Data Universe™ Specifications

Brian McMillin

Abstract

This paper describes a distributed data exchange system for the Internet. The techniques allow for a totally decentralized exchange of arbitrary data among participating computer systems. Techniques for maintaining data integrity and anonymity in the presence of arbitrarily unreliable connections and storage media are discussed. All data structures required for a working implementation are described.

The distributed search mechanism described here represents a subset of a more generalized distributed computing capability. Potential extensions would use these data transport and directory search features to form the basis of a global supercomputing network.

Introduction

The Data Universe™ is a peer-to-peer file exchange system. Its implementation is totally decentralized and anonymous. Directory structures and bulk data are stored in anonymous, variable-length blocks on one or more host computer systems. File data is divided into multiple blocks, each of which may reside on one or more host computers. Distributed queries allow network-wide searches for files and their constituent blocks. All network transfers take the form of blocks pushed from host to host. Fault tolerance is inherent in the design in that no host or data link is required to be reliable. Redundancy of directory and data storage, distributed processing of search functions and autonomous movement and replication of all data provide network robustness.

All data within the Universe tends to replicate and "popular" data tends to replicate faster due to the action of Queries returning duplicates of that data. Once introduced into the Universe, it is difficult to remove or censor a piece of information. Queries are non-deterministic, so there is no guarantee that all copies of a particular data block could ever be found. The anonymous nature of data blocks themselves means that individual hosts never need be aware of the actual content of the Blocks in their repository. File Description blocks may be updated with new annotations, typically reflecting user's experiences with a particular file. These newly annotated Blocks are added to the Universe but do not supplant the original descriptions.

Data Blocks tend to migrate to multiple Hosts, each of which automatically make those blocks available to Queries and access by other Hosts. This connectionless data movement has several advantages over traditional file transfer methods. Data moving between computers with different speeds of physical media do not waste resources. Transfers to a fast computer may be coming from multiple slower computers. Transfers to a slow computer do not tie up connection slots on a fast machine. "Hot spots" in the network tend to be eliminated since Hosts that receive popular data Blocks automatically share them, thus acting as a distributed resource to relieve congestion on the first Hosts for the Block.

Security and Intellectual Property

It is recognized that the Data Universe will be used for the distribution of all types of data. This distribution and ready availability of data from anywhere in the world is one of the fundamental tenets of the entire Internet.

Certain jurisdictions and organizations attempt to exercise arbitrary dominion over what they consider proprietary, copyrighted or unlawful sequences of data bits. The Data Universe architecture is designed to protect storage and transport providers from arbitrary regulatory action. All data files in the Universe are separated into at least two parts: a File Description (FD) and the User Data (UD) Block(s). The Description and Data need not reside on the same host, but both are required to re-create the original data file. This insulates the Host computer and its administrator from any claims that it contained proscribed data.

A series of policies enforced on the Host can provide any desired degree of protection from such claims.

1) At the lowest level, the host may ignore all semantic issues and simply participate in the Universe.

2) A host may adopt a policy to never simultaneously have resident in its repository all of the Blocks required to re-create a particular file.

3) A host may choose to allow File Descriptions (FD) or User Data (UD) but not both in its repository.

4) When adding files to the Universe, a host may choose to break even small files (less than the maximum single Block length of 65,500 bytes) into multiple data Blocks.

5) When adding files to the Universe, a host may choose to encrypt the User Data (UD) Blocks. The key information required to decrypt the data would be stored independently within the Universe. This adds a third component (in addition to the FD and UDs) that must be present at a single machine to extract the original data.

6) When adding files to the Universe, a host may break the original file into multiple Blocks in Stripes instead of blocks of consecutive data.

Getting Started

When joining the Data Universe, the administrator of a particular host sets some simple policies relating to the resources that he wishes to contribute to the Universe. He chooses an amount of disk space, a TCP/IP socket, a CPU usage limit and inbound and outbound network bandwidth limits. From this point on, the operation of the Universe is autonomous. Files may be added to or extracted from the Universe via a user interface whose operation is essentially independent of the Universe itself.

The initial distribution of the Data Universe software contains a seed version of a Host List (HL) Block. The new Host adds itself to the list. In the absence of anything else to do, the Universe Idle Process periodically pushes its new version of a Host List to the other Hosts already on the list. Eventually, one of these transfers should find a live Host. This host will eventually push an updated Host List back. The new host has now joined the Data Universe.

An identical process occurs when a Host restarts, except that the Host List (HL) Block that it uses is the most recent one stored before shutdown.

Glossary

Block - A block is a variable-length sequence of bytes. Blocks may contain from 4 to 65,500 bytes. Blocks come in five types based on their content: User Data (UD), File Description (FD), Directory List (DL), Host List (HL), and Query (QU). The first 4 bytes are reserved for a signature. The maximum length is selected to ensure that a Block can always be transported in a single IP Frame.

Block ID - A Block ID is a sequence of from 22 to 32 printable characters. The Block ID is the printable representation of a LMD5 descriptor of the data contained within the block. The printable form uses 64 characters from the set ['0'..'9','a'..'z','A'..'Z','$','%'] to represent 6-bit values.

File ID - A File ID is similar to a Block ID, except that it applies to the entire body of a disk file, not just a single block. File IDs are used to consolidate different File Names and/or File Descriptions that describe the same content. File IDs are also used to ensure the integrity of multi-Block files.

Host Computer - A Host is a computer that participates in the Data Universe by running the Universe kernel application. A host also makes available resources that include CPU time, Storage space, a TCP/IP socket and a certain amount of Bandwidth to the Internet.

Host ID - A Host ID in the current implementation is an IP address and port number in printable ASCII text. The format of a Host ID is ddd.ddd.ddd.ddd:ppp. The standard dotted decimal notation is used.

LMD5 - A modification of the RFC1321 MD5 message digest algorithm in which the input data length in bytes is pre-pended onto the 128-bit message digest value. LMD5 is the algorithm used to create Block IDs and File IDs.

Query - A type of data Block that contains instructions for searching the Repository of one or more Hosts and returning the results. The Data Universe Idle Process scans the Repository looking for Query Blocks. As they are found, they are processed and disposed of either by (1) returning results, (2) forwarding to another Host, or (3) discarding. A list of recent Queries prevents the same Query from running more than once on a given Host.

Repository - The storage area on a Host computer that contains data Blocks. A configuration parameter allows the administrator of each Host to limit the size of the Repository.

Slicing - A general term that means breaking up an arbitrarily large data file into a set of one or more User Data (UD) Blocks. Typically, the first step is to run a compression algorithm. The results are then divided into Blocks that do not exceed 65,500 bytes. The size of the blocks and whether they contain consecutive data (or are broken into stripes) are arbitrary decisions made at the time the file is entered into the Universe.

Timestamp - standard printable ASCII form for time-of-day values used in File Descriptions, Directory Lists, Host Lists and Queries. The value contains exactly 12 decimal digits representing Universal Time in the format: yymmddhhnnss. Since the timestamp is predominately used for expiration times and sorting, simple string comparisons will suffice in most instances. Differs from a file's DateTime which is variable length and based on local time.

DateTime - standard printable ASCII form of the creation date and time of a file. Used to allow recreation of more complete directory entries for files extracted from the Universe. The value contains up to 14 decimal digits in the format: yyyymmddhhnnss. If the leading digit is a "2" it and any subsequent zeroes are suppressed making this a variable-length form which will use only 11 digits during this decade.

Implementation

The Data Universe is implemented in a compact, easy-to-distribute form.

`UniverseKernel.exe` implements the data storage interchange functions.
`Universe.ini` is the configuration file that specifies resource allocations for the Kernel.
`Universe.exe` is the user interface that allows files to be shared in the Universe and searches to find files.
`Repository\` is the directory that contains data Blocks stored as individual files with Block IDs as names.
`AddFiles\` is the directory which contains files to add to the Universe.
`ExtractedFiles\` is the directory which contains files retrieved from the Universe.

The Data Universe creates a directory which houses the Repository of data Blocks. These files are named with their Block IDs and use the standard operating system file system for disk management. Any necessary file or disk maintenance may be done with existing tools. None are provided with the Data Universe. Note that the implementation under Windows uses a hexadecimal version of BlockIDs for naming files, since upper- and lower-case is indistinguishable in the file system.

Directories are provided for files to be added to or extracted from the Universe. This provides isolation of files for security and anti-virus quarantine. The speed of add and extract operations is non-deterministic, so autonomous operation is expected.

The Data Universe may be removed from a Host by simply deleting all associated files.

Data Universe Configuration File

The operation of the Universe is controlled by a simple configuration file, **`Universe.ini`**. A sample configuration file is listed below:

```
[Data Universe]
HostID=192.168.2.10:1234
CPU=10
Interval=5
Inbound=500KB
Outbound=100KB
RAM=15MB
Disk=1000MB
Slice=65500
Stripe=No
HostFD=Yes
HostUD=Yes
```

Data Formats

All data transfer and storage in the Universe is based on the use of named, variable-length blocks of data.

All Data Transfers are in the form of a connectionless datagram sent from one host to another. The datagram contains the Block ID and the Block of data. The recipient verifies that the Block ID matches the Block and stores the Block in its local repository.

Each Block contains signature bytes on the beginning that identify which of the five basic types of Blocks it is.

UD	User Data	The body of files, usually compressed
FD	File Description	The File Name(s), text description(s) and list of UD blocks that are the data.
DL	Directory List	List of Block IDs in the repository of a particular host.
HL	Host List	List of Host Addresses and their anticipated longevity in the Universe
QU	Query	Block containing a text description of a search to be performed on the repository.

UD - User Data Block

User Data may be stored in the Universe in raw, uncompressed form. The data is broken into segments of up to 65,500 bytes. The signature characters "##UDRD" are the first six characters of each block, followed by the segment of data.

User Data may be compressed using a LZH algorithm prior to entry into the Universe. The resulting compressed data are broken into blocks of up to 65,500 bytes. The signature characters "##UDLZ" are the first six characters of each block, followed by the segment of compressed data. Other compression algorithms may be used in the future.

FD - File Description Block

A File Description (FD) Block associates a File ID with one or more file names, zero or more file description text strings, and a list of one or more User Data (UD) Blocks that contain the actual file data. The FD Block is formatted in an XML-like manner for ease of searching and parsing.

Some files may be broken into an extremely large number of UD Blocks. This may cause the list of Block IDs to exceed the capacity of a single FD Block. Multiple FD Blocks may be cross referenced by including an indirect reference to an UD Block in the list of UD Block IDs. An indirect reference is a Block ID with an "@" on the front. This UD block is interpreted as a sub-list of UD Block IDs to be inserted in the list.

Each File Name and File Description within the FD block has an associated timestamp. Typically, names and descriptive text are sorted into reverse chronological order within a FD, with older information falling off the end. Names and text descriptions that share the same time stamp may be consolidated into one sub-record.

A File Description block is a block of ASCII text formatted as follows:

```
##FD<ID=FileID>
<fn="autoexec.bat",td="Everyone should have this file",dt=30721095432,030721123456>
<td="Solves all your DOS problems",030721123457>
<td="Danger! Reformats hard disk",030722010015>
<bl=BlockID>
```

Disk Files are uniquely identified by their File IDs and File IDs are based only on the file content. This unique content is associated with one or more File Names (Windows, etc. File Names), zero or more Text Descriptions, and an ordered list of one or more Block IDs. The list of Block IDs allow the file to be reconstructed from scattered pieces. The integrity of the result is verified by comparing against the File ID.

As part of its normal operation, a Host will typically scan its repository for FD Blocks with duplicate File IDs. The Blocks with the most recent timestamps may be arbitrarily retained, or File Name and Text Descriptions may be merged to create new, more appropriate FD Blocks. Note that it is possible for the same data file to be entered into the Universe many times, possibly using different Slicing or Compression strategies. This means that the Block IDs in the Block list would not necessarily be the same.

DL - Directory List Block

A Directory List (DL) Block contains a Host ID, an expiration timestamp and a list of Block IDs that are available on the host. The Block ID list need not be exhaustive and is chosen in a more-or-less arbitrary manner by the host. Directory List Blocks are created periodically during the Data Universe Idle process and pushed to arbitrary Hosts.

In addition, Directory List Blocks may be created by the operation of Queries. A Query may specify that the return value be a Directory List indicating which of a set of Block IDs are present on the target system. This is normally used in anticipation of requesting those data Blocks from one of several Hosts.

```
##DL<ID=HostID,030721123456>
BlockID,BlockID
BlockID,BlockID,BlockID
```

HL - Host List Block

A Host List (HL) Block contains a list of Host IDS and their anticipated longevity timestamps. Host Lists are composed and distributed by each Host as part of their Idle processing. Host IDs are contained in Directory List (DL) and Host List (HL) records received by each Host. These are combined and consolidated into new lists which are periodically pushed to other Hosts. Only the largest timestamp associated with a given Host ID is retained. The Host's own ID is included in any Host List (HL) Block pushed.

In general, contents of a Host List (HL) Block are prepared during the Idle phase of a Host's operation.

Host List (HL) Blocks are also prepared as a response to a Query in which Directory Lists are scanned for a particular Block ID. Hosts known to possess the required Block ID(s) are included in the response Host List (HL) Block.

The longevity timestamp is a time in the future after which the Host ID will be deemed to have expired. The Host computes its own longevity timestamp from the median duration of the most recent five times the Universe Kernel ran. The longevity of all other Hosts is the latest timestamp that has been seen for that Host.

The format of a Host List (HL) Block is printable ASCII text as follows:

```
##HL
<ID=HostID,030721123456>
<ID=HostID,030721123456>
<ID=HostID,030721123456>
```

QU - Query Block

A Query (QU) Block is a block of printable text in an XML-like format. It contains parameters that are used by a Host to search other Blocks contained in its Repository. Query Blocks are processed asynchronously on each Host, as time permits. Successfully finding a desired Block results in a response being sent back to the originator of the Query. Responses are simply data Blocks pushed back to the originating Host. Queries which fail may be replicated intact to other Hosts where they will also be processed. After processing (or expiration) Query Blocks are eliminated from each Host's repository.

The format of a Query (QU) Block is printable ASCII text as follows:

```
##QU
<ReplyTo=HostID,030721123456>
<Expiration=030721123456>
<BlockInterval=secs>
<Search=UD/FD/DL>
<Reply=UD/FD/DL/HL>
<ReplyMax=nnn>
<FanOut=nn>
<Query=expression>
<Phrase=expression>
```

The **<ReplyTo=HostID,longevity>** parameter indicates the Host that originated the Query. It is the Host ID to which any successful responses will be sent. The longevity is included to allow Hosts that receive the Query to update their Host List (HL) records so that direct communication with the Query originator will be possible. This parameter is required.

The **<Expiration=timestamp>** parameter indicates a time after which the Query will be discarded by all Hosts. No further responses will be sent after this time. In general, Hosts will also discard Queries with Expiration times too far into the future. This helps prevent (hypothetical) malicious or mal-formed Queries from overwhelming the ReplyTo Host. This parameter is required.

The **<BlockInterval=secs>** parameter indicates the number of seconds that will elapse between Block transmissions resulting from this Query. If the query was successful, it may return many result blocks. This interval specifies how fast these blocks will be sent to the ReplyTo HostID. If the query fails, the Query Block itself may be sent to other Hosts to process. This interval specifies how fast these replica Query Blocks are to be sent. If not specified, the Host determines based on its configuration parameters.

The **<Search=UD/FD/DL/HL>** parameter specifies the nature of the Blocks to be searched by the Query. One or more of the four options will actually be included in the parameter. This parameter is required.

The **<Reply=UD/FD/DL/HL>** parameter specifies the nature of the Blocks to be returned by the Query. One or more of the four options will actually be included in the parameter. This parameter is required.

Not all combinations of Search= and Reply= parameters are valid. The table indicates the results to be expected from each possible pair. The actual implementation may execute several of the ten meaningful operations based on multiple values for Search= or Reply=. This allows, for example, the return of FD and UD Blocks pertaining to a particular file with a single Query.

		Search (Domain)			
		UD Search the UD Block IDs (not contents)	FD Search the contents of the FD Blocks	DL Search the contents of the DL Blocks	HL Search the contents of the HL Blocks
Reply (Range)	UD	Return the UD Blocks themselves	Return the UD Blocks listed in matching FDs		
	FD		Return any FD Blocks with specific contents.		
	DL	Return a DL Block with only the matching UD Block IDs	Return a DL Block with only the matching FD Block IDs	Return any DL Blocks with specific contents.	
	HL	Return a HL Block with only the current Host listed.	Return a HL Block with only the current Host listed.	Return a HL Block with only the current Host listed.	Return any HL Blocks with the specific contents.

Using Reply=HL is generally reserved for a preliminary Query which could (potentially) result in a flood of responses. The compact, single Host response minimizes bandwidth requirements and allows the ReplyTo Host to choose the strategy for additional queries.

The **<ReplyMax=nnn>** parameter indicates the maximum number of Blocks that will be sent to the Host ID specified in the ReplyTo= parameter. This allows a limit to be placed on traffic that will be transmitted as a result of a particular Query. If not specified, one response Block will be allowed.

The **<FanOut=nn>** parameter specifies the number of Hosts that an unsuccessful Query will be replicated to. In general, Queries that succeed return data to the ReplyTo=HostID. Queries that fail are sent to nn randomly selected other Hosts in an attempt to generate some success. If not specified, queries are not replicated and are simply discarded with no response. A FanOut value of one will cause the Query Block to move randomly from one Host to another until (1) it succeeds and sends a response to the ReplyTo HostID, (2) it expires and is discarded, or (3) it returns to a Host that has already processed it and is discarded.

The **<Query=expression>** parameter specifies the actual boolean expression used to search the repository. Operands in the expression are given names in <Phrase=ID:expression> parameters. One Query=expression parameter is required in a Query Block. The evaluation of the expression yields a True or False value which ultimately determines the success or failure of the entire Query.

The **<Phrase=ID:expression>** parameter specifies the named operands and literal comparisons to be used in the repository search. A separate Phrase=ID:expression parameter is required to define each different operand used in the Query=expression.

Search Expressions

Search expressions are composed in a partially pre-parsed within Query Blocks. The syntax of complex searches is thus moved to the user interface application and is not resident in the Data Universe Kernel. This is an attempt to provide functionality similar to regular expressions but without the computational overhead.

Examples of Search Expressions follow:

Examples are TBD

Future Extensions

A simple ASCII text method of describing simple directory searches was outlined above. Extensions to this concept would allow any computation to be requested via this standard Query Block mechanism. Input data for the computations is available from User Data (UD) blocks in the Universe. Complex computations could be described in User Data (UD) Blocks containing Java applets, or (with much more danger to Host integrity) actual executable programs. Computational results would be returned to the requesting Host as new data Blocks just as the results of a directory search are in the reference implementation.