harddisk or for times when the end-user is offline and the webserver cannot be reached. The IMP offers background downloads using the available internet bandwidth to avoid any perceived delays for the viewing of large media files. Large files can also be served in segments that are downloaded separately, checked against a checksum and then combined by the IMP before shown in the embedded browser. In this mode, files can be archived, compressed as well as encrypted for increased transfer efficiency and security.

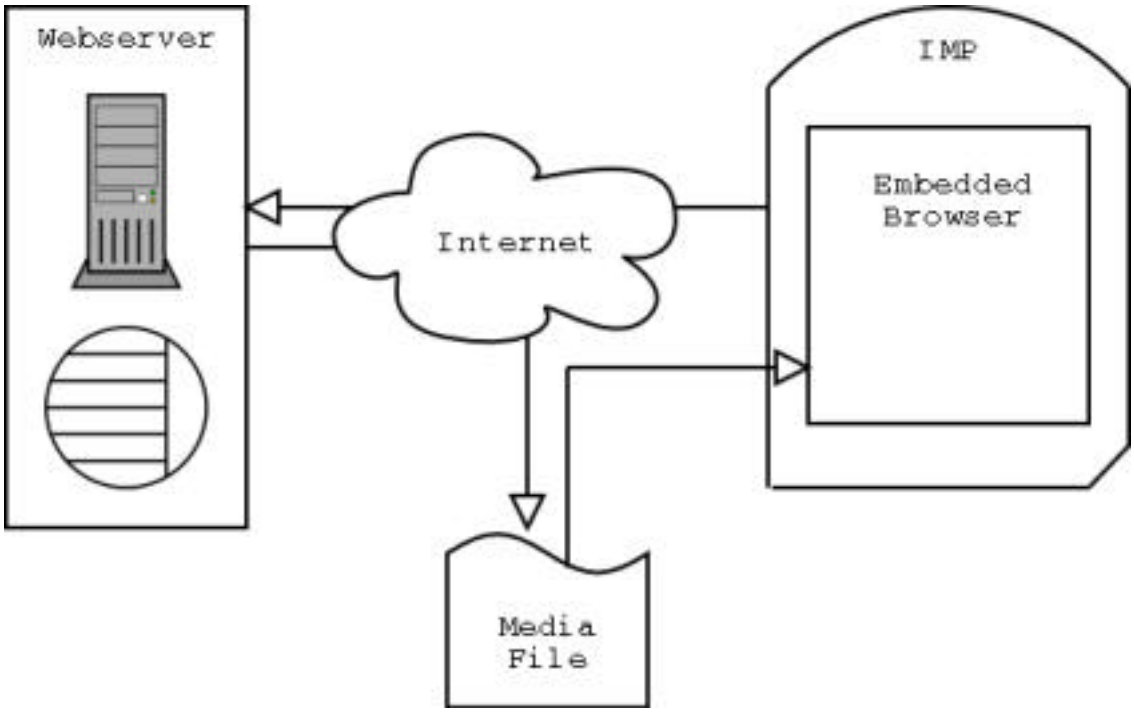


Fig. 3

Shown in the diagram of Fig. 3 is the flow of information for the two step cached web retrieval. The *IMP* requests one or more files through the *Internet* from the *Webserver* and stores it locally creating a *Media File* for display in the *Embedded Browser*. At this stage special processing such as unarchiving and decryption might take place. The download is an invisible background operation, as are the file processing steps. The content is only shown when the complete file or collection of files have been retrieved.

Event Generation
The content display or retrieval is usually triggered by events and the controlled by the code logic of the IMPs program. These events are either generated in the IMPs user interface and the IMPs program, or they are generated in external sources such as a broadcast system or a multicast UDP message to be received and processed by a networked IMP.

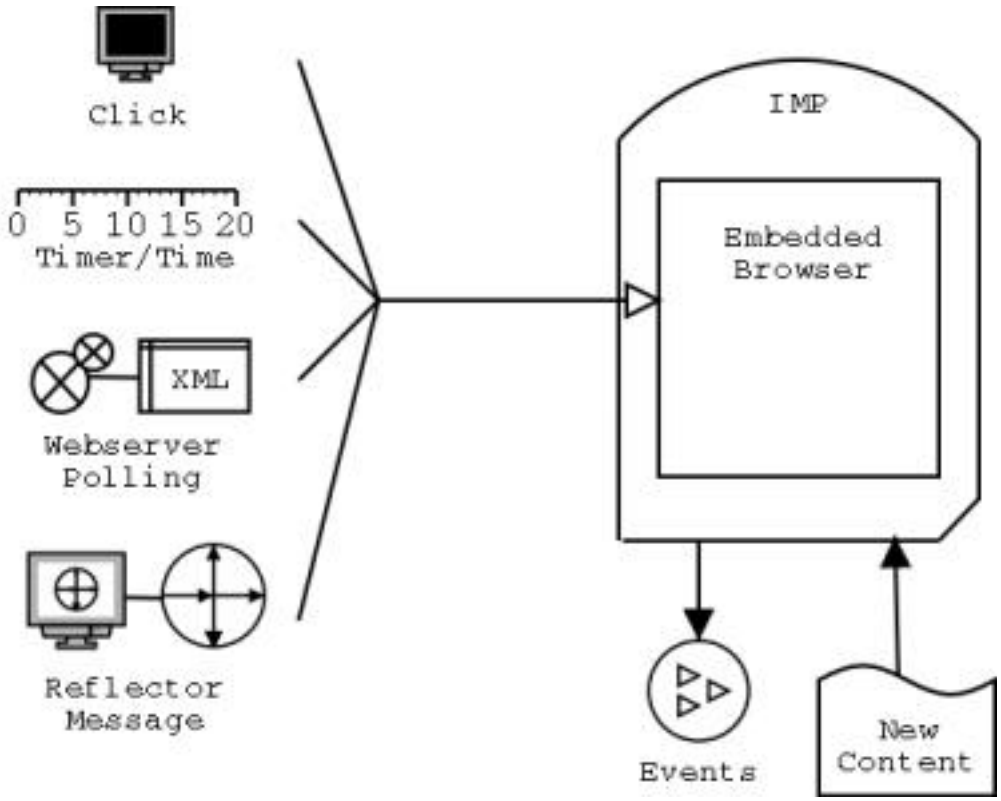


Fig. 4

In the the diagram of Fig. 4 the items on the right list the four different types of sources that can cause the IMP generate an event to fetch, process and display new content using any of the methods described in the preceding section.

- **Click (GUI)**: Simplest of all, the IMP will respond to the user-interface like a mouse-*click* on a button to provide access to content. These events are typically part of the IMPs programming logic and are set when the user interface is designed. (Example: help page, pre-defined schedule, contact page.)
- **Timer/Time**: A *timer* could retrieve and show information in regular intervals. Similarly the IMP could get information at a specific *time* of the day or day of the year. Again, these events are highly specific to the function of an IMP and have to be coded accordingly when the IMP is designed. (Example: WeatherNetwork's current temperature display retrieved from the website, stockmarket ticker based on regularly retrieved data.)

- **Webserver Polling**: A w*ebserver polling* can be used to retrieve centrally generated and stored events. The IMP back-end software manages a database of possible events, serializes them to they are received in order of creation as well as priority and returns an XML description file to the IMP for processing. It is the IMPs programs responsibility to parse and handle the XML content and take appropriate action to process and show the content. (Example: media messages to the desktop notifier).
- **Reflector Message:** With a background utility task, the *reflector*, the IMP can also receive *events* from other IMPs, any webpage that is displayed on the end-users web browser or by special UDP data packets received on the reflectors input port. (Example: interactive website where the IMP responds to website content.)

Webserver Polling Interface

The webserver polling interface is the standard way for centralized event generation and communication with the IMP. It is not the most efficient way of messaging as compared to more specialized proprietary solutions, but it always works as long as the end-user can access the internet through a browser, it is easy to use by the IMP developer and simple to deploy as a hosted or in-house backend service.

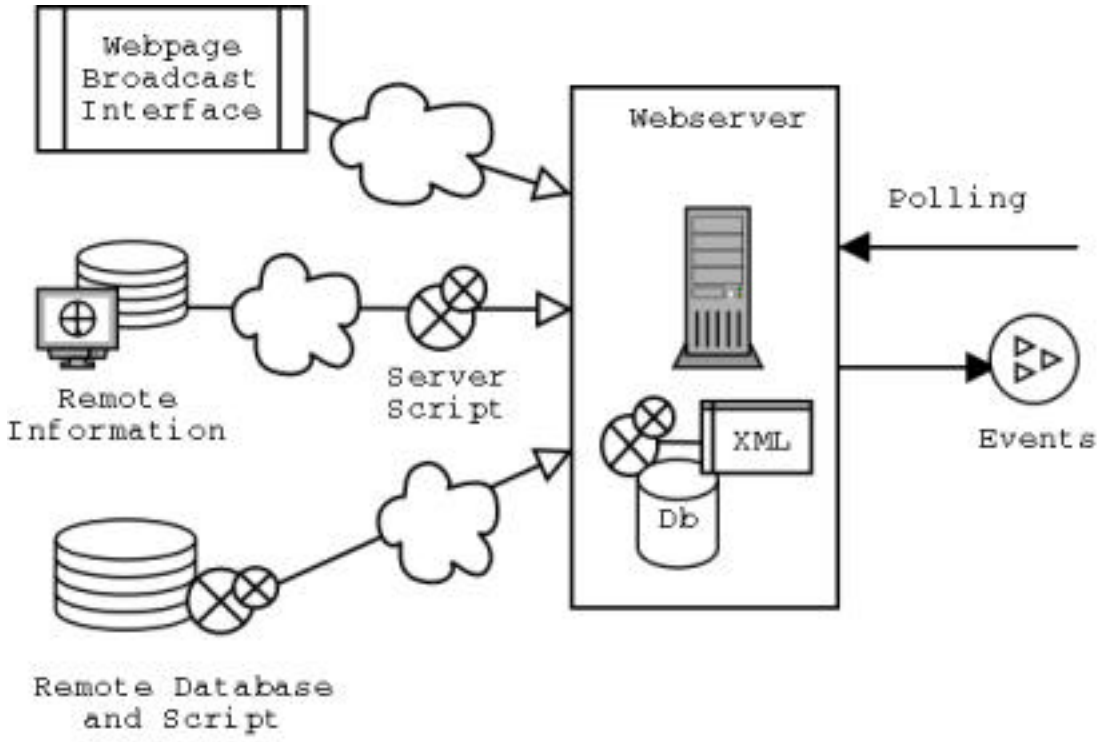


Fig. 5

The diagram in Fig. 5 outlines the flow information and the possible input sources for the events database.

The IMP will contact the messaging server in regular intervals controlled by the IMPs program logic. Since this is a standard web request, all normal setup and security methods for webpage retrieval can be used (proxys, SSL encryption). The webserver

maintains a database of events for the IMP based on the IMPs identity (product identity as well as user identity) and returns a set of event instructions as an XML file. Serialization of the events based on time or priority as well as event selection based on product, channel or user are done efficiently within the database of the event server.

The event database can be updated through a variety of methods:

- A webpage broadcast interface uses standard CGI techniques to generate content and insert events into the database, implementing an easy to use content creation and broadcasting system. (Example: MyApp broadcast section.)
- Specialized server scripts with direct access to the webserver and it database can contact remote information sources and databases, process the results that are received and generate events automatically. (Example: RDF newsfeed alerts.)
- Through a web-based interface, remote databases, programs and automated scripts can insert information into the event database.

Message Reflector

The message reflector is a subcomponent of the IMP and used for specific messaging tasks.

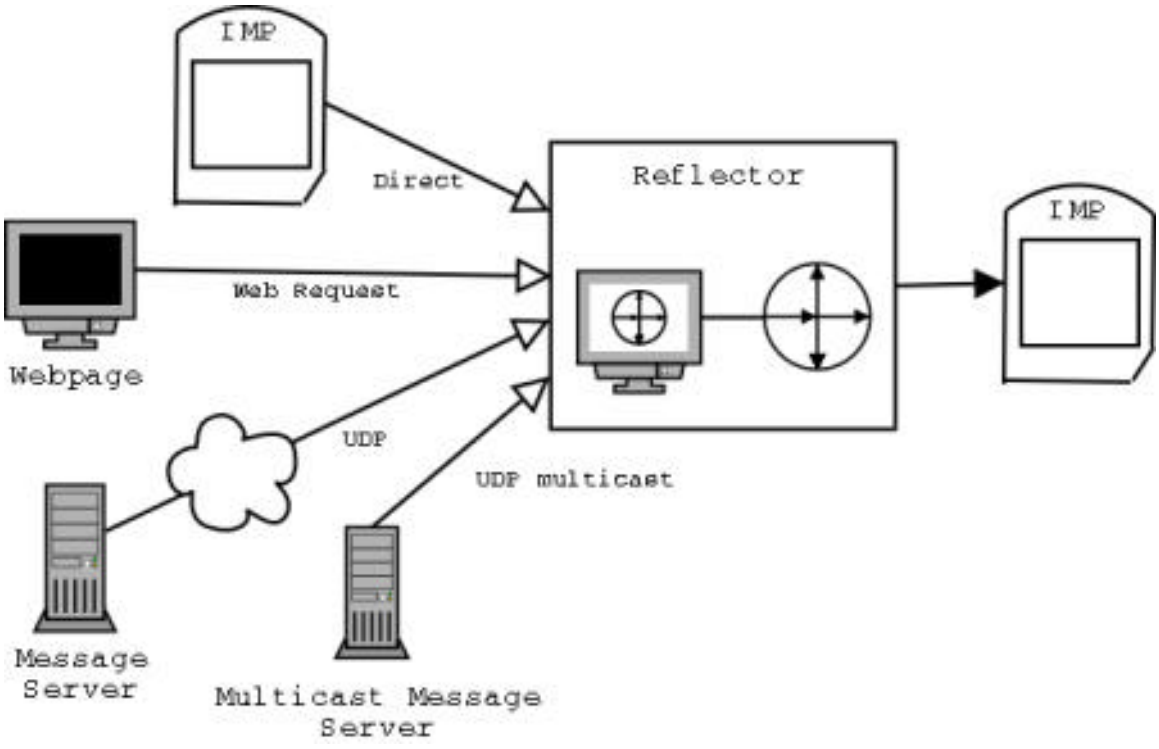


Fig. 6

When an IMP is running, it starts and connects to a R*eflector* background service task. This small local messaging service is used to pass information to the IMP and

allows the IMP to respond to several types of inputs as shown in Fig. 6:

- **IMP Direct:** An *IMP* might send a d*irect* message which will get dispensed to all locally running IMP instances. The programming logic of the IMP can then respond and thereby allow IMPs to interact with each other. Example: interactive IMPs.
- **Webpage Request:** Any Webpage can be designed in such a way as to include html GET requests to the reflector. Typically this can be used to embed events and data using web bugs (i.e. 1 pixel sized invisible images) but can also be used to serve locally stored files and service form requests from the embedded browser. Each request can include one or more messages with associated data. Example: interactive web page.
- **Message Server:** A m*essage server* can be used to send compact UDP-formatted packets to the reflector for efficient data broadcasting to the IMP. This method requires that the server can be reached by the IMP, i.e. firewall settings might have to be adjusted to accommodate the transmissions in the reflectors data port. It also involves a connection setup for each reflector with the server to allow end-users behind broadband-routers a connection.
- **Multicast Message Server**: A m*ulticast message server* can be used on a LAN for efficiently broadcasting events on a UDP data port. This is the most bandwidth conserving, immediate and scalable method of sending information to IMPs, but its use is only possible between locally connected computers (i.e. on a corporate LAN). To expand the reach of a broadcast system, a message server can be chained to one or more central servers to allow multiple disconnected LANs to communicate or receive events. It is typically a custom application to serve the specific communication needs of the IMP application.

Summary

The IMPs networking capabilities and backend services implement a flexible system for content delivery that build on existing standards for maximum compatibility with any existing networking infrastructure. While it it was designed for extreme easy-of-use for an IMP developer, it does offer many powerful solutions to integrate and broadcast content that make it compelling for the user. It also includes several advanced features such as UDP messaging and encryption to allow for scalable solutions that satisfy any need.
