Securing e-Loyalty Currencies

Aspen Olmsted
Department of Computer Science College of Charleston
Charleston, SC 29401

Abstract

In this paper I investigate the problem of providing e-commerce security constraints in distributed cloud systems while maintaining concurrency requirements. This study analyzes e-loyalty currencies used by e-commerce systems as a motivating example. I investigate loyalty incentives rewarded through electronic points and electronic currency programs. The hypothesis tested is if the modeled system can harden the surface from cyber-attack. I consider five loyalty activity categories rewarded by companies to their patrons; social networking rewards, web-site browsing rewards, mobile browsing rewards and referral/social circle rewards. I document surface area vulnerabilities with each loyalty category, propose and implement a solution that will validate the activity and administer the e-loyalty currency properly.

1. Introduction

A business to consumer enterprise spends a great deal of money on customer acquisition. According to KISSmetrics [1], a company can pay seven times the amount on acquiring a new customer over what it would cost for retention of a current customer. In an attempt to retain customers, some businesses have begun to implement customer social networking loyalty programs that reward customers with electronic currencies. A social networking loyalty program has two possible aims. One is to encourage their current customers to interact with the business' digital assets in an attempt to strengthen the relationship between the company and the patron. The second aim is to leverage the current customer's relationships and social clout to acquire new customers from the current customer's social circle.

Social clout is very valuable to a company. Valuable enough that a business, Klout.com, has started that offers an algorithm to rate individual's social clout. Klout uses Bing, Facebook, Foursquare, Google+, Instagram, LinkedIn, Twitter, and Wikipedia data to create Klout user profiles that are assigned a unique "Klout Score". [2] Klout scores range from 1 to 100, with higher scores corresponding to a higher ranking of the breadth and strength of one's online social influence. Recently, American Airlines offered a one-day pass to individuals with a Klout score of 55 or higher for American Airlines' Admirals Club in 40 different airports, regardless of whether they're booked on one of the carrier's flights. The admission includes first-class benefits like free Wi-Fi and beer.

In my previous work [3] [4] I provided an extension to the lazy replica update propagation method to reduce the risk of data loss and provide high availability while maintaining consistency. The Buddy System executes a transaction on the primary replica. However, the transaction cannot commit until a secondary cluster, "the buddy", also preserves the effects of the operation. The rest of the clusters are updated using one of the standard lazy update propagation protocols. This architecture allows the Buddy System to guarantee transactional durability. The effects of the transaction are preserved even if the server hosting the primary replica crashes before the update can be propagated to the other replicas. It also provides efficient update propagation (i.e., my approach requires the synchronized update between two replicas only, therefore adding minimal overhead to the lazy-replication protocol).

The Buddy System uses an application-layer dispatcher [5] to select the buddies based on the data items and the operations of the transactions, the data versions available, and the network characteristics of the WS farm. A limitation of the Buddy System is there is no way to add end-user configurable constraints at implementation time without either sacrificing consistency or requiring a recompile of the client application. In past work, I addressed issues related to maintaining high availability while adding guarantees of correctness by enforcing hierarchical constraints [5]. Traditionally these hierarchical constraints are not enforced by the system due to the expensive run-time cost. The Buddy System materializes the data required to apply the restriction without requiring navigation of the hierarchical relationship.

In this paper, I investigate the problem of enforcing application domain constraints in a distributed architecture while maintaining high availability. To do this, I investigate eCurrency and loyalty rewards programs and extend the Buddy System by adding constraints that apply the e-Currency reward rules in a secure way to ensure they are rewarding the activity the company wants to reward. The Buddy System is enhanced to allow for application domain constraints. The application domain specific constraints defend a system from cyber-attack by implementing the intent of the e-Currency program.

2. Social Networking Rewards

Social networking rewards are activities where a patron is incentivized to share information about a business or interaction with a company to their social circles. The business is interested in leveraging the patron's social clout to gain greater brand awareness. I studied three loyalty programs that included social networking rewards:

1. Marriott™ Hotels - Marriott operated a beta promotion called "Marriott Rewards - Plus Points." Participants could earn up to 2,000 Marriott Points each month by participating in social media applications like Facebook, Twitter, Instagram, and Foursquare. [6] The Marriott Rewards' activities are micro-blogs on Twitter using a specific hashtag of #MRpoints. The reward for the micro-blog post or micro-blog re-post is twenty-five points. The currency value of each post is valued at around $0.42. They cap the daily value of all micro-blog posts at one hundred points. There are no constraints placed on the tweets semantics, timing or the validity of the Twitter account.
2. JetBlue Airways – JetBlue is commercial airline operating in the United States. JetBlue operates a reward program named TrueBlue. Patrons earn badges and points for flying on JetBlue and partner airline flights. The program encourages patrons to share their accomplishments on Facebook and Twitter to earn extra badges.[7] The airlines program is different in a few ways; there is automation of the instantiation of the micro-blog to force the post to include links to the program, they reward badges as a middle layer to earning loyalty currency, and they utilize micro-blogging on Facebook as well as Twitter.
3. Social Rewards – Social Rewards is a Social Media based loyalty marketing program that rewards consumers for engaging in social media activity such as watching YouTube videos, brand mentions via Twitter, Instagram and Facebook fan activity. [8]

2.1. Modeling

I considered three types of validation requirements: the blogs semantics, the timing of the blog and the validity of the bloggers account.

To evaluate the semantics of the micro-blogs, I pulled a sample of the tweets consisting of a day's worth of the Marriott Rewards micro-blogs. The sample consisted of 3,982 unique micro-blogs including the tag and 1,474 re-posts of these blogs. Only five percent of the blogs included reference to a specific hotel. My JetBlue sample consisted of a week's worth of micro-blogs from Twitter that used the tag #TrueBlueBadges. It turned out participation was very low. The sample only consisted of twenty-eight posts and zero re-posts (JetBlue did not reward badges for retweets). JetBlue provides a template when the micro-blog is created. Unfortunately, none of the posts included additional data from the link template created by the JetBlue website.

To evaluate the timing of the micro-blogs I used the same samples from above. For the Marriot program, I compared the first and last post on a specific day for each user. Eighty-two percent of the users in the sample who posted multiple micro-blogs did so in a timespan of fewer than five minutes. For JetBlue, I wanted to understand why participation was so low I schedule a job that pulled the tweets every minute for a weeks' time. To my amazement, sixty-three percent of the posts were deleted shortly after the post. Unlike the Marriott program that rewarded the points at a later point in the day, the JetBlue program would reward the credit immediately at the time of the posting. My new sample data consisted of ninety-seven original posts. Sixty-one were deleted, and twenty-one were part of batches of posts done in less than five minute time periods. I concluded that only fifteen percent were valid based on the timing activities.

To evaluate the validity of the micro-blog user accounts I used the same samples from above. For each poster account, I examined the activity on their Twitter account. I defined a valid account as one where the tweets represented less than twenty percent of the tweets. Seventy-four percent of the user accounts in the sample were classified as not valid in my study. I attribute this low percentage of validity to the ability of a user to clean their Twitter account immediately after a post and still earn the reward.

3. Web-Site Browsing Rewards

Web-Site browsing rewards are activities where a patron is incentivized to visit a business' web assets. The business is interested in creating habits in the patron's internet browsing activities. I studied two loyalty programs that included web-site browsing rewards:

1. Microsoft™ Search Engine - Bing Rewards lets you earn credits for searching on Bing or trying newfeatures from Bing or other Microsoft products and services. Bing Rewards credits can be redeemed for a variety of gift cards and other rewards. [9]
2. JetBlue Airways – JetBlue's TrueBlue program was referenced in earlier rewards activity categories. Patrons can also earn badges by visiting their TrueBlue page. [7]

To evaluate the validity of the browsing loyalty programs data, I automated a series of visits, to the web assets referenced earlier, using the iMacros plugin for the Firefox web browser. [7] The scripting tool allows a combination of JavaScript and custom


```
1 var i, j;
2 var macrolist = new Array();
3 macrolist.push("bing/user1.iim");
4 macrolist.push("bing/user2.iim");
5 macrolist.push("bing/user3.iim");
6 macrolist.push("bing/user4.iim");
7 macrolist.push("bing/user5.iim");
8 macrolist.push("bing/user6.iim");
9 macrolist.push("bing/user7.iim");
10 macrolist.push("bing/user8.iim");
11 for (i = 0; i < macrolist.length; i++) {
12     retcode = iimPlay("bing/bingsignout.iim");
13     retcode = iimPlay(macrolist[i]);
14     iimPlay(pausemacro);
15     for(j = 0; j < 30; j++) {
16       iimPlay("bing/bingsearch.iim");
17     }
18     for(j = 0; j < 15; j++) {
19       iimPlay("bing/mobilebingsearch.iim");
20     }
21 }
22 iimPlay("bing/bingsignout.iim");
```

Algorithm 1. Script to Iterate Through Users

```
1 VERSION BUILD=8890130 RECORDER=FX
2 TAB T=1
3 SET !USERAGENT "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:35.0) Gecko/20100101 Firefox/35.0"
4 URL GOTO=https://www.bing.com/search?q=georgia&qs=n&form=QBRE&pq=georgia&sc=8-7&sp=-1&sk=&cvid=b94a6da1f4004670b5cfa65adaf95bfc
5 TAG POS=1 TYPE=INPUT:SEARCH FORM=ID:sb_form ATTR=ID:sb_form_q CONTENT=In<SP>history<SP>{{!NOW:ss}}
6 TAG POS=1 TYPE=INPUT:SUBMIT FORM=ID:sb_form ATTR=ID:sb_form_go
```

Algorithm 2. Script to Search Bing

```
1 VERSION BUILD=8890130 RECORDER=FX
2 TAB T=1
3 SET !USERAGENT "Mozilla/5.0 (Android; Mobile; rv:22.0) Gecko/22.0 Firefox/22.0"
4 URL GOTO=https://www.bing.com/search?q=georgia&qs=n&form=QBRE&pq=georgia&sc=8-7&sp=-1&sk=&cvid=b94a6da1f4004670b5cfa65adaf95bfc
5 TAG POS=1 TYPE=INPUT:SEARCH FORM=ID:sb_form ATTR=ID:sb_form_q CONTENT=In<SP>history<SP>{{!NOW:ss}}
6 TAG POS=1 TYPE=INPUT:SUBMIT FORM=ID:sb_form ATTR=ID:sb_form_go
```

Algorithm 3. Script to Mobile Search Bing

Algorithm 1 shows a script used to iterate through a series of users. Each user runs a script to login to the site, perform 30 Bing searches, and then log out. Algorithm 2 is a call from Algorithm 1 online sixteen. The iteration on line fifteen of Algorithm 1 causes this to happen thirty times. Algorithm 2 sets the active tab of the browser and sets the user agent to look like a Windows Firefox browser. Lines five and six create a search that changes every second by including the static text "history" and the seconds' portion of the time. This random approach was used to ensure non-repetitive queries.

4. Mobile Browsing Rewards

Mobile browsing rewards are activities where a patron is incentivized to visit a business' web assets through a mobile application. The business is interested in creating habits in the patron's mobile internet browsing activities. I studied one loyalty programs that included mobile browsing rewards:

1. Microsoft™ Search Engine - Bing Rewards program was referenced in earlier rewards activity categories. Patrons can also earn credits by searching, through Bing, from a mobile device running iOS, Android or a Windows phone operating system. [9]

To evaluate the validity of the mobile browsing loyalty programs data, I included a step in my earlier script, Algorithm 1, to emulate a mobile browser. Lines eighteen and nineteen iterates across Algorithm 3. This algorithm sets the active tab of the browser and sets the user agent to look like an Android Firefox browser. Lines five and six create a search that changes every second by including the static text "history" and the seconds' portion of the time as I did in the earlier test.

5. Referral or Social Circle Rewards

Referral and social circle rewards are activities where a patron is incentivized to refer others to a business's reward program. The program can also be based on the number of members that are members of the reward program in their social circle. The business is interested in building their patron base by leveraging the patron's social clout. I studied two loyalty programs that included referral or social circle rewards:

1. Microsoft™ Search Engine - Bing Rewards program was referenced in earlier rewards activity categories. Patrons can also earn credits by referring others to join Bing Rewards. The referral can be done via a hyperlink or through Facebook friends. [5]
2. JetBlue Airways – JetBlue's TrueBlue program was referenced in earlier rewards activity categories. Patrons can also earn badges based on the number of Facebook friends that are also TrueBlue members. [3]

To evaluate the validity required of a social circle, I created fake Facebook user accounts and attempted to build the social clout required to earn the highest TrueBlue social circle badge. This badge requires you to have one hundred Facebook friends that are also TrueBlue members. I discovered a Facebook public group named JetBlue TrueBlue Badges [7]. The group is made up of over eight hundred members interested in the TrueBlue loyalty program. My fake FaceBook accounts submitted friend requests to all the members of this group and in less than a weeks' time had acquired the required social circle to earn the rewards.

6. User Opinion Rewards

User opinion rewards are currency systems where the user can earn points for filling out online surveys. Often these points can be exchanged for airline tickets, hotel reservations or car rental days. An example of an online opinion panel that rewards patrons with e-currencies is e-rewards [13]. Often opinion panels present a pre-survey to the user as a way to eliminate unqualified responses. In the case of e-rewards there is a small reward for taking the pre-survey and a larger reward for taking the full survey.

The vulnerability with these online opinion surveys is that it is very easy to script answer or multi-select answers to dozens of questions. The Google™ Chrome browser allows extensibility [5] of its functionality through the development of JavaScript extensions. We developed Google Chrome extensions that allowed recognized a grid of answers and pre-selected a complete column randomly. The current defense the opinion poll operators deploy is the throw fake questions that require a specific response. For example, a question would say "Select answer C for this question". We improved the extension to recognize these fake questions and answer appropriately.

7. Hardening

To tighten the application that manages the loyalty program, I define a series of constraints in Object Constraint Language (OCL). The constraint types include constraints on the grammar allowed, constraints on the timing of service calls, constraints on the validity of accounts, constraints on the client user agent. Declaration of the specification of the constraints is done in the Buddy System configuration files.

7.1. Hardening Against Unwanted Meaning

To express the grammar constraint, I allowed a feature grammar file defined in a text file [12]. The file contains the parts of speech tags that are acceptable for earning a reward. For the Marriott example, I created a simple grammar where a patron could micro-blog a positive experience at one of the Marriott properties. Figure 1 shows a sample microblog before and after tagging. Table 1 shows a subset of the production rules for the tweet feature grammar.

In the example from Figure 1 the parts of speech from the micro-blog are matched against the production rules in Table 1. In this case, the entire blog entry passes. In my implementation, points were earned, only if the grammar production rules successfully tag the complete microblog. To encourage customers to micro-blog the organization implementing a grammar would need to assist the user to help them blog successfully.

The Google™ Chrome browser allows extensibility of its functionality through the development of JavaScript extensions. We developed a Google Chrome extensions that would assist with the creation of successful micro-blogs. The extension would see the user was on the micro-blog site and detects the failure of the parse through web service


```
I enjoyed my stay last night at the Marriott Atlanta
I/PRP enjoyed/VBD my/PRP$ stay/NN last/JJ night/NN at/IN the/DT Atlanta Marriott/HOTEL
```

Figure 1. Sample Microblog with Tags

Table 1. Sample User Attribute Points

Phone Number	Unique	Non-Unique
Fixed Line	40	10
Mobile	20	5
PrePaid	5	2
VoIP	2	1
Pager	5	2
Payphone	0	0
Personal	-20	-40
Voicemail	-20	-40
Invalid	-40	-40
Address	**Unique**	**Non-Unique**
DPV	40	10
Invalid	-40	-40

calls to the parser. The unparsed tags in the micro-blog are highlighted with a change in the font color.

7.2. Hardening Against Scripted or Deleted Activities

There are two timing activities I constrain; deletion of rewarded activity and activities that happen to rapidly. In the case of social networking micro-blogs I ensure rewarded activity is not immediately removed from the blogs. In the case of browsing activity, I ensure that the actions taken are not scripted in a way that is replayed faster than a human can perform the steps. To express the timing constraints, I developed a method to store a copy of each tweet with a buried start date and end date. In the Buddy System configuration, the user can express the duration of the required relationship for the micro-blogs to be valid.

To ensure browser page visits are not scripted, I record the time between requests for a single user. If two requests in a row are received below the threshold time, then the activity is marked invalid. The solution also helps against opinion panel scripted attacks. A similar approach has been used to protect against dictionary attacks on login screens, by slowing validation with a fake delay time.

7.3. Hardening against Fake Users

To express user account validity constraints, I defined an algorithm that scores a user based on a set of attributes including:

- Mailing Address – The mailing address is scored for validly and uniqueness. The address is validated for correctness using the US Postal Service Delivery Service
- Phone – The phone number entered is scored based on the telephone type (landline, mobile, voice over IP, pre-paid) and uniqueness.
- Email- The email is validated using an anchor tag sent in an HTML validation email to the email address provided. If the user clicks the hyperlink, the score is increased.
- Twiter Account – The following attributes are used in the calculation of the validity of the account from twitter data; the presence of a Twitter account, the number of people following this account, the number of accounts this account follows and the number of times the account tweeted a micro-blog.

- FaceBook Account– The following attributes are used in the calculation of the validity of the account from Facebook account data; presence of an account, number of friends, number of status updates, number of posts to another friend's wall, number of times the account liked someone's status update
- Instagram Account t– The following attributes are used in the calculation of the validity of the account from Instagram account data; the presence of an account, the number of people following this account, the number of accounts this account follows and the number of times the account posted a micro-blog.

The score will range between zero and one hundred, and the user can specify a score, in the Buddy System configuration, that ensures enough believability to accept the user as a real user. Table 2 shows a sample attribute point stable used to score the user on the given phone numbers and address value. In the test, a score of 60 or higher was considered a valid user. In Table 2, point values are earned or deducted based on the validity of the phone number and the address. Values that were not duplicated across users in the system were treated as more credible.

An example user calculation is an online user who enters a valid address that passes the US postal service DPV web service lookup and a mobile phone number. The Postal Service™ DPV® database is essentially a "yes/no" table for checking the validity of any known individual house, apartment, post office box, rural box, mail drop, or commercial address that receives mail. The phone number would be determined to be mobile based on the reverse phone number lookup. These two pieces of information would summate to 60 if the user were the only user in the system who used the phone and address value. In this case, the unique, valid data is enough to assume the user is a valid user. In other cases, social networking data can be used to add value to the likelihood of the user's validity. Table 3 shows a sample extended attribute table to apply for social networking activity. For example, if the user has 50 friends on Facebook then they would score addition 2.5 points of validity. (50 * .05). The extended rules in Table 3 have maximum caps for each of the activities so one activity cannot contribute too much to the validity of the user.

7.4. Hardening Against Agent Spoofing

To express agent validity constraints, I need a way to ensure that the user agent passed in the HTTP header is valid for the machine passing the header. In the earlier sections, I described the web browsing activity rewarded in the Bing Rewards program. This program rewards searches separately that are performed from both a workstation and separately from a mobile device. In Algorithm 3 the user agent is changed to appear as if the browser is a mobile device. The HTTP protocol does not allow a server to know the validity of the user agent attribute. To compensate for the statelessness of the protocol I developed a solution that would remember the user agent value for a particular machine. Identifying the machine's user agent is easily completed by pairing the source IP address of the requester with the user agent on an incoming request. If the constraint observes a request where the user agent has changed, then the request is marked fraudulent.

Unfortunately, this solution does not work if the client machines are hidden behind a NAT (Network Address Translator) device. The NAT device will share a single or a few IP addresses with the hosts behind the device. If a hidden host makes a request for an asset outside the local network, then the NAT device will replace the source address in the IP packet header. The shared IP addresses make it difficult to know if the user agent attribute has changed or if the requests are coming from a second device behind the NAT. I utilize a technique developed by Bellovin [9] to categorize traffic behind a NAT-based router. The technique is based on the IPid attribute in the TCP header. This method was originally developed to count the number of devices behind the NAT. In the technique, IPids are used to identify traffic coming from unique devices. The IPid is inside the TCP/IP packet header. It is included to allow the packet fragmentation on the source device and reassembly on the destination device. NAT devices do not change the value of the IPid, which allows us to identify unique devices behind the NAT and store the matching user agent attribute. If the Buddy System recognized the attribute change in a category of traffic, the request is marked fraudulent.

Table 2. Sample Production Rules

Tag	*Rule*
M - Microblog	PRP VBD PRP$ NN JJ NN IN DT HOTEL
PRP - Personal pronoun	I
VBD - Verb, past tense	Stayed
PRP$ - Possessive pronoun	My
NN - Noun, singular or mass	stay, night
JJ - Adjective	last
IN - Preposition or subordinating conjunction	at
DT - Determiner	the
HOTEL - Hotel	Atlanta Marriott, Charleston Marriott

Table 3. Sample User Extended Attribute Points

	Fixed	Per	Max
Facebook	5		
friends		0.05	10
photos-uploaded		0.05	10
photos-tagged		0.05	10
posts		0.05	10
posts-commented		0.05	10
posts-liked		0.05	10
Twitter	5		
followers		0.1	10
tweets		0.05	10
following		0.05	10
Instagram	5		
follows		0.1	10
Followed by		0.2	10
comments		0.01	10
liked-media		0.01	10

8. Empirical Results

A web application was built for a model e-Commerce store that provides loyalty points with promotions similar to those mentioned earlier. The system was tested with concurrent users accessing the server in groups of current users ranging in blocks of one hundred from one hundred to one thousand. The users hit two different systems; One system is utilizing the Buddy System with eight clusters, and one system utilizing lazy replication. Figure 2 shows that the Buddy System providing higher consisting by applying the application domain security constraints outperformed the lazy replication system with the lower security and consistency.

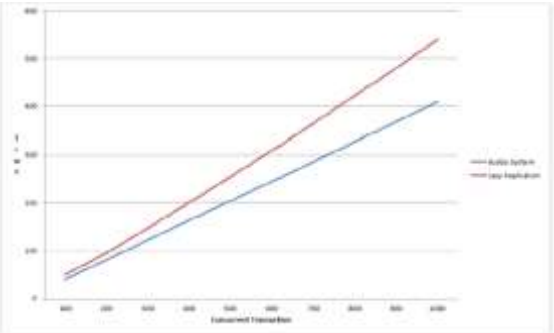

Figure 2. Empirical Results

9. Related Work

Constraint specification and enforcement have been a part of the relational database model research since Codd [6] originally wrote the specification. Recently work on auto-generation of SQL code to enforce these constraints from the UML model has been done by Heidenreich, et al. [7] and Demuth, et al. [8]. In both these works, the focus is on the generation of the SQL code for relational databases to enforce the invariants. In the cases of the loyalty rewards programs addressed in this paper, the constraints need to span the partitions of the web application architecture and include the state of the individual web sessions.

There are many regression testing tools available to simulate the behavior of an actual human client. Bayo Erinle [18] describes a method to use the open source tool JMeter to script regression test to ensure a web application can handle a specific load capacity. This same methodology could be used by malicious users to script the behavior that is being rewarded by the loyalty program.

Edwards, et al. [19] study the use of an online scoring metric to make judgments and perceptions of a person's credibility can be made from examining posts, tweets, or other indicators on social media. In the study, they use the Klout.com score to evaluate the credibility of a social media micro-blog poster. Klout.com developed a popular indicator of this influence that creates a single score based on the idea that "Everyone has influence-the ability to drive action".

Buddy System: In my previous work [11], I provide architecture and algorithms that address three problems: the risk of losing committed transactional data in case of a site failure, contention caused by a high volume of concurrent transactions consuming limited items, and contention caused by a high volume of read requests. I called this system the Buddy System because it used pairs of clusters to update all transactions synchronously. The pairs of buddies can change for each request allowing increased availability by fully utilizing all server resources available. Consistency is increased over lazy-replication because all transactional elements are updated in the same cluster allowing for transaction time referential integrity and atomicity.
An intelligent dispatcher is in front of all clusters. The dispatcher operates at the OSI Network level 7. The placement allows the dispatcher to use specific application data for transaction distribution and buddy selection. The dispatcher receives the requests from clients and distributes them to the WS clusters. Each WS cluster contains a load balancer, a single database, and replicated services. The load balancer receives the service requests from the dispatcher and distributes them among the service replicas. Within a WS cluster, each service shares the same database. Database updates among the clusters are propagated using lazy replication propagation.

The dispatcher picks the two clusters to form the buddy pair. The selection uses versioning history to

determine which clusters are qualified. If a version is in progress, and the request is modifying the data, then the dispatcher chooses set containing the same pair currently executing the other modify transactions. Otherwise, the set contains any pair with the last completed version. The primary buddy receives the transaction along with its buddy's IP address. The primary buddy becomes the coordinator in a simplified commit protocol between the two buddies. Both buddies perform the transaction and commit or abort together.

The dispatcher maintains metadata about the freshness of data items in the different clusters. The dispatcher increments a version counter for each data item after it has been modified. Any two service providers (clusters) with the latest version of the requested data items can be selected as a buddy. Note, that the database maintained by the two clusters must agree on the requested data item versions but may be different for the other data items.

In a follow-up work [12] I extend the Buddy System with constraint optimization. In this work, I allow modeling of constraints using OCL (Object Constraint Language). The dispatcher reads in the OCL and will enforce the constraints when an incoming web-service request is received.

10. Conclusion

In this paper, I investigate the problem of providing application domain security constraints to distributed systems will maintaining high availability. I propose an extension to the buddy system to provide an application domain constraint filter. To develop the application domain constraint filters, I consider five loyalty activity categories rewarded by businesses to their patrons; social networking rewards, web-site browsing rewards, mobile browsing rewards, referral/social circle rewards and video viewing rewards. I document vulnerabilities with each activity category, propose and implement a solution that ensures the activity being rewarded is the activity that is intended by the reward program. In this work, I handled specific e-Currency correctness constraints. In future work, I will develop a modeling mechanism through the combination of OCL and UML stereotypes to specify correctness of large collection e-Currency constraints.

11. References

[1] "Fastest Way to Lose Customers," KISSmetrics, [Online]. Available: https://blog.kissmetrics.com/retaining-customers/. [Accessed 31 03 2015].

[2] A. Ha, "Klout Users Can Now Add Bing To Their Account And Include Instagram In Their Score," TechCrunch, http://techcrunch.com/2013/03/28/klout-instagram-bing/. [Accessed 21 04 2015].

[3] A. Olmsted and C. Farkas, "High Volume Web Service Resource Consumption," in Internet Technology and Secured Transactions, 2012. ICITST 2012, London, UK, 2012.

[4] A. Olmsted and C. Farkas, "The cost of increased transactional correctness and durability in distributed databases," in 13th International Conference on Information Reuse and, Los Vegas, NV, 2012.

[5] M. Aron, D. Sanders, P. Druschel and W. Zwaenepoel, "Scalable content-aware request distribution in cluster-based networks servers," in Proceedings of the annual conference on USENIX Annual Technical Conference, ser. ATEC '00, Berkeley, CA, USA, 2000.

[6] A. Olmsted and R. Stalvey, "Service Constraint Guarantees," International Journal of Intelligent Computing Research, vol. 5, 2014.

[7] "Marriott Rewards Plus Points," [Online]. Available: www.marriottrewardspluspoints.com. [Accessed 01 08 2014].

[8] "Introducing TrueBlue Badges," JetBlue Airways, [Online]. Available: https://badges.jetblue.com/how-it-works. [Accessed 01 12 2014].

[9] "Social Rewards," [Online]. Available: http://socialrewards.com/. [Accessed 05 12 2014].

[10] "Microsoft Bing Rewards," [Online]. Available: https://www.bing.com/rewards/faq#Howdoestherewardsprogramwork. [Accessed 25 03 2015].

[11] "iMacros Overview," 27 03 2015. [Online]. Available: http://imacros.net/overview. Figure 2 – Empirical Results

[12] "JetBlue TrueBlue Badges," FaceBook, Inc, [Online]. Available: https://www.facebook.com/groups/281151118694303/. [Accessed 03 04 2015].

[13] S. Bird, "NLTK: the natural language toolkit," in COLING-ACL '06 Proceedings of the COLING/ACL on Interactive presentation sessions , Stroudsburg, PA, 2006.

[14] S. M. Bellovin, "A Technique for Counting NATted Hosts," in IMW'02, Marseille, France, 2002.

[15] E. F. Codd, The Relational Model for Database Management, Boston, MA: Addison-Wesley Longman Publishing Co., Inc., 1990.

[16] F. Heidenreich, C. Wende and B. Demuth, "A Framework for Generating Query Language Code," Electronic Communications of the EASST, 2007.

[17] B. Demuth, H. Hußmann and S. Loecher, "OCL as a Specification Language for Business Rules in Database Applications," in The Unified Modeling Language. Modeling Languages, Concepts, and Tools., Springer, 2001, pp. 104-117.

[18] B. Erinle, in Performance Testing With JMeter 2.9, Packt Publishing, 2013.

[19] C. Edwards, P. Spence, C. Gentile, A. Edwards and A. Edwards, "How much Klout do you have...A test of system generated cues on source credibility," Computers in Human Behavior, 2013.

[20] A. Olmsted and C. Farkas, "Buddy System: Available, Consistent, Durable Web Service Transactions," Journal of Internet Technology and Secured Transactions, vol. 3, 2013.

[21] Google, "Gooogle Chrome Extension Overview," [Online]. https://developer.chrome.com/extensions/ overview. [Accessed 8 4 2014].

