Version: 3/21/2001 4:45 AM

FARGOS/MultiManagement
Planning Guide

NOTE: The information contained within this document refers to a product under development and contains forward-looking statements. Some Application Programming Interfaces may be altered in response to experiences and suggestions obtained in the field.

FARGOS/MultiManagement: Planning Guide

FARGOS Development, LLC
757 Delano Road
Yorktown Heights, NY 10598
http://www.fargos.net
mailto:support@fargos.net

Notice of Rights

Trademarks

Abbreviations

FARGOS Development, LLC is a Limited Liability Company registered with the State of New York. It is required to identify itself as such in its name, hence the ", LLC" suffix. For purposes of readability in this document, the ", LLC" suffix is sometimes dropped. The phrase "FARGOS Development" always denotes "FARGOS Development, LLC" and is not intended to suggest any alternate form of organization.

Notice of Liability

Contents

About this document

This document is composed of several sections, each of which goes into increasing levels of technical detail. The first section provides a high-level overview of the components and functionality provided by **FARGOS/MultiManagement**. The second section illustrates the use of FARGOS/MultiManagement with the hypothetical scenario of a firm that does web site development, an activity with which many readers may have at least a peripheral understanding. This is followed by a fourth section, which describes in further detail the interrelationships between various FARGOS/MultiManagement components. A final section provides another hypothetical illustration as to how the system could be customized to implement distributed cataloging, royalty accounting, and on-line delivery of multimedia properties (such as digitized videos).

An Overview of FARGOS/MultiManagement

FARGOS/MultiManagement is an integrated suite of tools that manage the storage, retrieval and distribution of digital assets. Examples of such assets include conventional documents (the simplest case), and multimedia files (e.g., images, audio and video clips). The use of the word "asset" is intentional because it also carries with it the semantics that the item in question is perceived as important and its continued retention is useful, perhaps critical, to the organization.

Supporting Multimedia Assets

Conventional document management systems, so-called Digital Libraries, tend to crash when faced with the problem of dealing with multimedia assets. While exceptions exist, multimedia assets tend to be of significant size. Digitized films are an excellent example of such large assets. Because their physical size can easily exceed the available memory on a host, document and image management systems that were designed to work on only small portions of a file at a given time will crash when attempting to deal with such files.

There are other characteristics of multimedia assets that need to be taken into account by a suitable management system. Multimedia assets often are used in conjunction with other distinct assets. A simple example that illustrates some of the issues involved is an HTML page: most HTML documents also require a set of affiliated images (GIF, JPEG, etc.) in order to present the desired content. Another issue is related to performance: correct playback of audio requires that the data be available before it is required: any delays introduced while waiting for data destroys the listening experience and usually renders the audio incomprehensible. While video is less sensitive to delays than audio, it's worth noting that most video presentations also include an associated audio track. Video data requires orders of magnitude more bandwidth than audio. Consequently, the streamed delivery of audio/visual data benefits significantly from any actions taken to improve the offered quality-of-service. For example, scheduled use of available bandwidth can help avoid over-commitment of constrained resources.

In today's environment, and increasingly so in the future, rather than create a CD-ROM master, authors of multimedia content are distributing/publishing works directly on the public Internet or within their private enterprise. As a result, integration of the entire workflow process, from creation, editing and revision through publication, greatly simplifies the workload of content developers and administrators. While this is beneficial for anyone, it is all the more important for organizations that develop and maintain content in an environment that is sensitive to the timeliness of published information (e.g., news organizations).

Another benefit to automating the publication process is to enable publication on a per-user basis. Many multimedia developers have created collections of interesting content but they have no way to generate revenue from such libraries and keep track of associated royalties. In many cases, content providers feel that they need to personalize each copy of a digital asset they sell. This is done through encryption of the asset or application of a digital watermark where feasible (e.g., in the case of images). If a self-serve e-commerce system is to be deployed, the application of such transformations to the original asset needs to be an automatic process.

Deployment Model

The deployment model for a FARGOS/MultiManagement installation is broken down into several logical blocks:

- Asset Repository
- Developer workstation
- Publication point

Figure 1 (on page 4) illustrates the relationship between these logical blocks. In practice, deployed systems will typically have more than one instance of each logical component and they may be under the control of distinct administrative domains.

Asset Repository

An asset repository is the location where a given asset is permanently stored. The repository is responsible for maintaining backups of each asset, keeping track of who has accessed an asset, preventing simultaneous updating by multiple developers, enforcing access control and licensing restrictions, etc.

Developer Workstation

Each developer workstation runs appropriate content editing tools and these are outside the scope of FARGOS/MultiManagement[1]. FARGOS/MultiManagement provides each developer workstation with an interface that allows retrieval of assets from asset repositories and the subsequent transfer to the developer's local workstation. When new content is prepared, the inverse is also performed: content is sent from the developer's workstation back to an asset repository. Modifications to an item on a developer workstation that was registered with the asset repository can be automatically detected and trigger the process that saves the updated content in the asset repository.

Publication Point

Assets ready for publication can be transferred from the repository to a publication point. This transfer can take place either as an immediate result of permitting an asset to be published or in response to a request received from an outside source. The former (push) is useful when publishing a small number of assets intended to be viewed by many users and is easily illustrated by examples such as an HTML page and associated imagery or the digitized video of the advertising preview (trailer) for a new movie. The latter (pull) is more appropriate when making available a large collection of potential assets, each of which may be viewed by a small, perhaps non-existent, population of users (e.g., video-on-demand services against an archive of digitized films or television shows).

The process of transferring an asset often involves more than just simple transmission of the asset. Instead, publication of an asset frequently involves a series of transformations, such as format conversion, downsampling, customized digital watermarking, encapsulation using an encrypted enveloping technology, etc.

A consumer of content (i.e., an end-user) is logically viewed as a publication point. In contrast to a developer, consumers do not have access to the native asset, much less permission to alter its contents.

[1] Examples of content editing tools include Adobe Photoshop, Microsoft FrontPage, Apple Quicktime, Macromedia Shockwave and Flash along with thousands of other development tools.

Figure 1

Summary of Component Functionality

At a minimum, each site that deploys FARGOS/MultiManagement needs an asset repository and a developer's workstation. In theory, some development organizations may not need to publish prepared assets, so they might not utilize any publication points.

An asset repository's most significant feature is lots of high-performance storage space. Multiple servers can be utilized in conjunction to create a scalable cluster and commodity hardware will do well for such purposes, but the disk I/O subsystems need to be significantly more sophisticated than a pair of IDE disk drives. FARGOS/MultiManagement has no direct control over the hardware, but its high-performance multithreaded architecture will engage in more than one activity simultaneously and a disk I/O sub-system that can only perform one operation at a time (e.g., like an IDE drive) will severely compromise the performance of the system.

Functions of an Asset Repository

The primary function of an asset repository is to catalog and archive assets, while providing a secure interface that controls and records access to the repository's contents. This simple, albeit accurate, description belies that slew of benefits that are obtained as a result. Some of the beneficial capabilities are listed in Table 1.

Table 1

Function	Comments
Asset Preservation through Backup and Duplication	The repository performs a critical role in the asset preservation process by archiving files to alternative storage media for long-term/off-site storage (such a tape). Automatic duplication of content between redundant repositories also serves to protect assets.
Advisory locking	While a developer is editing an asset, other developers who attempt to access the same asset, either from the local repository or a copy, are advised that updates are pending.
Access control and tracking	A given asset may be as important and confidential to an organization as a financial

	statement. Protection against unauthorized retrieval or modification is a necessary capability of the asset repository. Logging access can assist with theft prevention and provide documentation as to the authorship of an asset.
Royalty accounting	Organizations that wish to make available access to their assets for a fee need a mechanism to keep track of who/which/and how many times an asset has been accessed. Automatic generation of a digital watermark can also be performed at the time an asset is retrieved.
Synchronization of Redundant Repositories	Assets may be duplicated in more than one repository for performance reasons. Updates to an asset are automatically propagated to such redundant repositories.
Generation of previews	The sizes of some multimedia assets are daunting and thus infeasible for browsing purposes. Automatic conversion-on-demand of an asset to a smaller, lower-bandwidth sample (e.g., thumb nails) can enable the previewing of an asset by developers/editors searching for interesting content. This can only be performed on content directly understood by the repository.
Inventory Control and Cataloging	Without appropriate cataloging, a repository of assets will be nothing more than a confusing collection of file names that represent the barest hint as to the enclosed content. While some file formats make provision for embedding a restricted amount of meta information within the file, it is invariably very limited. Many organizations have their own, distinct standards (e.g., the Associated Press for photographs), so the ability to map between multiple profiles of meta information is a needed capability.
Versioning	Given the critical nature of most of the assets maintained by the repository, the process of updating an existing asset does not normally involve the replacement of the original. Instead, the newly altered version is stored in the repository along with the original.

Distributed Capabilities

While asset repositories can be deployed as isolated clusters, additional capabilities are obtained when two or more repositories are permitted to interact. Developers are able to search and retrieve more content, locally created assets can be backed up off-site for further protection, commonly used assets available from remote repositories are maintained locally for improved performance, etc. FARGOS/MultiManagement does support different

administrative domains, so enabling the sharing of some content does not require that a site provide complete access to all of its assets.

Scaling

While the functions of a given asset repository can be realized by a single server, the inherent scaling problems can be circumvented by using a cluster of multiple servers. FARGOS/MultiManagement does not assume that a given server owns a unique copy of an asset. Rather than preclude their use, this enables the intelligent utilization of higher-end disk drive units that can accept more than one controller (e.g., twin-tailed SCSI) as well as networked file servers. Such a configuration is illustrated in Figure 2.

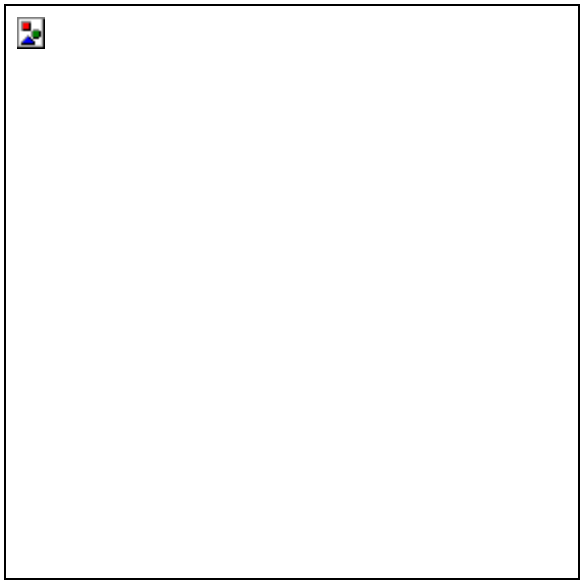

Figure 2

When content is duplicated from an asset repository to more than one host (i.e., another asset repository, developer workstation or publication point), FARGOS/MultiManagement uses multicast transmission whenever permitted by the underlying network infrastructure. In environments where multicast is not available[2], repositories cooperate to avoid sending a file over the same network link twice and perform their own application-level routing of asset transmissions. Because of the significant size of many multimedia assets, this provides benefits if an asset is being sent to interested populations with as few as two members. This functionality means that FARGOS/MultiManagement always transmits content in the optimal fashion and has scaling properties that remain unaffected by the number of users.[3]

Functions of a Developer Workstation

Developer workstations run the specific tools needed to create and manipulate content. Such tools deal with tasks from the mundane, such as text editors, to image manipulation, web site creation, audio and video acquisition and assembly. All such tools are outside of the scope of FARGOS/MultiManagement. Instead, developer workstations are provided with

[2] Many wide-area network (WAN) environments suffer from lack of multicast support.
[3] For the theoretically inclined: traditional unicast algorithms have scaling properties of O(n), thus the amount of data transmitted goes up linearly based on the number of users. The use of multicast transmission keeps the scaling properties of FARGOS/MultiManagement at O(1), or constant.

facilities that allow searching and retrieval of existing assets from the asset repository. Once a particular asset has been selected, the developer can request that it, and any additional assets upon which it depends, be placed on his local workstation.

Retrieval of an Asset

Before a copy of an asset is placed on the workstation's local disk, several checks are performed. If another developer is working on the asset in question, the requestor will be informed of this and given the option to not proceed with the retrieval of a copy of the asset. In a similar fashion, if retrieval of the asset requires payment of a royalty, the requestor will be informed of the cost that will be charged to his account and given the option of not proceeding with the operation. A set of custom programs can also be executed before and after the transfer takes place. These steps provide the opportunity to verify a complex set of prerequisites, reserve disk space and ensure the appropriate level of the editing tool is available, etc.

Retrieval of an asset can be requested to take place as quickly as possible or deferred to when it is convenient. Deferred retrievals allow the use of multicast transmission to transmit copies of an asset in which several developers have expressed an interest.

Updating an Asset

When the developer has finished modifying an asset, he uses the FARGOS/MultiManagement user interface to inform the system that he is finished with his modifications. He can request that his current copy be immediately stored in the repository or the action can be deferred. If the system has determined that additional parties besides the repository are interested in the asset, multicast transmission will be used for the transfer. In environments where a pool of developers are cooperating on a project, this enables a given developer's updates to be transmitted to the repository and his peers in a single transmission; the other alternative would require transmitting the asset at least twice: once to the repository and then back again from the repository to the peers.

Depending on the policy of the local site, a work-in-progress can be stored in the repository for safety. Because the repository supports versioning, this is still a safe action. The local site's policy can also determine if an asset is accessible by others after a developer has replaced it in the repository or if it needs to be reviewed and approved by other individuals (e.g., a manager) first. In such a case, other developers/users seeking the asset in question will be provided the most recently approved version even though an updated, albeit unapproved, version exists; however, the advisory locking facility will also inform them of the existence of a not-yet-released version.

A developer does not have to explicitly inform the system that an asset has been modified; retrieved assets are periodically checked for modifications. When modifications are detected, the preservation of a snapshot will be scheduled.

Functions of a Publication Point

In the FARGOS/MultiManagement model, a publication point is a logical destination to where sets of assets are delivered in a structured fashion. Once at a publication point, the resulting assets are expected to be accessed by the end-user using some application-specific means. This might be as simple as a file copy operation from the local file system. Typically, it will be something more interesting, like access via an FTP or HTTP server. The process of publishing an asset can either be a scheduled operation or initiated on demand.

A web site is a common example of a possible publication point: individual files, representing HTML documents and imagery, are utilized together to create a collection of interrelated web

pages. In the simplest case, all of these files need to be copied from the asset repository to the host upon which the HTTP service runs. For scaling purposes, many web sites utilize more than one physical server front-ended by some load-distribution technology[4]. In these scenarios, the server farm can be viewed as a collection of publication points.

Publication of Assets

Theoretically, it is possible to have a publication point retrieve all of the necessary data on demand at the time it is requested by end users and there is good deal of elegance to be found in this approach. For some sites, duplicating all of their potentially available content onto every server is problematic. Examples include sites that either retain all of their historical content (e.g., news sites) or have numerous assets with non-trivial sizes (e.g., digitized videos). Typical conventional deployment techniques are to treat such items as special cases and direct requests for such items to a subset (perhaps one) of servers that maintain the content in question. By treating such activities as special cases, these approaches fail to scale and are ineffective in environments, such as video-on-demand, where the normal case is random access to huge multimedia data files. Special case treatment of servers also creates single points of failure and yields fragile systems. A FARGOS/MultiManagement-based solution would perform publication of the asset at the time of the request.

The major downside to retrieval of assets on demand is performance. In many deployment situations, pre-staging every individual HTML page, image, etc., to each server works well and avoids introducing latencies that are part of the price of retrieval-on-demand. Also, in practice, certain technologies do not provide an opportunity to intercept incoming requests and retrieve the appropriate asset from the repository.

Consequently, a publication point can either pull assets from the repository as needed or an administrator can force assets to be pushed from the repository onto a publication point so that everything is prepared ahead of time. A publication point does not have to make exclusive use of either technique. A sample configuration of a publication point suitable for video-on-demand systems appears in Figure 3. This illustration shows how a web-based interface makes use of assets that are both pre-published and published on demand.

[4] Products from F5 Networks, IBM's e-Network Dispatcher, or even FARGOS Development's own patent-pending **FARGOS/SolidConnection™**, as well as numerous others, fit the bill.

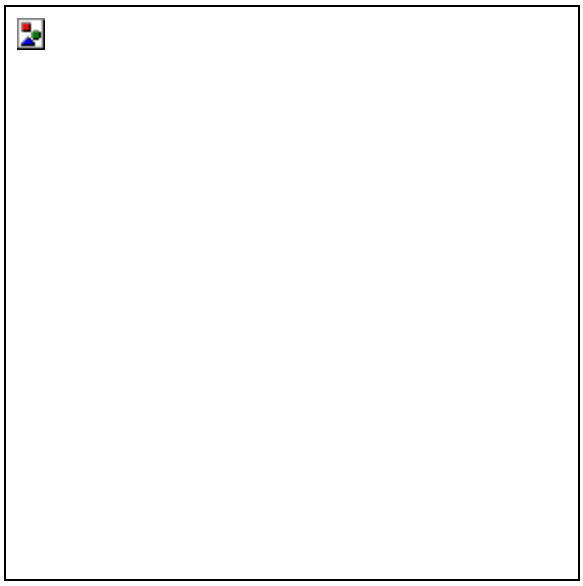

Figure 3

Deployment Scenario: Web Site Development

As an illustration of how FARGOS/MultiManagement supports common asset storage/retrieval/publication tasks, consider the activities involved with the on-going design and deployment of a large web site. Our hypothetical design firm has several developers that use content creation tools from vendors such as Adobe and Microsoft. They have a pre-existing collection of HTML templates and images that they have created over time.

When starting the design of a new site, a developer will search for and retrieve appropriate assets from the repository. As he creates brand-new assets, he registers them with the asset repository. Each night, the system takes a snapshot of the work on his development machine and stores copies of any updated assets in the repository. Each copy is treated as a new version, so work-in-progress does not destroy the original asset.

Colleagues who are interested in a particular asset (like a GIF image) also retrieve it from the repository and grant permission to have their local copy automatically updated. Whenever changes are made to the asset in question, each local copy is updated by the system without further intervention on the user's part.

When it is time to publish the web site, the appropriate set of publication instructions are created. With a typical web site, this would mean that a set of assets are copied from the asset repository to appropriate locations on the file system of the HTTP server. Once the appropriate instructions are defined, the process of propagating modifications to the site to the physical servers becomes automatic.

Deployment Scenario: Video-On-Demand

The complexities faced by a publication point can be partially illustrated by considering the example of a web site that makes available videos-on-demand from a large collection (several thousand) of digitized films. Each film is approximately a gigabyte, so the total collection represents over a terabyte of disk space, which is not practical to duplicate on every server that will be streaming content to viewers. Consequently, such assets are best published on demand.

After a user requests a film via a web browser interface, several processing steps are required in order to set up a streaming session. These include:

- selection and allocation of a streaming server from a pool of servers
- transfer of a copy of the film asset from the asset repository to the selected streaming server
- redirection of the user's browser to the correct port on the selected streaming server so that transmission of the stream can begin

Given the bandwidth required for video, a single server can easily be overwhelmed with work, so multiple servers are required to handle a non-trivial population of simultaneous users. Some obvious issues during the selection process are the load on a potential streaming server and whether or not the asset in question is already present on the server's local disk. If multiple users are interested in the same content, then significant benefits are gained by having them share the same copy of a digitized film.

Given the duration of a typical movie (e.g., approx. 2 hours), it adds to the user's experience if playback of the film can begin before the entire film asset is copied to the streaming server's local disk. This requires intelligent buffering that continues to copy data from the asset repository while the video streaming process is already underway.

A user's experience, however, will be severely impacted if the copying of the asset to the streaming server is slower than the streaming of data to the user's host. Such a scenario will

eventually lead to exhaustion of data on the streaming server, causing a pause in the playback. Thus over-commitment of resources will yield unacceptable results. It is worth noting that systems that stream audio or video data do not degrade gracefully: like the straw that broke the camel's back, once the system is overburdened with too much work, all (not just the newly added) active connections can expect to see unacceptable performance.

The following narrative illustrates how a video-on-demand system coupled with a FARGOS/MultiManagement asset repository would operate:

1. The user enters the URL for the web site into her browser. The browser connects to the HTTP daemon running on the front-end server dedicated to that purpose. The HTTP server hands back the top-level web page. Standard browser/HTTP server interactions subsequently take place to load the page's contents and associated images.
2. The user selects a page that contains a form that allows the entry of a query against the film catalog.
3. When the user presses the appropriate submit button, her browser issues a POST request to the site's HTTP server. The POST request is processed by the HTTP query application. It parses the request's contents and issues a corresponding query to the asset repository. The resulting list of available films is formatted into a dynamically generated HTML document and sent back to the user's browser.
4. The user selects a film for viewing from the resulting list. The URL of the displayed link actually refers to another application within the web server that will dynamically generate a HTML document that display details about the indicated film and provides a link that can be used to initiate a viewing.
5. Clicking on the link invokes another application within the web server whose function is to request the selection of an appropriate server within the streaming pool and initiate the transfer of the film to the selected streaming server. The server selection process takes into account the load on each streaming server and as well as what files are currently cached on each server. In addition, other administrative constraints, such as scheduled maintenance, are also processed by the allocation system: if the server is to be taken down in 30 minutes, starting a 2-hour streaming session on it is not appropriate.
6. Once a streaming server is selected, a new streaming session is initiated on that host. The controlling application for the session oversees the operation of several subordinate applications. One is the actual streaming application: this application owns the port to which the customer's MPEG-2[5] player will connect and might be a **FARGOS/SolidConnection**-enabled component[6]. The streaming application obtains its data from a FARGOS/MultiManagement application that is an intelligent buffer that owns the local copy of the film. All sessions that are streaming the film in question utilize the same instance of the buffer. This buffering application provides the functionality that allows a streaming session to begin before the entire contents of the film are resident on the front-end server's hard disks and allows physical memory to be used more effectively. The buffering application in turn is responsible for obtaining the video file from the asset repository. If the file was already present on the local file systems, then that particular piece of work is complete. In most cases, however, the buffering application will have to initiate a transfer of the film from the asset repository to the streaming server's local hard disks. This may require recovery of disk space from the local file systems by deleting the least recently used cached film. If the contents of a previously cached film are deleted, the server allocation process must be informed.

[5] MPEG-2 is used as a concrete illustration, but it could just as easily been another file format (e.g., Quicktime, RealVideo, etc.).

[6] **FARGOS/SolidConnection** is patent-pending technology for the fault-tolerant delivery of streams of data.

7. Once the various applications have been successfully initiated, the session controller sends back to the web server application an appropriate document that will trigger the customer's MPEG-2 streaming video player. As an example, this might be a Microsoft ASX file. This dynamically generated file contains the address/port of a new streaming session associated with the requested film.
8. After the HTTP server sends back the file, the user's web browser interprets the Content-type field in the HTTP header and starts the appropriate MPEG-2 viewer. The streaming video player connects to the indicated streaming server and starts receiving video data.
9. The streaming application processes the new connection attempt. It requests data from the buffering process associated with the video file and transmits packets across the network to the user's video player.

From the description above, it can be seen that interaction between a publication point and an asset repository is more than just a matter of making a few queries against a database and copying files and is actually best realized as a unified distributed system.

FARGOS/MultiManagement Capabilities

Asset Tracking

Every asset maintained by the repository is described by a logical database record. Each asset is assigned a unique Id[7] along with a human-readable name and its description record includes information that notes to whom the asset belongs. The database also keeps track of the physical location of each copy of an asset. Some of the record keeping data is illustrated by the relational database model of Figure 4:

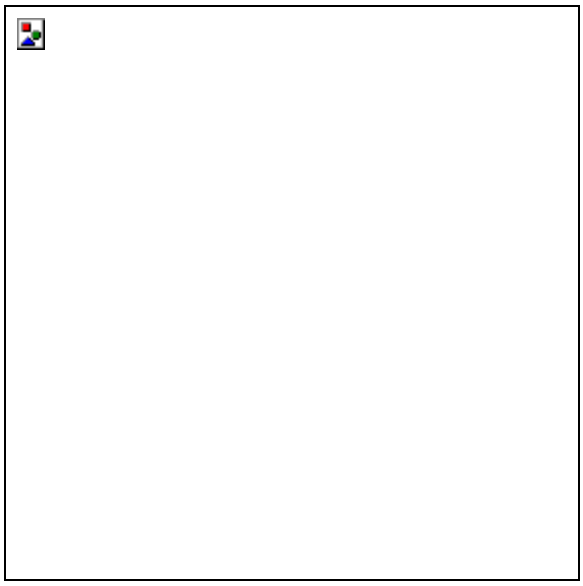

Figure 4

Many media-dependent attributes are obtained directly from a physical copy of the asset instead of being maintained in a database record. While retrieval of such information requires the implementation of file format-aware utilities, it ensures that the requested data is always correct and eliminates the need to rely on constant manual updating by the asset's author. Examples of such attributes include the file size (often available from the underlying file system), the height and width of an image, the playback time of a digitized video.

A given asset will often have copies in multiple physical locations and keeping track of each instance is one of the responsibilities of the asset management system. Such redundancies ensure protection against permanent loss as well as transient server outages while providing opportunity for increased scalability and reduced bandwidth utilization of inter-site links.

Asset Retrieval

An access control list is associated with each asset. The access control list holds information that determines who is allowed access to the asset, their privileges and associated charges. A given user may be permitted to modify the asset or might be restricted to only being able to retrieve it while simultaneously incurring a fee for use. With appropriate extensions, content creators can automatically charge end-users for the use of assets. To protect against inadvertent simultaneous modifications to an asset, advisory locks are maintained. Requests

[7] Such as a Document Object Identifier. See http://www.doi.org for further information.

for an asset also indicate whether or not the user intends to modify the asset or if they want to be notified if the asset is subsequently modified.

After a request for an asset is successfully processed, the asset's contents can be transferred immediately or scheduled for delivery to the destination at a more appropriate time.

Scheduled Transfer of Assets

While much easier to implement, immediate transfer of requested assets greatly reduces the opportunities for minimization of the utilization of the network. When the physical size of assets are large, such inefficiencies have significant impact. Through the use of intelligent scheduling, multicast transmissions can be used to efficiently duplicate an asset on a number of hosts. When used effectively, each time an asset is updated should result in a copy being transmitted only once over a given network link.

Publication of Assets

In the FARGOS/MultiManagement model, the publication of an asset is viewed as a process that creates a read-only copy and installs it at a desired location. The read-only copy does not need to be an exact duplicate of the asset; instead, it can often be useful to convert the asset to a different file format (e.g., GIF to JPEG) or personalize the copy by applying a digital watermark that identifies the intended recipient. FARGOS/MultiManagement provides for the execution of pre- and post-install checks that can be used to assert that needed resources (e.g., disk space) are available and that the asset was successfully installed. Assets can be grouped together through the specification of pre-requisites.