

# Open Web Repository Synchronization

---

## 1. Introduction

Our recent research reveals that many Internet companies hold repositories of Internet website pages' crawled and parsed data (texts, images, video, etc), web page visit statistics and other publicly available documents/articles. Those repositories could be categorized to specialized and general. Specialized repositories have captured/filtered data used for specific analysis or vertical searches. General repositories are commonly used for horizontal searches.

Web pages crawled content is duplicated in several repositories and statistical data is incomplete per each repository. These repositories in most cases are proprietary and its access is reserved for owners' indexing or analyzing software tools.

Open Web Repository Synchronization (OWRS) defines four schemes for Repositories cooperation.

- First Scheme enables to organize available data exchange between Repositories.
- Second Scheme enables to share crawling tasks between different Repositories.
- Third Scheme which enables companies to provide their computing resources to Open Repositories for receiving necessary crawled web data in return.
- Fourth Scheme enables companies to aggregate accurate usage statistics jointly.

Benefits of OWRS are efficiency of crawling resources utilization, integral website statistical data acquisition, etc. Third and fourth schemes enable multiple repositories to be integrated into a single Open Repository, thereby various practitioners and researchers may process high-volume data available in a single directory which will assist the development of Semantic Web.

The main goal to be achieved is to boost new technologies for analyzing of websites content and visit statistics, public documents, etc.

## 2. General Concepts

### 2.1. Open Web Repository Synchronization Abstraction

Open Web Repository Synchronization Abstraction (OWRSA) is an abstract model which defines entities and describes the procedures necessary for Open Data collection, synchronization and exchange between multiple repositories. Open Data is a record (document, article, etc.) that does not contain any secure or private information.

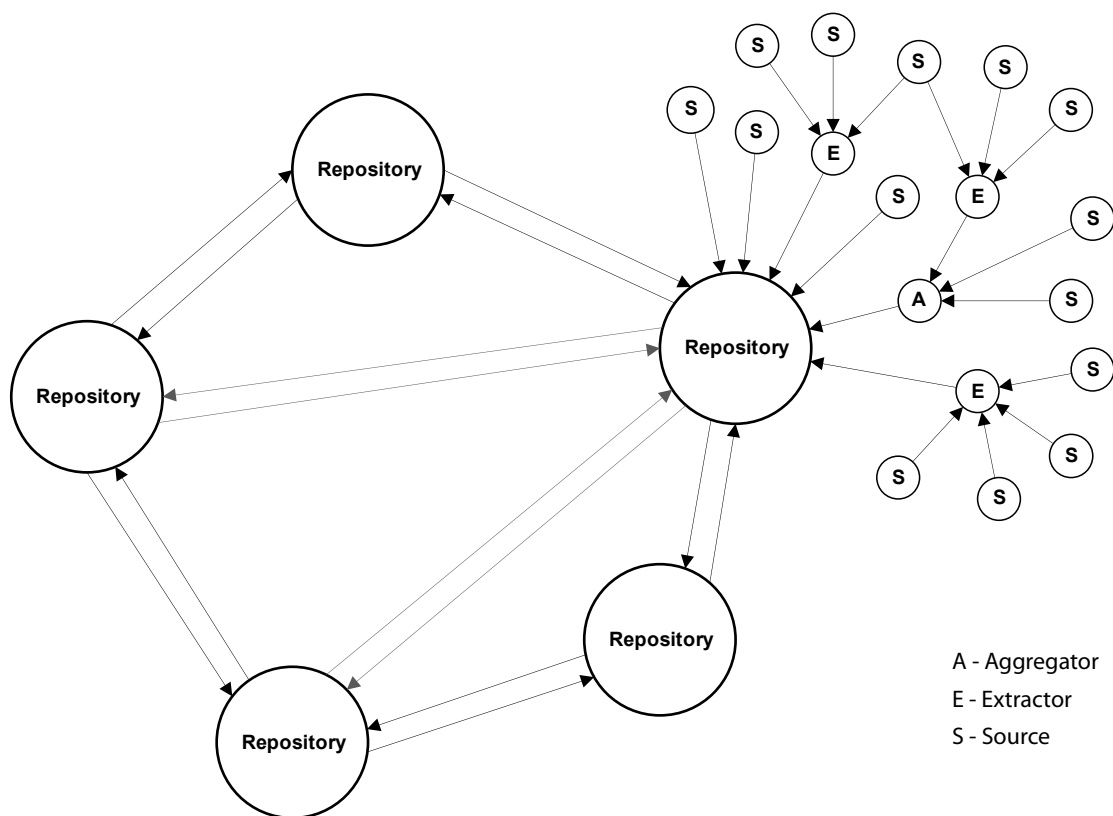
OWRSA defines the following entities:

- **Source** - declares, distributes and publishes Open Data for commercial, legal and/or social purposes (e.g. website).
- **Extractor** - extracts data from multiple Sources (e.g. web crawler).
- **Aggregator** - collects data from Extractors and/or Sources and sends to Repositories.
- **Repository** - forms collected Open Data archives.
- **User** - retrieves and uses Open Data.
- **Coordinator** - coordinates Open Data sharing and crawling synchronization processes.

# Open Web Repository Synchronization

- **Manager** - controls crawling procedure for single Repository.
- **Dashboard** - directory available to all entities for read/write operations.

**Figure 1. Abstract model of Open Data collection**



OWRSA supposes four possible data collection chains:

- Sources send Open Data to Repository directly;
- Sources send Open Data to Repository through Aggregators;
- Extractor gets Open Data from Sources and sends to Repository directly;
- Extractor gets Open Data from Sources and sends to Repository through Aggregators.

OWRSA model implies verification procedure which is aimed to assure data extraction and aggregation correctness.

OWRSA undertakes specified four schemes of Repositories cooperation and correspondingly defines following concepts:

- Data Exchange Based on Reciprocity
- Crawling Synchronization
- Open Web Repository
- Open Web Statistics

## Open Web Repository Synchronization

---

### 2.2. Packages

OWRSA defines abstract model of Packages for data transmission between OWRSA entities. Package is an associative array which elements are either simple values or packages (hierarchical packages). OWRSA defines interface to ensure package processing. According to defined interface packages are being serialized to be recorded or transmitted via network connection, and recipient de-serializes received stream to process data sent in the package. Packages abstraction and serialization increases efficiency of hierarchic data exchange for both binary and textual data.

Each package has at least three elements: package ID, sender ID and package type.

Package ID is a unique number that consists of following parts: creator ID, generator ID and sequential number. Creator ID is the identifier assigned to the entity which has created specific package. In most cases it matches with sender ID. Generator ID is the identification number of the module generated specific package. Sequential number is an automatically generated sequentially increasing number assigned to each package.

Sender ID is the identifier assigned to the entity which has sent specific package.

Package types are being defined by each OWRSA scheme separately in accordance with specified communications between entities. Depending on type the package may have additional elements which number and type is not limited.

Serialized data stream consists of two fields: serialization type and body, where type indicates serialization mode and body contains the serialized package. Serialization type has the following structure: general-mode/sub-mode.

### 2.3. Security

OWRSA uses asymmetric cryptography to encrypt serialized packages which assumes two approaches: digital signatures and public key encryption. To assure high level of privacy and security, OWRSA implements combination of asymmetric cryptography approaches. It helps to ensure authenticity of the sender and confidentiality of sent data simultaneously.

Any entity randomly generates two pairs of public and private keys: one pair for packages encryption and one for packages decryption. Encryption and decryption public keys are submitted to Dashboard for other entities usage.

OWRSA assumes two possible ways of package encryption depending on specified communication features:

- sender encrypts a package using its encryption private key only (digital signature);
- sender encrypts a package firstly by its digital signature, and secondly using recipient's decryption public key, so that recipient need to decrypt the package using its decryption private key and sender's encryption public key.

OWRSA security level adds security head to encrypted serialized packages.

### 2.4. Transport

OWRSA does not define any specific protocol or approach for packages transmission. It can be implemented using TCP, UDP, HTTP or any other protocol.

## Open Web Repository Synchronization

### 2.5. Reciprocity Score

OWRSA does not imply direct regulation of the communication between entities. OWRSA only encourages but not obliged entities to notify the Coordinator about received and sent packages. Coordinator logs received information details into the communications history directory of corresponding entity on Dashboard.

Each entity should notify Coordinator about readiness to provide action logs or not. Based on history logs Coordinator periodically calculates Reciprocity Scores for all entities. If specific entity does not notify about readiness to provide any communication history or notifies but does not perform, that entity's Reciprocity score will be negative.

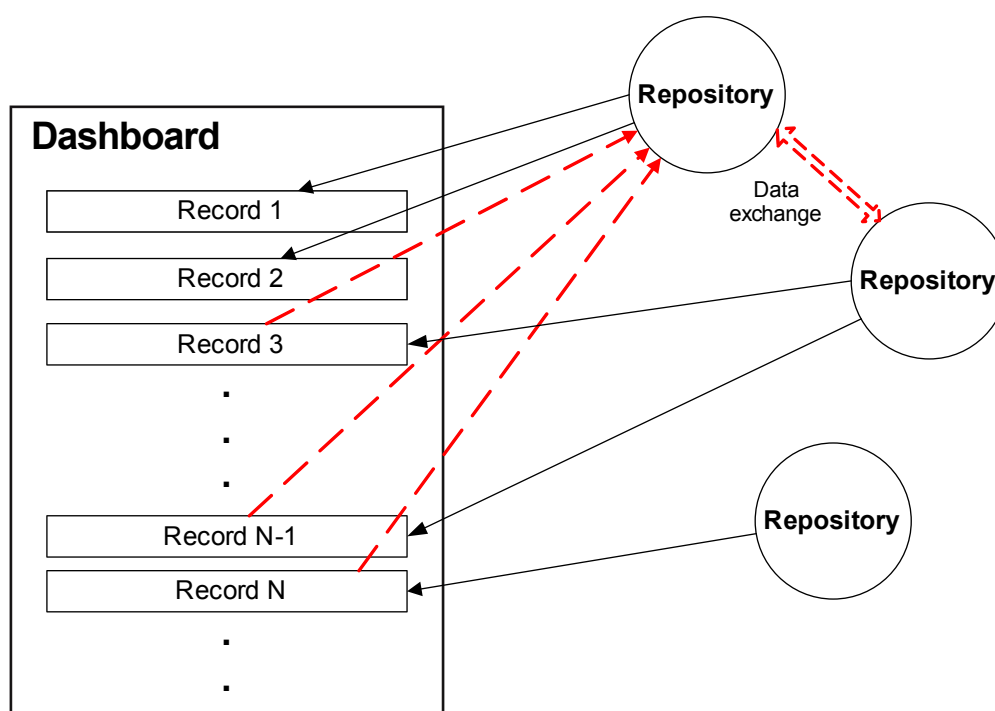
### 3. Data Exchange Based on Reciprocity

Data Exchange Based on Reciprocity (DEBR) is the first scheme of OWRS that enables to organize available data exchange between Repositories. DEBR relies on OWRSA Coordinator, Registrar and Repository entities. DEBR assumes data exchange procedure based on "publication" operation. Coordinator and Repositories may "publish" any record on Dashboard using corresponding packages.

DEBR defines package types which allow Repositories to publish the following information:

- list of available content's Sources
- periodicity of content updates
- list of necessary Sources

**Figure 2. Data Exchange Based on Reciprocity**



## Open Web Repository Synchronization

Published information is available to other Repositories. Any Repository may find records of interest and send corresponding data exchange suggestion package to a Repository which holds necessary Sources' content. Responding Repository may accept/reject received suggestion or send another suggestion package to requestor. As soon as responding side accepts suggestion, Repositories exchange data and notifies Coordinator about successful/failed communication. Coordinator logs communication history and recalculates Reciprocity Scores for those Repositories.

### 4. Crawling Synchronization

Crawling Synchronization (CS) is the second scheme of OWRS which enables to share crawling tasks between different Repositories. CS allows Repositories to crawl different Sources and exchange extracted content, thereby increases efficiency of resource sharing and data exchange. CS declares the following concepts and defines corresponding package types (simple or hierarchical) for them: Source Profile, Source Group and Group Profile.

Source Profile stores Source content characteristics. For example, if Source is a website, Profile may contain the following fields:

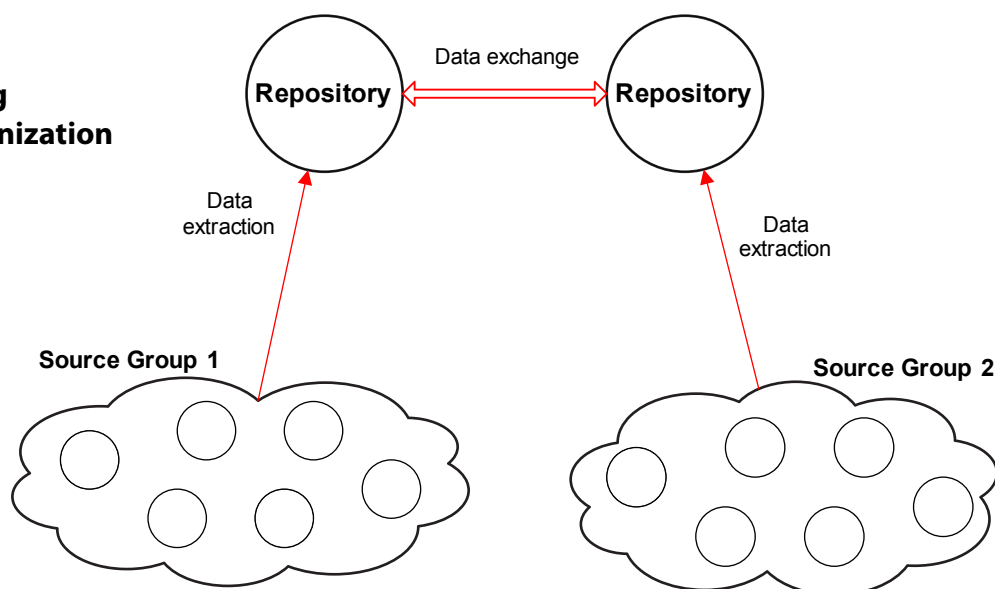
- size of website in bytes
- number of pages
- average size of one page in bytes
- number of images and its proportion to number of pages
- number of PDF files and its proportion to number of pages
- number of SWF files and its proportion to number of pages

Coordinator receives Profiles for the same Source from several Repositories and generates integrated accurate Source Profile. Cluster analysis tools are being used to generate Sources Groups based on Source Profiles characteristics. Coordinator assigns unique ID to each Group, generates Group Profile and places those data on Dashboard.

CS defines package types which allow Repositories to publish the following information:

- list of available crawled Source Groups
- periodicity of content updates
- list of necessary Source Groups

**Figure 3. Crawling Synchronization**



## Open Web Repository Synchronization

Data exchange between Repositories is implemented in the same way as DEBR but for Source Groups.

### 5. Open Web Repository

Open Web Repository (OWR) is the third scheme of OWRS which enables companies to provide their computing resources to Open Repositories for receiving necessary crawled web data in return. OWRS assumes two roles:

- Crawling Resource Contributor, which provides operational resources and Internet bandwidth.
- Open Repository Beneficiary, which gets access to Open Repositories content.

OWR undertakes financial reimbursement scheme for Crawling Resource Contributors. Open Repository Beneficiaries pays for computing resource utilization. The same entities may act in two roles at the same time. In this case provided computing resources may be reimbursed by received data. The main principle of OWR is paying for servers' crawling resource utilization only.

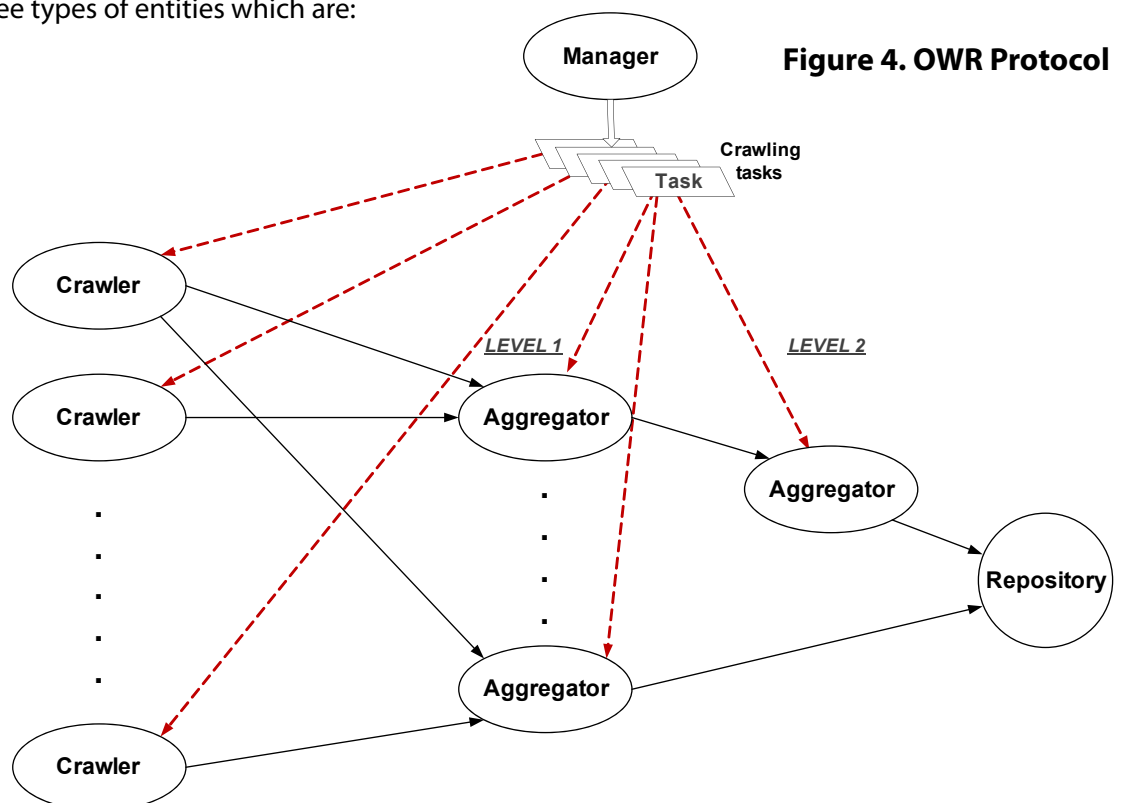
Contributed computing resources will be used for open crawling framework which can either be an open-source product or belong to all Crawling Resource Contributors and Open Repository Beneficiaries.

#### 5.1. OWR Protocol

One of the OWR protocol's basic concepts is a Crawling Task - a record which contains crawling details: targeted host name, URL patterns, crawling start and end schedule, list of aggregators to submit crawling results, crawling methods and plug-ins.

OWR protocol defines three types of entities which are:

- Crawler/Extractor
- Aggregator
- Manager



**Figure 4. OWR Protocol**

# Open Web Repository Synchronization

Crawlers notify the Manager about available free computing resources such as network connection bandwidth, hard drive free space, RAM and CPU. Based on received information from Crawlers, Manager assigns them corresponding Crawling Tasks. Crawler executes Crawling Task and submits crawling results to specified Aggregators along with computing resources spent.

Aggregators form a hierarchical network to assure high level of extracted data accuracy. Aggregators get crawling results from Crawlers, verifies received data and submits to the next level Aggregators or directly to Repository.

Manager’s main purpose is to link and co-ordinate Crawlers and Aggregators (hereinafter “nodes”). Manager registers the nodes by assigning them unique IDs. Manager detects the optimal topological structure of the nodes, and sends corresponding Tasks to Crawlers and Aggregators.

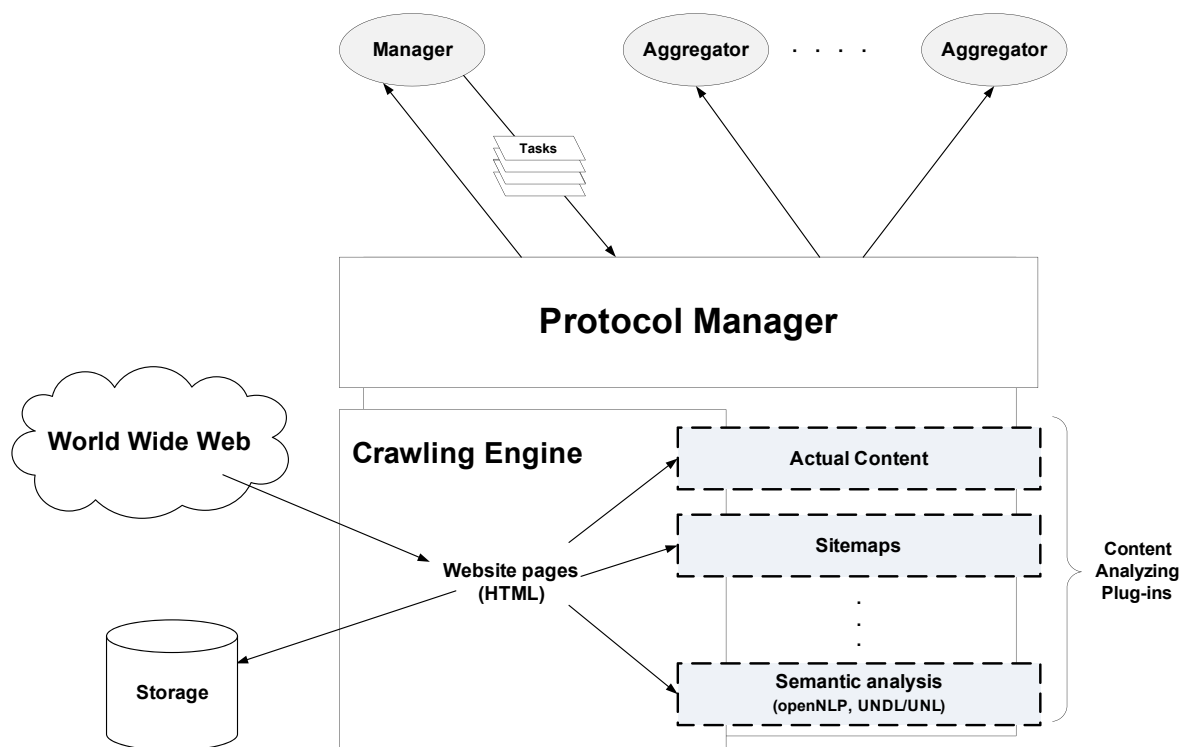
## 5.2. Open Web Repository Crawler Architecture

Open Web Repository Crawler Architecture (OWRCA) is pluggable framework architecture of web crawling client application.

OWRCA consists of four main parts:

- Protocol Manager
- Crawling Engine
- Analyzing Plug-ins
- Spent Resource Calculator

**Figure 5. Open Web Repository Crawler Architecture**



Protocol Manager makes possible communication between Crawler and other entities (e.g. Manager and Aggregators). Protocol Manager gets Crawling Tasks from OWR Manager and starts crawling processes in accordance with given schedule and mode.

## Open Web Repository Synchronization

Crawling Engine is a multi-thread system with parametric number of threads which is responsible for crawling algorithm realization. It downloads website pages, detects internal and external links, stores HTML content in the storage and activates content analyzing plug-ins if any.

Crawling Engine has two modes: simple and extended. The main difference between these modes is that in case of extended mode Crawling Engine embeds Mozilla Firefox web browser and Mozilla Firefox plug-ins to interpret and analyze JavaScript codes.

Content analyzing plug-ins perform additional analysis of downloaded content and submit results into a storage with certain format. For instance, OWRSA allows to integrate open text analyzing tools into crawling client application, such as Open Natural Language Processing (OpenNLP) and UNDL/UNL generator.

Spent Resource Calculator is responsible for calculation of computing resources spent on each Crawling Task execution per each and all sub-modules. Calculated data is converted into a package with corresponding format to be sent to a Manager.

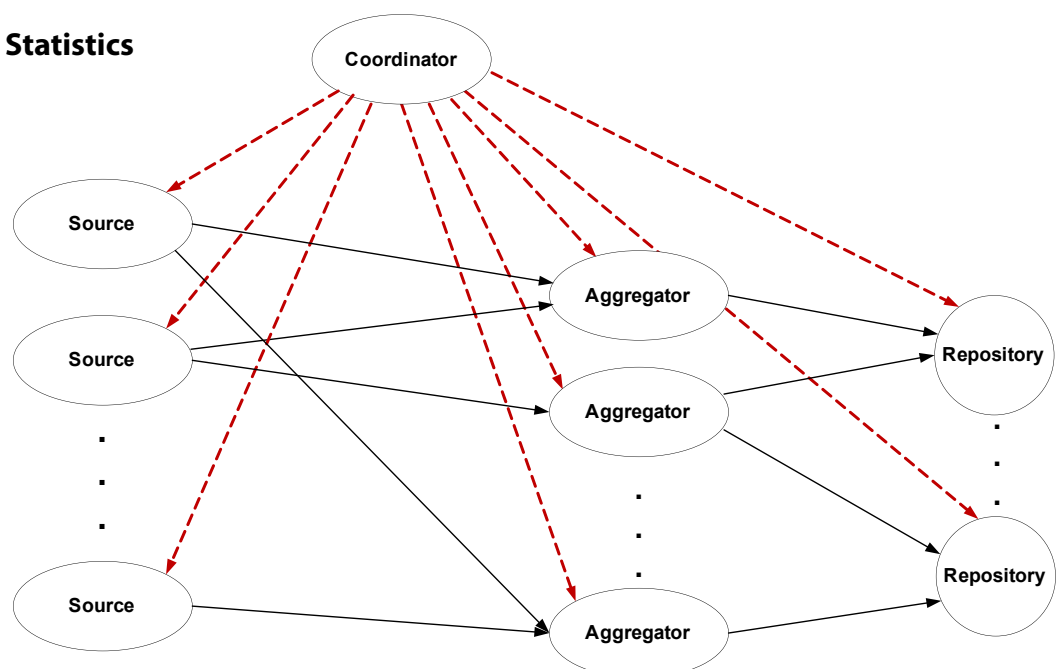
### 6. Open Web Statistics

Open Web Statistics (OWS) is the forth scheme of OWRS which enables Sources to publish their usage statistics to all interested entities. It is implemented by the following entities: Coordinator, Source, Aggregator and Repository.

Each Source requests Coordinator for the list of Aggregators to submit its visit statistics. Coordinator selects Repositories which are interested in that Source statistics and works out Aggregators' list to be sent as response. Coordinator may rearrange Aggregators based on requests received from Repositories.

Source submits visitors' action logs (referral URL, clicks, downloads, outcome URL, etc.) to assigned Aggregators which submits received information to corresponding Repositories.

**Figure 6. Open Web Statistics**



## Open Web Repository Synchronization

---

### 7. Open Web Integration

OWRSA undertakes two possible ways of Open Web Integration which are:

- URI scheme for entity profiles access on Registrars;
- New XML domain and tags deployment.

#### 7.1. URI scheme

Entities' profiles stored on Registrars are publicly available information. OWRSA defines the following URI scheme to access interested entities' profiles.

```
oweid://registrar/entity
```

where "oweid" (open web entity id) is the scheme name that refers to OWRSA specification; "registrar" is the authority component for the URI hierarchy that indicates certain Registrar storing necessary specific entity profile; and the remainder part - "entity" is an entity unique identifier assigned by Registrar which corresponds to URI path.

#### 7.2. XML domain

OWRSA defines new XML domain - "ow" for integration with any type of XML schemas (XML, RDF, OWL, etc.). "ow" defines two tags, which are:

- <ow:entity> - entity identifier
- <ow:copyright> - copyright protection

<ow:entity> has the following syntax:

```
<ow:entity id="oweid://registrar/entity">  
.....  
</ow:entity>
```

"id" refers to the entity/object which name is enclosed within <ow:entity> and </ow:entity> tags. It allows XML parser to detect specific entity even if it is described in an unusual form (e.g. instead of OMFICA it is indicated as OMFICA.org).

<ow:copyright> has the following syntax:

```
<ow:copyright owner="oweid://registrar/entity">  
.....  
</ow:copyright>
```

"owner" refers to the entity which holds copyright for the content enclosed within <ow:copyright> and </ow:copyright> tags. This mechanism allows Copyright Office Repositories to register copyright automatically and track possible cheating.