# EDOS Distribution System: a P2P architecture for open-source content dissemination

Serge Abiteboul[1], Itay Dar[2], Radu Pop[3], Gabriel Vasile[1] and Dan Vodislav[4]

1. INRIA Futurs, France {firstname.lastname}@inria.fr
2. Tel Aviv University, daritay@post.tau.ac.il
3. INRIA-Mandriva, Paris, France radu.pop@inria.fr
4. CEDRIC-CNAM Paris, France vodislav@cnam.fr

**Abstract**. The open-source software communities currently face an increasing complexity of managing the software content among theirs developers and contributors. This is mainly due to the continuously growing size of the software, the high frequency of the updates, and the heterogeneity of the participants. We propose a distribution system that tackles two main issues in the software content management: efficient content dissemination through a P2P system architecture, and advanced information system capabilities, using a distributed index for resource location.

## 1 Introduction

Faced with the increasing need of sharing, retrieving and loading data on the web, the problem of distributing content to large communities across the web has acquired a growing importance.

In the particular case of open-source software distribution (e.g. Linux), very large amounts of data (tens of Gigabytes) must be disseminated to a very large community of developers and users (up to millions of members). Moreover, content is frequently updated to new versions of the software modules. For a Linux distribution, content is generally disseminated either as ISO images of a full Linux release, or as packages that group binaries or source code for a single software module. The problem with the first approach is that successive Linux releases have many common parts that users will uselessly download several times. The finer granularity in the second approach requires more complex data management, with frequent package updates and freshness problems.

The main requirements for an open-source software distribution system could be summarized in four points: (i) avoid excessive charges on the distribution servers and on the communication lines, that lead to poor global performances; (ii) provide advanced search of content based on metadata properties; (iii) provide support for maintaining freshness on content for each user in the distribution network; (iv) ensure robustness in case of failure of some system components. Current distribution

*Please use the following format when citing this chapter:*

Abiteboul, S., Dar, I., Pop, R., Vasile, G and Vodislav, D., 2007, in IFIP International Federation for Information Processing, Volume 234, Open Source Development, Adoption and Innovation, eds. J. Feller, Fitzgerald, B., Scacchi, W., Sillitti, A., (Boston: Springer), pp. 209–215.

architectures, centralized or based on a set of mirrors, fail to fulfill these requirements. We believe that peer-to-peer (P2P) architectures, that uniformly share the effort among participants and provide replication, are a good solution for software distribution.

We present here the content distribution solution proposed in the context of the EDOS European project [5]. EDOS stands for *Environment for the development and Distribution of Open Source software* and addresses the production, management and distribution of open source software packages. We only present the EDOS content distribution system, which proposes a P2P dissemination architecture including all the participants to the distribution process: publishers, mirrors and clients. The system was implemented as an application to the distribution of Mandriva Linux packages.

The main contributions of our system are: (i) aP2P architecture providing resource sharing, load balancing and robustness; (ii) advanced information system capabilities, based on distributed indexing of XML content metadata; (iii) efficient dissemination based on clustering of packages and multicast; (iv) support for freshness maintaining on updates, by using subscription/notification. Due to space limitations, the paper only presents an overview of the EDOS distribution system –a detailed description may be found in [8].

## 2 Related work

Linux distributions use various dissemination methods (an overview is presented in [7]), based on sets/hierarchies of mirrors in most cases, on notification channels in RedHat Network [15], or on versioning repositories in Conary [4].

P2P architectures for content distribution mainly address load balancing and bandwidth sharing (Coral [13], Codeen [3]). We extend this primary use by adding a *distributed information system* based on XML metadata indexing and querying, together with efficient *file sharing and multicast dissemination*, such as BitTorrent [12]. Among the various P2P infrastructures [11], the most appropriate in our context are *structured overlay networks* (Chord, Pastry, CAN, ...), that provide better performances for locating and querying large quantities of data. We use Free Pastry, an implementation of the Pastry [14] distributed hash table system.

## 3 System functionalities

The goal of the EDOS distribution system is to efficiently disseminate open source software (referred at a more general level as *data or content*) through the Internet. Published by a main server, data is disseminated in the network to other computers (mirrors, end users), that get copies of the published content.

EDOS system is articulated around a distributed, P2P information system that stores and indexes *content metadata*. This metadata-based information system allows querying and locating data in the EDOS network.

The choices for the functional architecture are driven by the three main aspects that define the system: the data model for the content management, the actors and their roles in the P2P architecture, and the usage scenarios.

### 3.1 Data model

There are *three types* of data units employed by the EDOS distribution system:
- **Package**: the main data unit type, represented by an RPM file;
- **Utility**: individual file used in the installation process;
- **Collection**: it groups together packages, utilities or sub-collections, to form a hierarchical organization of data.

A *release* is a set of data units that form a complete software solution -it corresponds to a full Linux distribution. Its content is described by a collection.

*Content dissemination* is initiated by *publishing* data units in the system. Publishing consists in generating metadata for each data unit and indexing it in the distributed system. Periodically, the main server publishes a new release. Updates to the current release are realized by publishing new versions of packages or utilities. At some moment, the main server decides to transform the current status of the current release into a new release.

**Metadata management** is a key issue in the distribution process. We aim at building a global, distributed information system about data to be disseminated in the network. This system is fed with content metadata. The ability to express complex queries over metadata and to provide effective distributed query processing is a major contribution of this project.

In the largest sense, metadata consists in the set of properties that characterize data units. We classify metadata properties in three main categories:
- *identifiers*, i.e. properties that uniquely identify a data unit –in our case, the name and the version number of the data unit.
- *static properties*, that do not change in time for a content unit, e.g. size, category, checksum, license, etc.
- *changing properties*, i.e. properties that may vary in time: locations of replicas in the network and composition for collections.

The XML structure chosen for EDOS metadata is a compromise between efficiency needs for both *query processing* (that requires large XML files, containing all elements addressed in a query) and *metadata updates* (that need small files). Our choice is to create separate XML files for each package (package properties) and for each release (release composition).

### 3.2 Actors, roles and usage scenarios

Peers of the EDOS P2P distribution system maybe classified in *three categories*:

1. **Publisher**: the main distribution server, that introduces new content in the system. Its roles are to *publish* the new content in the distributed index, to manage client *subscriptions/notification*, and *flash-crowd dissemination* of data, as explained below.

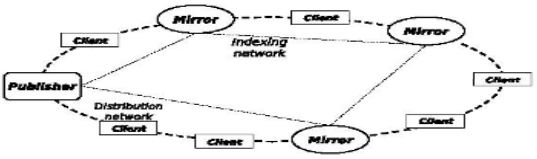


Figure 1 presents the actors in the P2P distribution network.

2. **Mirrors**: secondary servers and trusted peers. Mirrors and end-users have similar roles: *download* content from other peers, query the system and *subscribe* to new data. The main role is to keep copies of the published content, providing additional downloading sources in the network. Unlike end-users, Mirrors are trusted peers, that can participate in *index management*. Also, they are rarely unavailable and provide better QoS.
3. **Clients**: end-user computers, not trusted for index management. They need an entry point to the indexing network for querying the metadata.

Figure 1 presents the actors in the P2P distribution network. Actors are connected in two distinct networks:

– *The distribution network*, composed of all the peers - they store, download and share EDOS data, i.e. software packages, utilities and collections.
– *The indexing network*, composed of trusted peers (Publisher and Mirrors) - they store the index on content metadata. For security reasons, Clients are not allowed to participate in metadata and index sharing, but can provide content, whose validity may be verified by using checksums.

There are two main distribution cases: flash-crowd and off-peak.

**Flash-crowd** distribution corresponds to situations where new, popular and large size content is published (typically a new release), and many users want to get this content as soon as possible. Flash-crowd distribution uses efficient dissemination methods, based on clustering of data units and multicast. Each user asking for the new release may already have some of its packages -therefore he computes a *wish list* containing only the missing data units. Based on the wish lists gathered from users, the Publisher computes the clusters of data units to be disseminated. Each user will only download the minimal set of clusters that cover its wish list -download is realized in parallel for all the users in a common *multicast* process.

**Off-peak** distribution corresponds to periods between flash-crowd situations. During these periods, the Publisher may publish updates to the current release, Mirrors and Clients may query the system, download query results, subscribe to distribution channels, receive notifications on such channels and download software updates.

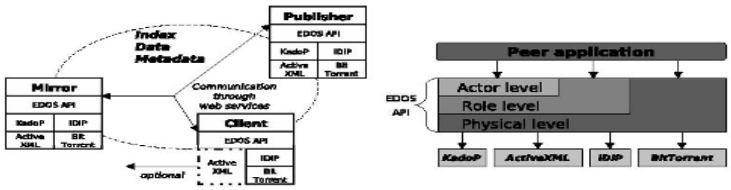


Fig. 2. Software modules and API structure in the EDOS distribution system

## 4 Architecture and implementation

The EDOS distribution functionalities have been implemented as a Java API, based on a set of external software modules:

- **ActiveXML** [9, 1]: provides an extended XML format for EDOS metadata, storage for metadata documents published in KadoP, and web services for inter-peer communication.
- **KadoP** [10, 6]: distributed index for (Active)XML documents, that allows publishing, indexing and querying EDOS metadata. Based on the Distributed Hash Table (DHT) system Pastry [14], KadoP uses the ActiveXML module as a local repository for metadata documents KadoP decomposes ActiveXML documents into key-value pairs stored in the DHT, and uses this decomposition to compute answers at XML query processing.
- **IDiP**: dissemination platform that implements functionalities for the flash-crowd usage scenario: content clustering and multicast dissemination.
- **BitTorrent** [12, 2]: the well-known filesharing/downloading system, that optimizes the transfer of large files between peers. We use a slightly modified version of *Azureus*, a Java implementation of BitTorrent, for multicast in IDiP and for download from multiple replicas.

*The structure of the EDOS distribution API* is presented in Figure 2. The API is organized on three levels:

1. **Physical level**: lowest level, provides EDOS peer basic functionalities. The physical level is composed of several modules: a content manager for local content, an index manager for the distributed index, a channel manager for subscription, a dissemination manager for flash-crowd distribution, etc.

Programming distribution applications at the physical level requires more effort, but offers the greatest flexibility.
2. **Role level**: built on top of the physical level, provides a default implementation for each role in the distribution network, i.e. publishing, downloading, replicating, querying, and subscribing.
3. **Actor level**: highest level, provides a default implementation for each actor type (Publisher, Mirror or Client), by combining several roles.

The first **prototype** of the EDOS distribution system is implemented as a set of web applications on top of the EDOS API (seeFigure2). Each peer in the EDOS network runs a Java/JSP web application -there is an application for each actor type: Publisher, Mirror or Client. Peer applications use a Tomcat web server for deployment, with Axis for web services.

The Publisher web application allows publishing new content, managing subscription channels and driving the flash-crowd dissemination process. Mirrors and Clients have the same user interface, allowing queries, downloading, subscriptions to channels and notification handling.

Tests with the first prototype demonstrated the relevance of P2P-based solutions for large-scale content distribution, the ability of managing very large amounts of metadata with KadoP and the improvements brought by IDiP for flash-crowd dissemination. More details are presented in [8].

Next steps will address intensive testing in a real large scale network such asGrid5000 and improvements in massive publication of metadata, in security (peer authentication), in firewall/NAT traversal, in the user interface, etc.

## References

1. ActiveXML web page. http://activexml.net.
2. BitTorrentprotocolspecification.http://wiki.theory.org/BitTorrentSpecification.
3. Codeen. http://codeen.cs.princeton.edu.
4. Conary software provisioning system. http://wiki.rpath.com/wiki/Conary.
5. EDOS project: Environment for the development and Distribution of OpenSource software. http://www.edos-project.org.
6. KadoP web page. http://gemo.futurs.inria.fr/projects/KadoP.
7. EDOS deliverable 4.1: Distribution of code and binaries over the Internet, 2005. http://www.edos-project.org/xwiki/bin/view/Main/D4-1/edos-d4.1.pdf.
8. EDOS deliverable 4.2.2: Report on the p2p dissemination system, 2006. http://www.edos-project.org/xwiki/bin/view/Main/D4-2-2/edos-d4.2.2.pdf.
9. S. Abiteboul, O. Benjelloun, I. Manolescu, T. Milo, and R. Weber. Active XML: Peer-to-Peer Data and Web Services Integration. In VLDB, 2002.
10. S.Abiteboul, I. Manolescu,and N. Preda. Constructing and querying peer-to-peer warehouses of XML resources. In V.T.ChrisBussler, editor, Second International Workshop on Semantic Web and Databases (SWDB). Springer-Verlag, 2004.
11. S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. In ACM Computing Surveys, 2004.

12. B. Cohen. Incentives Build Robustness in BitTorrent. In Proceedings of the Workshop on Economics of Peer-to-Peer Systems, 2003.
13. M. Freedman, E. Freudenthal, and D. Mazieres. Democratizing content publication with coral. In 1st USENIX/ACM Symposium on Networked Systems Design and Implementation, 2004.
A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In IFIP/ACM Middleware, 2001.
14. S. Witty. Best practices for deploying and managing linux with RH Network.