Design and Implementation Mobile Data Collection Using Mobile Proxy

Aris Prawisudatama
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
Email: soedomoto@gmail.com

I Gusti Bagus Baskara Nugraha
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
Email: baskara@stei.itb.ac.id

***Abstract*—Data collection using mobile device is something that is quite common recently. However, most of those applications are using static validation rules. The use of mobile proxy to duplicate rule and data is believed could improve the flexibility and implementation of the validation rule. Rule could be easily designed in modular basis, installed, updated, as well as deleted from application using OSGi framework. This paper is aimed to research about the design and implementation of mobile proxy that could be used to implement dynamic validation rule on SOA-based mobile device.**

I. Introduction

Data collection is one of the main tasks of statistical institution in the world, including BPS-Statistics Indonesia. Data collection process is conducted using various media, ranging from conventional method, such as paper questionnaire to the modern method using mobile device. Until now, most of data collection conducted by BPS-Statistics Indonesia are still using paper questionnaire. However, BPS is now starting using mobile device for collecting data.

In data collection, the most important thing to be considered is the data quality. Data quality is measured using consistency and validation parameter. Data collection using paper questionnaire still rely on humans carefulness and thoroughness in implementing the rule. Using mobile device has a benefit related to consistency and validation. Thus, generally, using mobile device could increase data quality.

The scenario that is commonly used in the implementation of mobile device application is to wrap it in a package including GUI and workflow. This scenario has a weakness, that workflows are used are static and cannot be added, removed, or updated, except by updating the whole application.

Another scenario which is also commonly used is by adopting a Service-Oriented approach. Service oriented approach basically separated into two components: the service provider (server) and a service consumer (client). In service-based approach, the workflow will be executed on the server, while the client access the workflow through a network. Service-based approach has been widely implemented, including implementation on mobile based applications. However, the implementation of service-based approach to the mobile device has several potential obstacles, mainly due to the availability of the Internet network, so mobile device that acts as a client cannot always connect to the server. No connection of mobile devices on the server can be categorized into three types [1]:

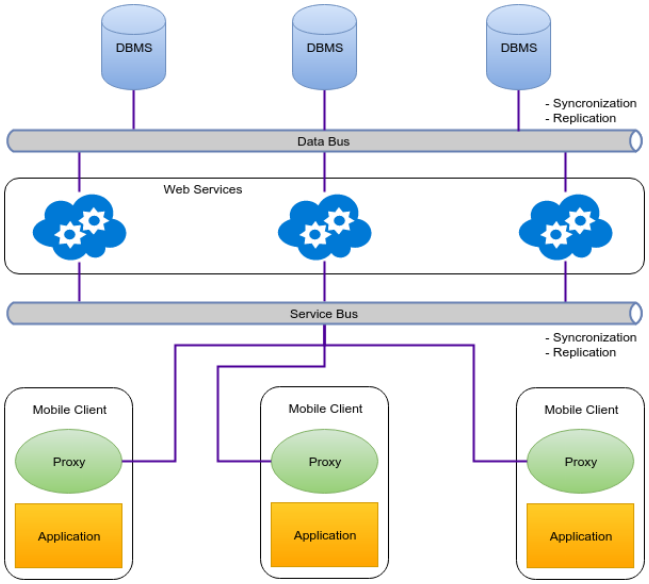


Fig. 1. Proxy Overview

- **Delay-based interruption:** delays in message delivery because of problems that occur between the sender and receiver,
- **Network outage:** a condition where a node is disconnected from the other nodes. Cause outage network include: network failure, mobile devices moving between zones, laptops going to sleep, unplugged cables, or inadvertent closing of the application,
- **Explicit departure:** a condition in which the user explicitly logs out of the session or quits the application.

The proposed alternative scenario is to adopt a proxy approach. Proxy approach basically is to put a server between the service provider and the service consumer that will intercept the request from the client to the server. Proxy can act as a forward proxy in a condition connected to the network, or cache server in conditions not connected with the network. Illustration of proxy will be designed can be seen in Fig.1.

This paper will examine how to design a proxy that can

be used in a condition connected or disconnected, which is divided into five sections. Section 1 is the background of this research. While the second section will discuss the literature review and research related to the design of the proxy. Section 3 will discuss the proposed design, followed by a discussion in section 4. Finally section 5 concludes the paper with a summary and a discussion of future work.

II. Theoretical Background

In general, the proxy can be said as a placeholder for other entities [2]. Due to its capability to intercept communication source and target, it allows deferring the initialization of the target, redirecting the communication target to one or more entities, performing additional capabilities before/after communicating with the target, etc. [3].

Implementation of the proxy have been conducted with various purposes, e.g., for memory efficiency [2], remote communication [4], and concurrent data evaluation/processing (e.g. for future object) [5]. Proxies have also been used in component-based middleware such as EJB [6], to provide various enterprise-level capabilities. Furthermore, proxies have also been widely adopted in contemporary work on application adaptation [7] [8] [9] [10] [11] [12], since they allow certain components of an application to dynamically reconfigure themselves (e.g. changing its behavior) transparently, i.e. without the awareness of the rest of the components in the application. While on the mobile device, the proxy is used to bridge the wired-wireless gap, and make all mobility, connectivity and context-dependent issues transparent to the application developer [9].

To support the mobility of a mobile device, the proxy must also be portable and dynamic. Rubinsztejn et al make the architectural design of the proxy in the form of middleware, which consists of both server and client API [9] called MoCA. MoCA architecture consists of a monitor and several services. Monitor is a daemon or service that runs on each mobile client to collect and submit data to the services that run on the server side.

Another approach in the design of proxy proposed by Cobarzan et al. To support the dynamism of proxy, they composed proxy that consists of two components, the daemon and the dispatchers [13]. Dispatcher is a component that acts as proxy. While Dispatcher is used to handle incoming client request, then forward them to the server or to other dispatchers, process it itself, or reject it. Meanwhile, the daemon is a component that runs automatically in the background and is used to monitor the code that contains the command to enable dispatchers. So that the proxy is not automatically activated by the daemon is active before, contrasts with the MoCA which automatically activates the proxy.

Proxy has several functions, such as content adaptation, protocol translation, user authentication, handover management, and cache management [9]. Cache has an important role in reducing latency and network traffic between server and client [14]. Caching can be implemented in several locations: at the client [15] [16], server [17] [15], and within the network [18] [16] [19] through a proxy server.

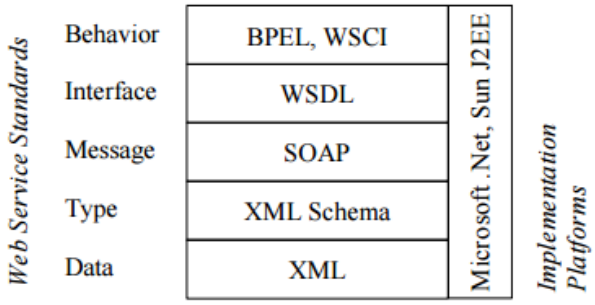


Fig. 2. Web Service Standards Stack

Caching is generally implemented on a web the document to store frequently accessed parts of the document [20] [21] [19], either dynamic or static web contents, although not uncommon also implemented at the application level [7] [8] [9]. Since the cache is a temporary object [22] that represents the data on the server, it needs a method to ensure the consistency of data stored in the cache and which are contained in the server.

Caching method in a distributed system, including on a mobile devices, have a different design with a centralized system. Takdir et al proposed a method of proxy, by caching both data and workflow that will be used by the composite application on the local storage [23]. This method provides a transparent resources (i.e. the data and web service logics) access using combination of synchronization, replication, and routing mechanism.

Meanwhile, Terry et al using XML web services caching method [24] that mimics the behavior of Web services to a limited extent. This XML cache is transparent either on the client side and the server component on a Web services. This method is able to handle disconnections for each Web services. However, this method is lack of consistency, so that an operation performed by the local user could change some of the results of the earlier requests that are stored in the cache.

In a workflow-based application, such as the application of data collection, in which workflows are defined, managed, and executed in accordance with the sequence of logic [25], one of the standard that is now being widely used is the Business Process Execution Language (BPEL) [26] [27], BPEL is built on the Web Service (WS) standards and Provides a recursive aggregation models for Web services [28] and used as glue between interacting services [27]. BPEL drives communication of web services in order to deliver business services that Contain several interconnected processes. Web services address interoperability issues in service level, whereas BPEL orchestrate web services in process level. BPEL is ideally suited to the service-oriented architecture, a set of guidelines for integrating disparate systems by presenting each system as a service that implements a specific business function [29].

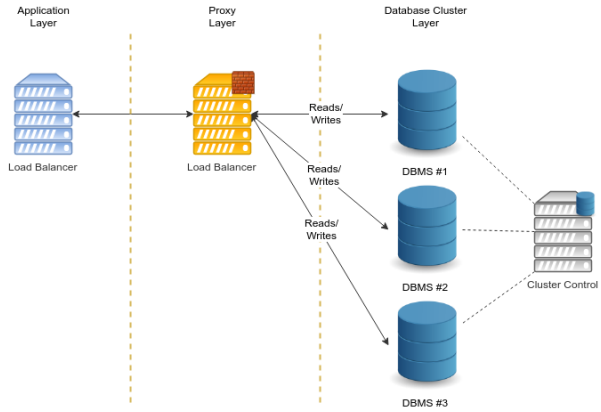
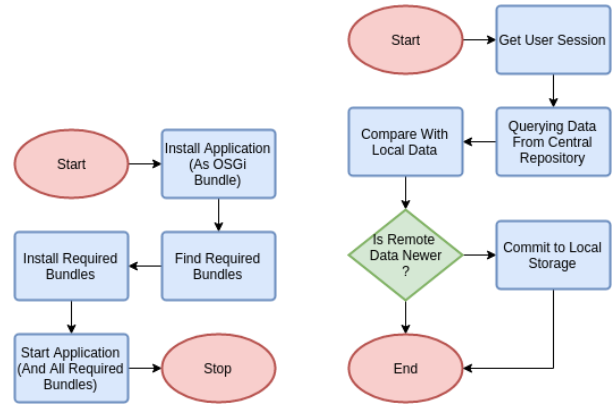


Fig. 3. Data Bus Layers



(a) Web Services Replication Strategy

(b) Data Replication Strategy

Fig. 4. Replication Strategies

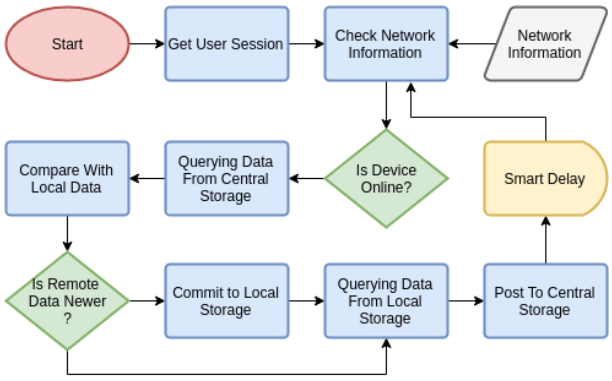


Fig. 5. Data Syncronization Strategy

III. Mobile Proxy Design

A. Design Overview

Proposed proxy mobile design consists of three components, data, services, and applications, which are connected via two buses, the data bus and bus service. Illustration of design can be seen in Fig.1.

Data bus Provides transparent access to the data source for web services. The data bus consists of two layers, ie cluster database layer and proxy layer. The cluster database layer consists of several database nodes that are equipped with a control cluster to ensure data synchronization. Meanwhile, the proxy layer consists of a proxy server that acts as a load balancer for database nodes. Load balancer is equipped with a healthy check to check the availability of each database node.

Similar with the data bus, the service bus handle web service invocation by the application. Web services that are representation of the rules, are packed as a bundle. Then the bundles are replicated, by composite application, from central repository to local repository. The bundles that are available in local repository will be installed and started dynamically by composite application. Moreover, installed bundles can also be stopped, removed, and updated easily.

B. Replication

Data and web services bundles are replicated from central storage to local storage as cache objects. Firstly, composite application replicate web services bundles from central repository to local repository. Each web services bundle then will replicate data that it is depend on, from central database into local storage. Fig.4 shows web services and data replication strategy.

C. Synchronization

Synchronization ensure the data and web service bundles are consistent between local and central repository. Web service bundle must implement versioning, so central server and proxy server must install bundles with same version to ensure the same logic is used. Composite application will periodically execute version check of the bundles, and replicate it when new versions are available in central repository.

Data synchronization is very depend on bundle, since each bundle require different data. Data will be synchronized in two way, from central database to local cache storage (e.g. master data) and from local cache storage to central storage (e.g. transaction data). Fig.5 shows data syncronization strategy.

D. Routing

Routing plays important role in distributed system. It manages the network traffic and point requests to appropriate destination [23]. Local web services are act as proxy server that intermediate composite application, that act as client, and central server. Proxy server will decide whether to forward request to central server or reply it on behalf of central server. Fig.6 shows routing strategy.

IV. Implementation

A. OSGi Framework

There are several implementations of OSGi, or commonly referred to OSGi Framework, ie: Apache Felix, Concierge, Equinox Eclipse, JBoss, Hitachi, Knopflerfish, and ProSyst. This study uses the Knopflerfish OSGi Framework, because

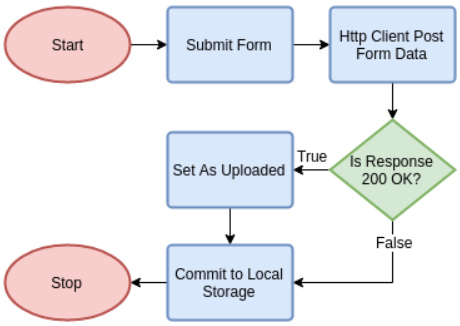
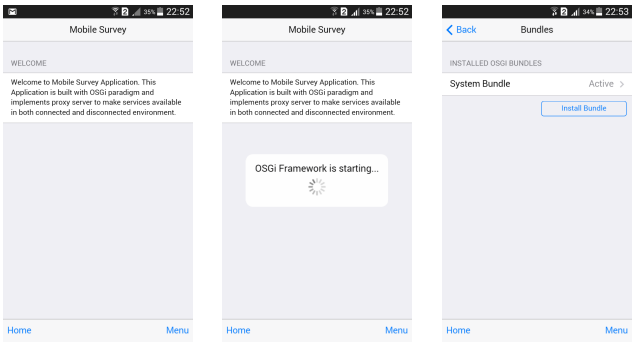
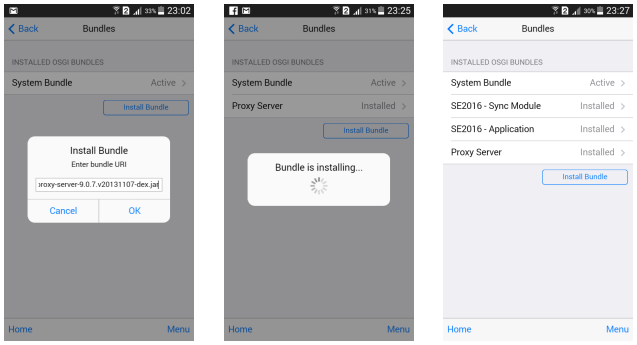


Fig. 6. Routing Strategy

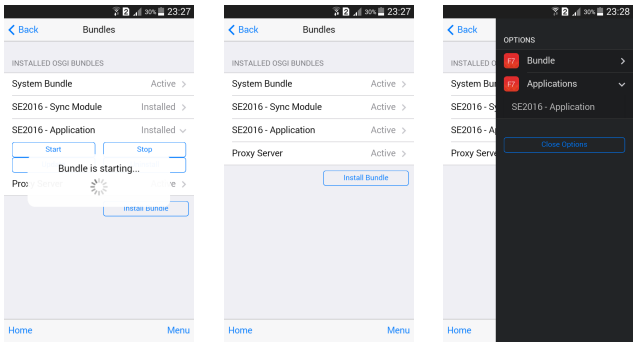


(a) Home Page (b) Autostart OSGi Framework (c) Bundle Page

Fig. 7. Android Application



(a) Install bundle (b) Bundle is installing (c) Bundles are installed



(d) Bundle is starting (e) Bundles are active (f) Composite application is loaded

Fig. 8. Android Bundles

Knopflerfish OSGi Framework has implemented the latest OSGi specification, the OSGi R6. In addition, the Knopflerfish OSGi Framework is also proven to be implemented in Android application.

B. Android Application

By default, Knopflerfish provides an implementation of the OSGi Framework that can be run on any operating system that supports Java SE. However, because the Android OS uses a different JVM implementations, the Dalvik VM, so it needs to be made Knopflerfish OSGi Framework implementations that can run on the Android OS.

Android application that is made must support the principle of OSGi, which is modular. The case studies used in this study is a case study for field data collection, so that the user interface is a questionnaire form. To accelerate the creation and simplify the loading of the page, so it is used Cordova. Cordova is a webview-based, so the control used is an HTML form element. Fig.7 shows screenshots of Android application.

C. OSGi Bundles

1) Proxy Server: Proxy server is an OSGi bundle that implements an embedded web server. Web server used in this study is Jetty 9 which is the highest major version available today. In the Proxy Server Bundle registered a service (API) ContextHandlerService, which is used to register the ServletContext initiated on Application Bundle.

2) Composite Application: Composite Application is an OSGi bundle that implements web services and clients in a same bundle. Web services are implemented in the form of Servlets collected in a single context. While the client is implemented in the form of a web-based application, written in HTML, Javascript, and CSS. In addition, the Composite Application also provides an API that can be imported dynamically by another bundle, ie: AccountHandlerService, DaoHandlerService, and PropertyHandlerService.

3) Synchronizer: Synchronizer is an OSGi bundle that is used to synchronize data in two directions: Remote to Local and Local to Remote. Synchronizer bundle requires some service provided by Composite Application, ie: AccountHandlerService to access the session to be synchronized and DaoHandlerService to access local storage.

Fig.8 shows screenshots of OSGi bundles installed and started in Android environment.

D. Remote Server

As described in IV-B, Knopflerfish by default already provides an implementation of the OSGi Framework that can be run on any operating system that supports Java SE. Remote server running Knopflerfish using minimal bundle, and installed the same module with the composite application. Listing.1 and Listing.2 shows screenshots of OSGi bundles are active in remote server.

```
java -jar framework.jar -xargs minimal.xargs
Knopflerfish OSGi framework launcher, version
 ↪  <unknown>
Copyright 2003-2015 Knopflerfish. All Rights
 ↪  Reserved.
See http://www.knopflerfish.org for more
 ↪  information.

Created Framework:
 ↪  org.knopflerfish.framework,
 ↪  version=7.2.0.
> Framework launched
```

Listing 1: Knopflerfish with minimal bundles

```
> install http://knopflerfish.soedomoto.tk/
 ↪  jars/pgfw7/proxy-server-9.0.7.v20131107-
 ↪  dex.jar
Installed: Proxy Server (#10)
> start 10
2016-09-05 21:57:02.294:INFO:oejs.Server:
 ↪  BundleStart #10: jetty-9.0.z-SNAPSHOT
2016-09-05 21:57:02.455:INFO:oejs.
 ↪  ServerConnector: BundleStart #10:
 ↪  Started ServerConnector@698b6705
 ↪  {HTTP/1.1}{0.0.0.0:5555}
Started: Proxy Server (#10)
> install http://knopflerfish.soedomoto.tk/
 ↪  jars/pgfw7/se2016-bundle-0.1-SNAPSHOT-
 ↪  dex.jar
Installed: SE2016 - Application (#11)
> start 11
...
2016-09-05 21:57:33.147:INFO:oejsh.
 ↪  ContextHandler:BundleStart #11: Started
 ↪  o.e.j.s.ServletContextHandler@390161c7
 ↪  {/se2016,null,AVAILABLE}
Started: SE2016 - Application (#11)
> install http://knopflerfish.soedomoto.tk/
 ↪  jars/pgfw7/se2016-sync-bundle-0.1-
 ↪  SNAPSHOT-dex.jar
Installed: SE2016 - Sync Module (#12)
> start 12
Started: SE2016 - Sync Module (#12)
...
```

Listing 2: Install and start required bundles

V. Testing

A. Response Time

Pengujian terhadap response time digunakan untuk mengukur latency antara client dan server. Pengujian ini menggunakan dua skenario, yaitu skenario tanpa proxy dan dengan proxy. Dari empat kali pengujian dengan jumlah transaksi yang bervariasi, seperti yang ditunjukkan dalam Table I, menunjukkan dengan menggunakan proxy dapat mengurangi response time sebanyak 20 kali lipat dibandingkan dengan tanpa menggunakan proxy.

TABLE I
Response Time Test (ms)

Transactions	Without Proxy			With Proxy		
	Total	μ	σ	Total	μ	σ
11	4226	384.18	77.10	862	78.36	224.09
110	41999	381.81	51.54	1376	12.51	9.41
1100	405892	368.99	59.95	15891	14.45	13.47
5500	2226949	404.90	206.73	64600	11.74	9.92

VI. Conclusion and Future Work

We propose a mobile proxy design that is dynamic, which can be distributed separately with composite applications and its synchronization. Distribution of proxy server, composite applications, and synchronization is done in the form of OSGi bundle. By adopting the principles of OSGi, synchronization workflows in the form of web services proved to be more effective, because each bundle can be updated independently, either manually or automatically.

By adopting the synchronization in a separate module also proven to be able to synchronize with better and more controlled because it can be start-stop and in every moment. Synchronization modules can also be updated, either manually or automatically, without affecting the other modules.

Our proposed design also makes it possible to install more than one composite applications and synchronization in an application with just a single proxy server.

References

[1] C. Gutwin, T. N. Graham, C. Wolfe, N. Wong, and B. de Alwis, "Gone but Not Forgotten: Designing for Disconnection in Synchronous Groupware," in *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '10. New York, NY, USA: ACM, 2010, pp. 179–188. [Online]. Available: http://doi.acm.org/10.1145/1718918.1718951

[2] E. Gamma, *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[3] H. Gani and C. Ryan, "Improving the Transparency of Proxy Injection in Java," in *Proceedings of the Thirty-Second Australasian Conference on Computer Science - Volume 91*, ser. ACSC '09. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2009, pp. 55–64. [Online]. Available: http://dl.acm.org/citation.cfm?id=1862659.1862669

[4] M. J. Wilson, "Get smart with proxies and RMI," Nov. 2000. [Online]. Available: http://www.javaworld.com/article/2076234/soa/get-smart-with-proxies-and-rmi.html

[5] P. Pratikakis, J. Spacco, and M. Hicks, "Transparent Proxies for Java Futures," in *Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, ser. OOPSLA '04. New York, NY, USA: ACM, 2004, pp. 206–223. [Online]. Available: http://doi.acm.org/10.1145/1028976.1028994

[6] "Fusion Middleware Understanding Oracle WebLogic Server." [Online]. Available: https://docs.oracle.com/cd/E24329_01/web.1211/e24446/ejbs.htm#INTRO255

[7] O. Holder, I. Ben-Shaul, and H. Gazit, "System support for dynamic layout of distributed applications," in *Distributed Computing Systems, 1999. Proceedings. 19th IEEE International Conference on*. IEEE, 1999, pp. 403–411. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=776542

[8] M. Philippsen and M. Zenger, "JavaParty - Transparent Remote Objects in Java," *Concurrency Practice and Experience*, vol. 9, no. 11, pp. 1225–1242, 1997. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.6874&rep=rep1&type=pdf

[9] H. K. Rubinsztejn, M. Endler, and N. Rodriguez, "A framework for building customized adaptation proxies," in *Intelligence in Communication Systems*. Springer, 2005, pp. 1–11. [Online]. Available: http://link.springer.com/chapter/10.1007/0-387-32015-6_1

[10] C. Ryan and C. Westhorpe, "Application adaptation through transparent and portable object mobility in java," in *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*. Springer, 2004, pp. 1262–1284. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-540-30469-2_29

[11] M. Tatsubori, T. Sasaki, S. Chiba, and K. Itano, "A Bytecode Translator for Distributed Execution of Legacy Java Software," in *Proceedings of the 15th European Conference on Object-Oriented Programming*, ser. ECOOP '01. London, UK, UK: Springer-Verlag, 2001, pp. 236–255. [Online]. Available: http://dl.acm.org/citation.cfm?id=646158.680014

[12] E. Tilevich and Y. Smaragdakis, "J-Orchestra: Automatic Java Application Partitioning," in *Proceedings of the 16th European Conference on Object-Oriented Programming*, ser. ECOOP '02. London, UK, UK: Springer-Verlag, 2002, pp. 178–204. [Online]. Available: http://dl.acm.org/citation.cfm?id=646159.680022

[13] C. Cobrzan, "Dynamic Proxy-Cache Multiplication Inside LANs," in *Euro-Par 2005 Parallel Processing*, ser. Lecture Notes in Computer Science, J. C. Cunha and P. D. Medeiros, Eds. Springer Berlin Heidelberg, Aug. 2005, no. 3648, pp. 890–900, dOI: 10.1007/11549468_97. [Online]. Available: http://link.springer.com/chapter/10.1007/11549468_97

[14] L. Rizzo and L. Vicisano, "Replacement policies for a proxy cache," *IEEE/ACM Transactions on Networking (ToN)*, vol. 8, no. 2, pp. 158–170, 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=336116

[15] A. Bestavros, R. L. Carter, M. E. Crovella, C. R. Cunha, A. Heddaya, and S. A. Mirdad, "Application-level document caching in the internet," in *Services in Distributed and Networked Environments, 1995., Second International Workshop on*. IEEE, 1995, pp. 166–173. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=470449

[16] P. Cao and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms." in *Usenix symposium on internet technologies and systems*, vol. 12, 1997, pp. 193–206. [Online]. Available: http://static.usenix.org/publications/library/proceedings/usits97/full_papers/cao/cao.pdf

[17] M. F. Arlitt and C. L. Williamson, "Trace-driven simulation of document caching strategies for internet web servers," *Simulation*, vol. 68, no. 1, pp. 23–33, 1997. [Online]. Available: http://sim.sagepub.com/content/68/1/23.short

[18] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, and E. A. Fox, "Caching proxies: Limitations and potentials," 1995. [Online]. Available: http://eprints.cs.vt.edu/archive/00000427/

[19] M. Busari and C. Williamson, "On the sensitivity of web proxy cache performance to workload characteristics," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2001, pp. 1225–1234. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=916617

[20] A. Mahanti, D. Eager, and C. Williamson, "Temporal locality and its impact on Web proxy cache performance," *Performance Evaluation*, vol. 42, no. 2, pp. 187–203, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166531600000328

[21] J. Shim, P. Scheuermann, and R. Vingralek, "Proxy cache algorithms: Design, implementation, and performance," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 4, pp. 549–562, 1999. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=790804

[22] B. D. Davison, "A survey of proxy cache evaluation techniques," in *Proceedings of the Fourth International Web Caching Workshop (WCW99)*. Citeseer, 1999, pp. 67–77. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.41.8087&rep=rep1&type=pdf

[23] Takdir and A. I. Kistijantoro, "Multi-layer SOA implementation pattern with service and data proxies for distributed data-intensive application system," in *2014 International Conference on ICT For Smart Society (ICISS)*, Sep. 2014, pp. 37–41.

[24] D. B. Terry and V. Ramasubramanian, "Caching XML Web Services for Mobility." *ACM Queue*, vol. 1, no. 1, pp. 70–78, 2003. [Online]. Available: http://queue.acm.org/detail.cfm?id=864024

[25] M. Wieland, K. Gorlach, D. Schumm, and F. Leymann, "Towards reference passing in web service and workflow-based applications," in *Enterprise Distributed Object Computing Conference, 2009. EDOC'09. IEEE International*. IEEE, 2009, pp. 109–118. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5277706

[26] C. Pautasso, "RESTful Web service composition with BPEL for REST," *Data & Knowledge Engineering*, vol. 68, no. 9, pp. 851–866, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169023X09000366

[27] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson, *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. Prentice Hall PTR, 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1051879

[28] J. Pasley, "How BPEL and SOA Are Changing Web Services Development," *IEEE Internet Computing*, vol. 9, no. 3, pp. 60–67, May 2005. [Online]. Available: http://search.proquest.com.proxy.lib.odu.edu/docview/197334619/abstract/5DFF699B3135476BPQ/1