Canopener: Recycling old and new data

Cosmin Basca Rob H. Warren Abraham Bernstein
Department for Informatics
University of Zurich
Zurich, Switzerland
{lastname}@ifi.uzh.ch

ABSTRACT

The advent of social markup languages and lightweight public data access methods has created an opportunity to share the social, documentary and system information locked in most servers as a mashup. Whereas solutions already exists for creating and managing mashups from network sources, we propose here a mashup framework whose primary information sources are the applications and user files of a server. This enables us to use server legacy data sources that are already maintained as part of basic administration to semantically link user documents and accounts using social web constructs.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous

Keywords

Systems Mashups, Social Networks, Distributed Information Systems

1. INTRODUCTION

With the advent of lightweight mashup technologies, an increasing number of data sources are combined and exposed as composite data sources, expanding current web boundaries. To date, the focus of the vast majority of mashups has been to create and expose new data by aggregating existing external data sources such as web pages or web services. Content creation is still primarily driven either by large suppliers of semi-public content (such as flickr), or through the hand coding of information by an end user.

A number of outstanding issues remain with the current approaches. Firstly, in order to integrate data from various sources diverse standards for data interchange have to be employed. This gives rise to a high cost of aggregating and maintaining the translation formulas, which can lead to a reduced number of data sources being supported. Most recent is the trend towards the automatic generation and exchange of semantic information by the application themselves, as in the case of the Drupal Semantic Web module[4].

The demand for such approaches is high as it enables for the seamless transfer of semi-structured information from an application to the web. The promise is especially apparent given the large amount of information locked in legacy databases and applications that were never meant to interact with other systems.

We propose Canopener a mashup architecture that enables one to expose system and user application information over the web using RDF [1] as the data interchange format. The information can be either statically or dynamically collected and is exposed via a SPARQL [2] endpoint allowing for expressive querying of the underlying data and the creation of enterprise level mashups with tools such as Semantic Web Pipes [10].

Whereas as in a number of solutions, the mashup is viewed from the perspective of the visualisation client, which agregates the data from multiple sources. Canopener creates a semantically integrated mashup from all of the user and application information that is available from the server.

In itself Canopener provides an integration endpoint at which information that should be shared can be added. It provides a mechanism for the automated discovery of the information available on a server, an amalgamation of the information into a server-specific mashup, a mechanism to schedule the data integration and the monitoring of installed application so that the data integration mechanisms do not become stale. It essentially provides a view of the information on a server that is cross referenced with the social information that can be derived from administrative information. The Canopener information is a mashup that can be transmitted to other applications as a file from the webserver, a SPARQL endpoint or through a webservice implementation.

Figure 1 is a representation of the Canopener Mashup. A small Canopener-specific ontology contains site-specific information such as server geo-location, historical daily workload in percentage and historical power loss information[3]. A second container lists other RDF markups of documents, applications, groups and individuals that are part of the community (or neighbourghood) of this server. The generic container is also intended to provide a flexible and easy to deploy solution to the *Sensing the real world* principle identified by Hopper and Rice[8] and to other dynamic data dissemination problems.

The paper is organized as follows: we begin by elaborating the rational and methodology of the approach, followed by the overall architecture of the mashup engine. Three different case studies are presented as well as possible future research directions given large scale data aggregation. We

WWW2010, April 26-30, 2010, Raleigh, North Carolina.

[1] `http://www.w3.org/RDF/`
[2] `http://www.w3.org/TR/rdf-sparql-query/`
[3] This should technically be a system application, but we put it here for demonstration purposes.

Figure 1: The Canopener Mashup currently provides basic system information through its own ontology container and application, and user data through a generic container.

then review the previous work in the area and conclude with future work.

Canopener makes the following contributions:

- It semantically links the information contained on a server using relationships implied from server administrative information such as user groups.
- It makes use of the package management subsystem as means of discovering the presence of integratable information and ensuring the ongoing maintenance of the integration recipes.

Canopener is an offshot of the development of the Avalanche distributed RDF framework, where disparate service metadata needed to be distributed in a lightweight fashion. It is used to advertise availability, policy and resource information that is not yet supported by available query protocols such as SPARQL.

2. APPROACH

Our proposed methodology rests on the observation that there exists a tendency to reinvent the same structures within information systems. To this end, we looked at the lowest element of the software stack: the Operating System (OS), which provides a number of low-level services to user and system applications such as identity, group membership and access control.

Given that in most cases we can separate purely administrative users and groups from actual people and social groups, Canopener takes the approach that the server itself is a community from which semantic information can be extracted and shared. Data from the installed applications can be exported as a single large mashup, according to the existing server configuration files, while at the same time being socially linked by both the user accounts and groups.

In a way, this is not unlike the sitemap.xml [12] markup used by some web sites to publish links to the resources available. Whereas such an approach is web document centric, ours is centric to the community of users on a specific server.

The novelty of this approach lies in the reuse of information which is already maintained within the server in order to determine what information can and should be within the mashup. Since the administration of the server is already being done for day-to-day operations, any reuse of its administrative database is a cost already paid for, not requiring additional effort. Furthermore, it promotes a model of data sharing that has a lower cost of operation in that applications that only share a small amount of information do not need their own mashup.

Most modern operating systems allow for complex Access Control Lists (ACL), including whether persons unknown should be allowed to read a document. This already serves to protect documents that could be published by a web server; it also protects files that contain user or system information that could be made public, such as IM accounts name, other public web server accounts, alternate mailboxes and profile information.

Similarly, group information can be used to generate membership information as well as group labels for mashup publications. Relationships declared through applications such as Instant Messaging, address books and cryptographic keys can similarly be exported as friendship constructs within a social web mashup.

By using Semantic Web technologies one can easily create different domain mashups without the need to redefine the interchange and integration format and techniques. Two components are needed to be manually created: 1) the mashup domain mapping, and 2) the actual legacy data collection / import plug-in.

This last element is actually a large obstacle: the problem of extracting information from system and user applications is with the large variety of applications available. Not only must a mashup generator be written for each of the applications, but we must discover whether or not the application is actually present on the server. Our solution is to link ourselves to the package manager of the server, which records what application (and version) is installed and allows us to determine whether information of interest can be extracted.

This does not completely solve the integration problem, as there exists an extremely large number of applications. But the package manager needs to be aware at a certain level of the information system of the application in order to install, configure and control it. This presents an opportunity for us to automate some of the integration and translation functions within the mashup, while using the package manager database for a list of possible data sources to translate.

Thus, our server-oriented mashup generation approach permits us to maintain a coherent mashup generation infrastructure. Its maintenance and content is guided directly by the policies already set by both the operating system, user security settings and whether a particular application is declared installed by the package manager. Lastly, the approach presents an opportunity to provide low value sensor and system data to the web for aggregation use at a negligible cost. If the servers security policy states that anyone can access the sensors (web-cams, weather stations, temperature monitors, etc...) then they are automatically exported to the mashup.

2.1 Information privacy considerations

Canopener is meant to support servers that facilitate collaborative activities for groups and individuals. As such,

this specialization assumes that there exists a limited amount of private or very private information within the individual user account, and that there is a need to communicate (broadcast) information more effectively. In this scenario, such as in the case of a software development project on an intranet, the distribution of the information is a wanted end-result. At the very basic level files which are visible and accessible through the web server are considered publishable by Canopener. More complex and sensitive information may or may not be exported according to the decision of the system administrator. Individual users themselves have the option of preventing their specific instance of that information from being published by using the file system access control list.

The perspective of Canopener is that of an agent for other mashups, as such it believes that if it can read a file then the specific information that it knows to extract from that file can be exported (if also permitted by the system administrator). Some user education may be needed to ensure proper ACL settings, however file system permissions are the primary security mechanisms of the server and should have fail-safe defaults.

3. ARCHITECTURE

To achieve the low-cost information reuse goal of our proposed approach, we strive to embody as much of the current system stack (both software and hardware) as possible within the proposed architecture.

Let us consider the stack of technologies as presented in Figure 2.

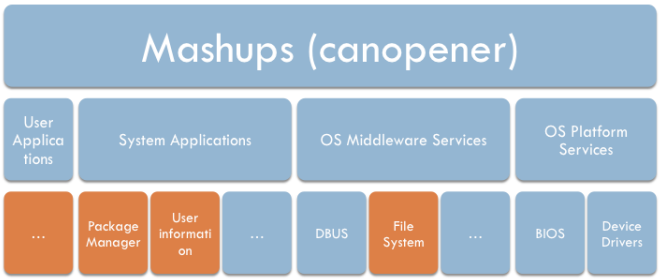

Figure 2: Technology stack

To ensure high flexibility in design, deployment and low cost we propose a simple yet powerful architecture based mainly on the current technologies that enable the Web. Most of the information produced by current Information Systems (IS), is either trapped inside the IS applications or exposed using highly customized systems meant for heterogeneous computing environments.

In the case where an enterprise wishes not to deploy another layer of software to link these systems, the knowledge worker is left with the possibility of linking the systems services based on his level of expertise. He then creates what is essentially an ad-hoc mashup with provided technologies, including sneaker-net and print-outs, in order to ensure the flow of information. It is evident that such an approach is expensive, prone to errors, hard to maintain and difficult to replicate; not to mention problematic from an IS security policy perspective.

We propose an architecture that formalizes the process of information dissemination within the enterprise by minimizing the effort of maintenance and interoperability. We identify the salient components of the process to sustain the aspect of information flow within the enterprise. The following are points of interest when considering this aspect:

- **Information**: domain schema, interoperability, composition and interchange
- **Information system**: standardized and secured communication channels at user, system and application level. Low maintenance cost, distributed database system.

Canopener successfully leverages the in-place components of the Operating System and installed user applications. This is achieved by means of pluggable components or scripts that serve a simple role within Canopener, namely to extract information from application and expose it as RDF markup. These plug-ins are matched to a specific application version tracked by the package manager and export the applications data as needed. A system-level plug-in exports OS user and group membership data after system maintenance has occurred.

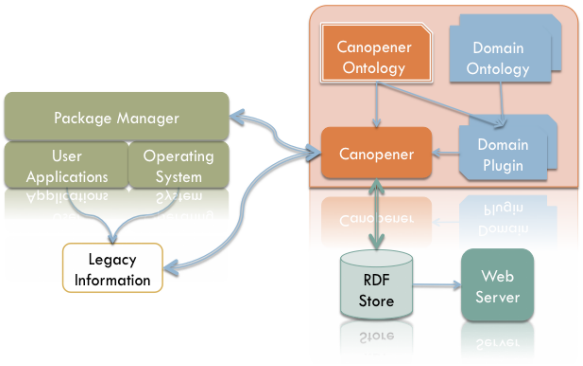

Figure 3: Canopener Mashup architecture

Figure 3 describes in detail the architecture and functionality implemented by Canopener. We chose to rely on RDF (Resource Description Framework) as the data interchange format given it's flexibility in expressing data as a graph. Some of the major benefits of RDF are:

- standard, expressive and open
- data interoperability
- schema unbound

Canopener provides information about it's mode of operation partially entailed within the Canopener ontology (Figure 4) while the rest is added as needed to the knowledge base under the form of domain specific RDFS [4] or OWL [5] based vocabularies. The complete ontology currently in use by Canopener can be downloaded from `http://www.ifi.uzh.ch/ddis/canopener/2010/02/spec#` in various formats.

Since the ontology alone cannot solve the problem entirely it is important to link a set of domain vocabularies with a

[4] `http://www.w3.org/TR/rdf-schema/`
[5] `http://www.w3.org/TR/owl-features/`

Figure 4: Canopener Mashup ontology snippet (RDFXML serialization format)

plug-in or set of plug-ins within Canopener. The plug-ins role is simple: gather legacy information *trapped* within the Operating System and transform it to RDF given a set of manually created mappings. The actual domain vocabulary is chosen by the plug-in itself as the Canopener container is generic. As an example, user and group membership information is exported as FOAF markup and document information is stored as Dublin Core markup. Such plug-ins can be implemented in their simplest of forms as collections of scripts (perl, python, bash, ...) that can get access to Systems resources and expose them as RDF. We ask the reader to recollect our previous mention of the knowledge worker trying to do just this using only the software provided on his desktop computer. Canopener provides a formal method to achieve this, by grouping functional blocks together via means of a plug-in manager component.

3.1 Process and Information flow

Canopener relies heavily on the Operating System's package manager. Currently Canopener is implemented as a APT package on the Debian [6] based operating systems such as Ubuntu, Linux Mint and the like, by being integrated with the Apt-Get [7] package manager. The mashup itself is deployed as a group of two interdependent debian packages. The advantage of relying on the Operating Systems package manager (or application installer) is many fold:

- the package manager acts as a the Canopener plug-in manager
- being already part of the technological stack it provides a number of features: automatic dependency resolution, global deployment (based on the adoption of the OS), update and version management support

We argue that by using the package manager in the mashup architecture both system administrators and users have more control over the mashups stability than web service based mashups. The information sources are local to the server, do not disappear at the whim of a third party and by virtue of mature version control, provide a certain guarantee over the data format of the sources. By contrast, XML or HTML scraping can fail silently when the data source changes without warning.

Lastly, there exists no 'data discovery' problem in this model as all instances of the applications are located by the package manager. There is no need to query a search engine for the data source location; the package manager is aware of the specific paths of all data files and can communicate them to the plug-ins as needed.

3.2 Mashing it all up and opening the can

The collected RDF data is imported into an RDF store also known as a triple store or quad store (if context is supported). We are using TokyoTyGR [8] as a process embedded RDF store which provides a high performance index of the stored RDF data according to the Hexastore [13] indexing paradigm while the data is publicly exposed on the Internet using the SPARQL protocol. A sample SPARQL query issued by Canopener could look like:

```
@PREFIX canopener:
<http://delphi.ifi.uzh.ch/ontologies/canopener/spec#> .
@PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .

SELECT ?location ?percent_load ?hour
WHERE
{
canopener:ddis_site a canopener:Site .
canopener:ddis_site canopener:systemload ?systemload .
?systemload canopener:percentload ?percent_load .
?systemload canopener:GMTHour ?hour .
canopener:ddis_site geo:lat_long ?location .
}
```

It would retrieve geographical location information associated with the system load at a given time for a specific `canopener:Site` [9] since Canopener reports a system load over a 24 hour time period by using the `canopener:GMTHour` predicate.

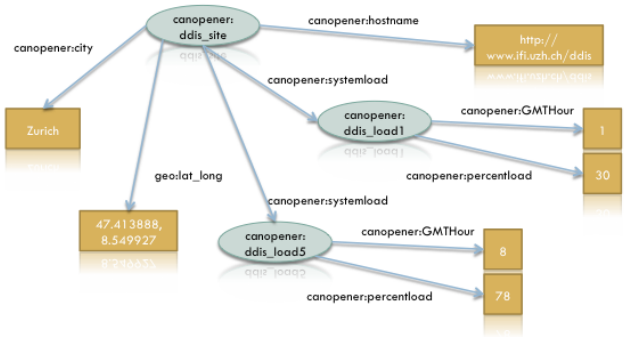

Figure 5: Sample Canopener RDF data

As can be seen in Figure 5 Canopener exports Linked RDF data physically distributed on the Web but logically interlinked within the same global RDF graph, contributing thus

[6] `http://www.debian.org/`
[7] `http://wiki.debian.org/Apt`
[8] An implementation of the Hexastore / TyGR RDF store
[9] As defined by the Canopener ontology

to the enrichment of the mashup space and the Linked Open Data [2] Semantic Web cloud. This is done by providing per system low-cost interchangeable information with high global value as demonstrated by the case studies presented next.

4. CASE STUDIES

There are several user-cases that motivated this research. Initially we had for an objective the enriching of existing documents with the semantics of the social web. However, we came to realize that it also provided a simple means of promoting knowledge exchange, the distribution of low cost public sensor information and the generation of data for further research. We review here each case in turn, along with its motivations and benefits.

4.1 Enriching the social web by recycling old data

The first was the enrichment of collaboration systems and of the social web to manage the multiple identities and accounts that academics with multiple appointments can have. FOAF [10] does provide as a markup that is capable of reconciling multiple accounts for the same individuals.

We wanted to be able to generate this cross account information automatically and were able to do so by creating a plug-in that extracts account information from SSH host and public key-lists. Similarly, friendships and IM account names are also extracted by plug-ins from the pidgin - libpurple applications. Obviously, a Canopener instance can only publish a partial graph: not all friendships are associative and some server accounts will have a limited amount of application data to link profiles with.

We note again that privacy and security are already provided by the OS in the form of file permissions. If the IM account, host file or documents can be read or accessed by a plug-in with no special access rights, then the information is assumed to be public. Should the users file permissions be restrictive, then the information is not published.

Figure 6 is a limited representation of the available data within an individual Canopener mashup. This representation is a high level abstraction of the full mashup which is available on the web and which is not reproduced here for space considerations. The friendships are reported using the Knows construct from the FOAF vocabulary based on both the users own FOAF reported friendships and the contracts that can be extracted from the IM configuration files of the individual users.

Here, we can see that explicit relationships have been reported between Cosmin ↔ Warren and Avi ↔ Reif. Group membership exist between two research units (seal, ddis) of the University of Zurich (UZH). Membership to individual research projects are also available for projects Canopener, CUMO, TyGR and Muninn. Note that user Warren has accounts in two different servers, both of which serve documents belonging to him.

This linked data permits the future implementation of smart retrieval: as the work load and location of the servers is made known by the mashup, it is possible to select the server that is the least busy or closets network-wise to fetch a document from. It also permits the rapid personalization of complex collaboration applications that can handle multiple

[10] http://xmlns.com/foaf/spec/

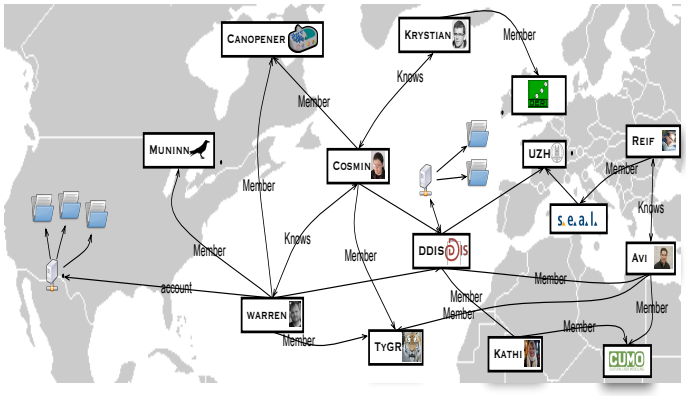

Figure 6: This mashup represents some of the data features of the exposed data.

content servers and team memberships. Similarly, the format lends itself well to support expert finding decisions by leveraging both the personal relationships of the individuals, his group memberships and his current ongoing projects.

4.2 Promoting knowledge exchange in semantically disconnected environments

This support for the markup of relationships, group memberships, document markups and multiple identities becomes an interesting repository of information useful to collaborative environments. The Canopener architecture can also be used within the intranet of an enterprise to promote the bottom-up dissemination of knowledge.

Most corporation will have a knowledge dissemination program or a project accounting system. Yet a lot of information is locked in a departmental or group server, segregated from all other systems. 'Stovepipe' or 'information silos' information barriers such as the ones represented in Figure 7 are not uncommon in most large organizations with information unable to cross functional boundaries.

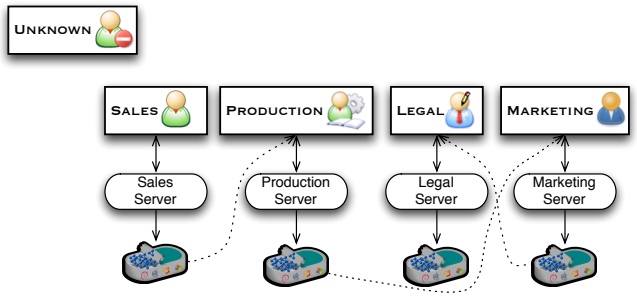

Figure 7: Canopener as an architecture makes it possible to access data within other functional lines using basic web standards.

Examples of this include smaller development projects that do not necessarily have the clout required to efficiently communicate their goals and objectives to the rest of the organization. The architecture enables these smaller departments to disseminate their information in a format easily integrated by desktop common tools while also leveraging

their existing organizational relationships, without requiring additional administrative effort.

While we had servers in mind for the deployment of Canopener, it is also possible to use the approach using individual workstations and laptops. While the machine community becomes that of a singleton, the relationships, accounts and documents remain linked across different computers and hence still enable collaboration.

4.3 The server as a sensor in a web mashup

The Canopener architecture is also a possible solution to the "sensing the real world" problem with respect to sharing opportunistic data by using current standards and technology stacks combined together according to the Mashup pattern. This enables us to expose low risk and low local-value information that is commonly trapped inside of one of the system applications.

The proposed architecture exposes the server as a sensor in a Global Sensor Network, by providing cheap local information such as system load, network traffic, public user information, and other relevant statistics. Although the exposed data is of low local value, globally it can be of potential great value as demonstrated by our application usage scenarios.

Such information can then be aggregated via means of tools such as SemanticWebPipes and exposed as a geo mashup, by providing geoIP information and power consumption under the form of a global consumption heat map overlaid on top of a GIS system such as Google or Yahoo Maps, with the possibility given a scalable data collection system of near real-time updates. We believe that such low value local information (usage statistics) collected and aggregated globally can be of very high value for the research community in various fields. The mashed up information can also be of high value to current cloud computing providers by acting as input to load balancing.

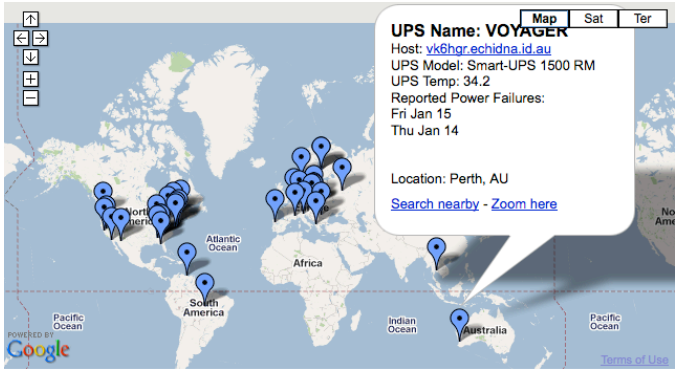

Figure 8: This mashup represents power failures reported by different servers on the web.

These cases are so far fairly localized in terms of the individuals and amount of data being processed. An application is the aggregate use of the information for situational awareness: Figure 8 is an example mashup that is supported by this approach.

We aggregate the information reported by the servers on their Uninterruptible Power Supply (UPS) and plot the reported power failures on a map. A prototype of this mashup is currently online [11] with a limited number of nodes. However as the number of Canopener nodes increases the coverage becomes pervasive and enables the tracking of power failures worldwide through the recycling and reuse of information already being collected.

4.4 Green computing

One of the major concerns today is the environmental impact of the IT sector, also known as Green computing or Green IT. The carbon footprint of modern day computers is increasing even-though there is a reported power per performance drop (Kw / MIPS). Some of the major power consumers in modern day desktop or laptop's are the GPU (Graphical Processing Units) which increase their performance according to Moore's Law. However, the power consumption itself is not dropping and there is a growing concern regarding the environmental impact of the IT infrastructure by both industrial and academic communities [5]. This concern has materialized in recent years under the name of Green Computing and it's main goal is to assess and reduce the impact the IT sector has on the environment on a global scale.

Hopper and Rice [8] have identified a number of Green Computing principles:

- Optimizing the digital infrastructure
- Sensing and optimizing the world
- Predictions on and reacting to a 'world' model
- Digital alternatives to physical alternatives

Measures are been taken by computing equipment manufacturers to reduce the negative impact computing has today on the World: environmental friendlier machines, renewable energy sources, data center placement near renewable energy or natural cooling locations, recyclable materials and advanced power management. Still the second principle is largely unmet at a global scale: we still do not know what is the power or usage profile of the current global computing infrastructure.

Canopener provides a platform for collecting important global usage statistics of servers world-wide by exposing such information via read-only SPARQL endpoints. As such, it is a first step in meeting items 2 and 3 of the principles proposed by Hopper and Rice.

We propose one Canopener usage scenario, that collects and exposes power consumption and usage statistics. The major benefit of using RDF as the data interchange format is the ability of mashup creators to use such information in a flexible and easy manner in conjunction with other Semantic Web data sources such as the Linked Open Data cloud [3].

As is, basic daily workload and hardware reporting is included as part of the Canopener ontology. This essentially provides basic reporting as to the computing capacity and its daily routine utilization. This taken together with any on-board sensor for temperature gives a gross profile of the actual power being used by the server and the amount of energy in use to cool it. On a global worldwide basis, the utility of such an aggregate 'computing-weather-report' could serve as a serious analysis tool for the goal of Green Computing.

[11] `http://www.dbdump.org/news/2010/01/tracking-power-failures-using-ups-status-pages.html`

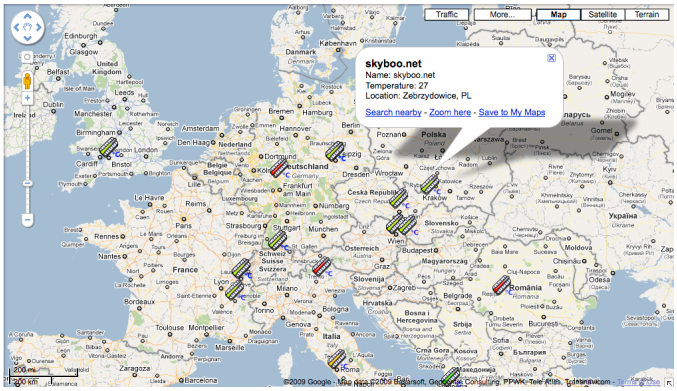

Figure 9: This mashup represents the ambient temperatures reported by different servers on the web.

Figure 9 is a mashup of the different temperatures of servers over central Europe in late January 2010. (The mashup is available on the web at [12]) It represents the type of information that can be extracted using the Canopener mashup on a larger scale, with each server becoming a public sensor of the global electrical load as a byproduct of performing its primary computing function.

5. PREVIOUS WORK

Most work has been done on mashup architecture, including specialized mashups for sensor networks [6]. Most recently there has been a move to micro-mashups for widget-side processing with the Mashlight [7] system. Similarly, attempts at improving security in multi-provider situations have been looked at within specialized environments such as the MashupOS[9] system. This is prompted by the increase in the generation of mashups for data exchange by the applications themselves, such as with the Drupal semantic web module [4].

The vast majority of data mashups today focus mainly on the following types of data sources: Web pages, XML/RSS/Atom feeds, SOAP/REST web services. The approach taken by Microsoft Popfly [13] and Marmite [14] focus mainly in establishing data flows from a collection of input sources augmented with web services. A similar approach is taken by Yahoo Pipes [11] that supports RSS feeds and web services as the input data sources for further transformation and customization. A semantic extension of Yahoo Pipes is represented by Semantic Web Pipes [10], that consider Semantic Web data sources as the input in producing the data mashup - such as SPARQL endpoints or publicly available RDF files. There is limited work with respect to mashing up legacy data sources and exposing them as standardized web services for other mashup systems present today.

The approach taken here is similar to the original screen-scraping approach where early mashup tools, would parse the HTML page of a particular web site to extract the embedded information, but it differs in a number of key aspects. Firstly the screen scraping mashups suffered mainly from increased instability since the model they adhere to cannot present any guarantees with respect to information type and layout stability. Web pages are subject to change according to the respective publishers agendas and are not regulated by a central authority as is the case of the package manager. The package manager enforces a level of stability via means of hard dependency resolution and long term major update cycle - as is the case for Ubuntu Linux which has a 6 months release cycle and long term support editions (LTS) have an even longer term cycle of release typically above 1 year. This facts guarantee that the application and system services targeted for information extraction change in a controlled manner and with a lower frequency than unregulated data sources on the web. Furthermore the package manager provides explicit version information so that the Canopener plug-in developer can actively choose to support or not certain versions of those particular services. Secondly screen scraping approaches employ a data discovery step that usually involves entities such as search engines or web directories to find the required information with a certain degree of uncertainty. This is not the case for Canopener since the package manager is a centralized location for application search and selection which is available globally across the OS distribution.

Extensive work on enterprise mashups has been conducted within the Damia [1] mashup platform considering the intranet as the deployment space of the mashup. Damia provides support for the so called *situational applications* - applications created by enterprise business users as a means of solving day-to-day problems. Although Damia employs a similar high level architecture as Canopener, it relies heavily on standard data interchange formats such as JSON and XML (RSS, ATOM) while Canopener is built on top of RDF for increased flexibility. The Damia Feed Server allows querying for information given a customized set based feed oriented protocol, as opposed to Canopener which exposes standard SPARQL query capabilities allowing expressive querying, while optimization is delegated towards the SPARQL processor level. Damia provides direct support for data sources composition, while this is not addressed in Canopener which concentrates instead on managing the availability of the data. At the same time Canopener has a lighter footprint with very few external dependencies, which lowers the cost of operations and installation.

6. CONCLUSION

In this paper we purpose a novel architecture that reused existing social and application information within servers to semantically augment the information being published. We do so in a manner that is low cost in that the existing administrative, policy and configuration interfaces are used to determine what content can and should be published. In this manner, we are able to promote the sharing of information previously locked within legacy applications while also supporting the opportunistic publication of publically useful data such as power and sensor information.

In future work, we will augment the relationship element of the FOAF markup using the FOAF relationship module[14], we are currently researching means to probabilistically assigning relationship types between users within the same server using an aggregate of the social information within the user account. Canopener has been targeted to

[12] http://www.dbdump.org/~warren/publications/heat_map.kml

[13] http://en.wikipedia.org/wiki/Microsoft_Popfly

[14] http://www.perceive.net/schemas/20021119/relationship/

the Debian Linux distribution and we are currently looking at porting it to other distributions and operating systems.

7. REFERENCES

[1] M. Altinel, P. Brown, S. Cline, R. Kartha, E. Louie, V. Markl, L. Mau, Y. H. Ng, D. Simmen, and A. Singh. Damia: a data mashup fabric for intranet applications. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 1370–1373. VLDB Endowment, 2007.

[2] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2009.

[3] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[4] S. Corlosquet, R. Delbru, T. Clark, A. Polleres, and S. Decker. Produce and consume linked data with drupal! *The Semantic Web - ISWC 2009*, pages 763–778, 2009.

[5] M. Garrett. Powering down. *Commun. ACM*, 51(9):42–46, 2008.

[6] D. Guinard and V. Trifa. Towards the web of things: Web mashups for embedded devices. In *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences)*, Madrid, Spain, Apr. 2009.

[7] S. Guinea, L. Baresi, M. Albinola, and M. Carcano. Mashlight: a lightweight mashup framework for everyone. In *2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009)*, 2009.

[8] A. Hopper and A. Rice. Computing for the future of the planet. In *Philos Transact A Math Phys Eng Sci*, volume 366, pages 3685–3697, 2008.

[9] J. Howell, C. Jackson, H. J. Wang, and X. Fan. Mashupos: operating system abstractions for client mashups. In *HOTOS'07: Proceedings of the 11th USENIX workshop on Hot topics in operating systems*, pages 1–7, Berkeley, CA, USA, 2007. USENIX Association.

[10] D. Le-Phuoc, A. Polleres, M. Hauswirth, G. Tummarello, and C. Morbidoni. Rapid prototyping of semantic mash-ups through semantic web pipes. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 581–590, New York, NY, USA, 2009. ACM.

[11] T. Loton. *Working with Yahoo! Pipes, No Programming Required.* Lotontech Limited, February 2008.

[12] U. Schonfeld and N. Shivakumar. Sitemaps: above and beyond the crawl of duty. In J. Quemada, G. León, Y. S. Maarek, and W. Nejdl, editors, *WWW*, pages 991–1000. ACM, 2009.

[13] C. Weiss, P. Karras, and A. Bernstein. Hexastore: sextuple indexing for semantic web data management. *Proc. VLDB Endow.*, 1(1):1008–1019, 2008.

[14] J. Wong and J. I. Hong. Making mashups with marmite: towards end-user programming for the web. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1435–1444, New York, NY, USA, 2007. ACM.