ReSIST: Resilience for Survivability in IST

A European Network of Excellence

Contract Number: 026764

Deliverable D14: Student Seminar Programme

Report Preparation Date: June 2006
Classification: Public Circulation

Contract Start Date: 1st January 2006
Contract Duration: 36 months
Project Co-ordinator: LAAS-CNRS
Partners: Budapest University of Technology and Economics
City University, London
Technische Universität Darmstadt
Deep Blue Srl
Institut Eurécom
France Telecom Recherche et Développement
IBM Research GmbH
Université de Rennes 1 – IRISA
Université de Toulouse III – IRIT
Vytautas Magnus University, Kaunas
Fundação da Faculdade de Ciencas da Universidade de Lisboa
University of Newcastle upon Tyne
Università di Pisa
QinetiQ Limited
Università degli studi di Roma "La Sapienza"
Universität Ulm
University of Southampton

Student Seminar

Programme
and
Collection of Extended Abstracts

Edited by Luca Simoncini (Università di Pisa)

Index

ReSIST Student Seminar

1. Aims of the ReSIST Student Seminar

The student seminar aims to foster the integration of the ReSIST Doctorate students within the network, via a) the presentation of their research work to other ReSIST students and researchers, and b) the discussions that will ensue.

This Seminar is only open to ReSIST members. Attendance at the Seminar is limited in order to encourage highly interactive discussions.

Participation of members of both genders is particularly encouraged.

2. Venue

The ReSIST Student Seminar will take place on September 5-7, 2006 at Centro Studi "I Cappuccini", Via Calenzano 38, San Miniato, Pisa, Italy

3. Call for Papers/Participation

The Call for Papers/Participation (Appendix A) was distributed to all ReSIST Member sites on March 1, 2006 and by the submission deadline the situation was, in terms of number of submitted papers, of student's authors and distribution among the ReSIST sites:

Total number of submitted papers: 30
Total number of student authors: 32 (two papers are with two authors)

Darmstadt: 1+1(joint with BUTE)
IRIT: 1
Newcastle: 1+1+1+1+1
BUTE: 1(joint with Darmstadt)+1
LAAS: 1+1+1+1
Pisa: 1+1
IRISA: 1+1
IBM: 1
Roma: 1+1
Southampton: 1 (two authors)
Lisbon: 1+1+1
Eurecom: 1+1
France Telecom: 1
City: 1+1+1

4. Decisions of the Program Committee

The Program Committee for the ReSIST Student Seminar was composed by the members of the Training and Dissemination Committee. They interacted via e-mail and the agreed decisions were:

on the format of the Seminar:

- Sessions of student papers should present clear problem statements, the core technology aspects and the lines they plan to follow during their PhD, moderated by more mature students. These sessions should have a long and free discussion part at their end;
- Panels-like sessions, with short statements presented in an informal way, mainly run by mature students on either results or open problems, with lively contributions of the audience;
- Free gathering and informal discussions in small number of students during the seminar, as requested by the students themselves;
- The role of participants to the Seminar:
    - **Speakers**: students presenting their paper should: a) distribute presentation time into 5/7 minutes dedicated to the general topic and 10/8 minutes on the associated research direction, avoiding all possible technicalities of his/her research. A maximum of slides (7 at most - 3 for the general topic and 4 for his/her associated research) is requested;
    - The general discussion at the end of each session should provide the present general framework and target of ongoing research in the specific field at the light of the presented papers. The **senior student** that will act as session chair and discussion moderator should: a) read all contributions in his/her session, b) prepare questions to foster discussion and c) try to summarize, with the help of the audience, what is the present situation on the topic at the light of the presented papers;
    - Panels: they must be visionary and should provide: a) identification of gaps that need to be covered for allowing resilience scaling technologies (evolvability, assessability, usability and diversity) and b) steps to cover the gaps. The **senior student** who moderates the panel should: a) have a clear view of the ReSIST goals, b) prepare a short introduction to the panel for channelling the discussion and c) provide instructions to the panellists on how prepare their statements. The **senior** who helps the student moderator in organizing the panel should: a) interact with the student moderator in preparing the panel and b) prepare a report of the panel for presentation in the last day general discussion.
    - **Seniors** attending the seminar should contribute to the discussions.

On the basis of these indications, the decisions of the Program Committee were:

- All 30 submissions were accepted, as well as structuring the Student Seminar into Sessions, discussion at the end of each sessions, panels on the topics of the session;
- Sessions will be chaired by a senior student (possibly one of the presenter) who will also moderate the final discussion at the end of the session;
- Panels will be moderated by a senior student with the help, in the organization part, of a senior member;
- Minutes of the panels will be taken and reported in the last day in the general discussion.

The accepted papers were assigned to several Sessions as in Table 1, were the first column indicated the name of the authors and the name of the thesis advisor, when either indicated or identified.

The entire organization, structure and Advance Program was presented to and approved by the ReSIST Executive Board that was held in Paris on June 6, 2006. The Sessions, the names of the proposed Session Chairs and Moderators for the Open discussions as well as the proposed Panels Moderators and Panel Co-organizers were presented and approved. It was finally identified the list of senior members that are willing to participate to the Student Seminar. They are: Neeraj Suri, Chidung Lac, Al Avizienis, Michel Raynal, Hugh Glaser, Holger Pfeifer, Birgit Pfizmann, Luca Simoncini, Jean-Claude Laprie, Karama Kanoun, Istvan Majzik, Marc-Olivier Killijian, Paulo Verissimo, Brian Randell, Roberto Bonato, Yves Roudier.
The final number of participants will therefore be 32 Students + 16 Senior Members, fitting the budget allocated for this event.

In Appendix B, the ReSIST Student Seminar Advance Program as well as the set of extended abstracts of the papers that will be presented, divided into sessions are reported.

The findings and suggestions arising from the Panels discussions as well as the final list of participants will be reported in the first Periodic Activity Report after the end of the Student Seminar.

	Security	Systems Modelling	Mobile Systems	Model Based Verification	Distributed Systems	Diversity
Alata/Kaaniche	X					
Andreou/van Moorsel	X					
Antunes/Neves	X					
Basnyat/Palanque		X				
Courtes/Powell			X			
Falai/Bondavalli					X	
Gashi/Popov						X
Gramoli/Raynal					X	
Iliasov/Romanovsky			X			
Jaffri-Rodriguez/Glaser		X				
Lamprecht/van Moorsel	X					
Lauritsen/Anderson		X				
Leita/Dacier	X					
Martin-Guillerez/Banatre			X			
Masci/Bernardeschi				X		
Mendizabal/Casimiro					X	
Mian/Baldoni			X			
Micskei/Majzik				X		
Minh Duc/Waeselynck				X		
Mueller/Pfitzmann		X				
Pham/Dacier	X					
Ramanathan/Chidung Lac		X				
Salako/Strigini		X				
Salatge/Fabre					X	
Sarbu/Suri				X		
Scipioni/Baldoni					X	
Serafini-Bokor/Suri				X		
Sousa/Verissimo					X	
Stankovic/Popov						X
Yuhui/Romanovsky					X	

Table 1

Appendix A

Call for Papers/Participation to ReSIST Student Seminar

ReSIST Student Seminar
5-7 September 2006,
Centro Studi "I Cappuccini"
Via Calenzano 38, San Miniato, Pisa, Italy

CALL FOR PAPERS/PARTICIPATION

Aims of the ReSIST Student Seminar

The student seminar aims to foster the integration of the ReSIST Doctorate students within the network, via a) the presentation of their research work to other ReSIST students and researchers, and b) the discussions that will ensue.

This Seminar is only open to ReSIST members. Submissions are welcomed from Doctorate Students of all ReSIST partners. Attendance at the Seminar will be limited in order to encourage highly interactive discussions.

Participation of members of both genders is particularly encouraged.

Format of the Seminar

Accepted papers will be selected for presentation in several themed sessions in relation to the ReSIST Description of Work (DoW).

A total number of 20-25 presentations is envisioned, in order to leave significant space for discussions.

We encourage a dynamic presentation style (as opposed to simply reading one's slides!!).

Time and Place

The ReSIST Student Seminar will take place on September 5-7, 2006 at Centro Studi "I Cappuccini", Via Calenzano 38, San Miniato, Pisa, Italy

* How to reach San Miniato from Pisa*

On Arrival:

By car:
If you arrive at Pisa Airport and rent a car, you have to follow, at airport exit, the direction towards Florence (Firenze) on S.G.C. (means Strada di Grande Comunicazione-High Traffic Highway). Exit at San Miniato, and follow direction to San Miniato Center. When you are up-hill in San Miniato, follow the direction to "I Cappuccini" (Parking lots available at Centro Studi) (average time 30-40 minutes).

By train:
At Pisa Airport proceed towards the train track, outside the main Hall. In the main Hall please buy a train ticket to Empoli, and do not forget to obliterate it at the machine next to ticket office or next to the train track. All trains are OK, with an average frequency of one train per hour. At Empoli station take a taxi to "I Cappuccini" in San Miniato (average cost 15 Euros).

On Departure:

Two buses will be organized to bring back to Pisa Airport the participants leaving on September 7 (after lunch) and for participants leaving on September 8 (after the end of the EB).

Registration fees

Participation to SS is charged to ReSIST, and the participants will have to pay a registration fee covering full board accomodation in the Centro Studi "I Cappuccini".
Fee per participant:

Students (from Sept. 4 to 7)	600,00
Seniors (from Sept. 4 to 8)	750,00
Seniors (arriving Sept. 7 to 8)	230,00

The fee includes:
1) for students (hotel room, breakfasts, coffee-breaks, lunches including Sept.7, dinners including Sept. 4 and banquet on Sept.6, transportation to Pisa airport on Sept. 7 afternoon)
2) for seniors (staying from Sept.4 to Sept.8 - that is attending SS, Committees and EB) (hotel room, breakfasts, coffee-breaks, lunches including Sept.8, dinners including Sept. 4, banquets on Sept. 6 and on Sept. 7, transportation to Pisa airport on Sept. 8 afternoon)
3) for seniors (staying from Sept.7 to Sept. 8 - that is additional seniors coming only for the Committees and EB) (hotel room, breakfast, coffee-breaks, lunches including Sept.7 and 8, banquet on Sept. 7, transportation to Pisa airport on Sept. 8 afternoon).

Submissions

Topics for Submission

All members of the Executive Board of ReSIST should proactively solicit their Doctorate Students to propose a submission, and will provide a preliminary screening on the topics of such proposed submissions, so to finalize actual submissions both in terms of numbers and in terms of relevance to the ReSIST DoW.

Instructions for Submission

1. Prepare an extended abstract (2 to 4 A4 pages) .pdf format on a topic relevant to ReSIST.

2. Complete the submission form (below), including suggestion(s) for discussions, and return it with the extended abstract to:
<**resist-td@laas.fr**>

+ The deadline for submissions is **14 April 2006**
+ Applicants will be notified by **19 May 2006**
+ The workshop is only open to ReSIST members
+ Multiple submissions from member partners will be accepted (subject to

the constraint of achieving as broad a representation of members as possible)
+ The collection of accepted abstracts will be made available on the ReSIST web site before the seminar, as well as the presentation slides after the seminar.

Programme/Organising Committee
The Program Committee is composed by the Members of the Training and Dissemination Committee

---

--------------Complete and return this form--------------
with extended abstract

To: <resist-td@laas.fr>
By: 14 April 2006

Name:

ReSIST Partner:

Email Address:
Telephone:
Fax:

Brief Biography:
(1-2 paragraphs including current position and research activities, with keywords)

Suggested topic(s) for discussions:

--------------------- End of Form ---------------------

(Attach extended abstract, equivalent of 2 to 4 A4 pages)

end
================

Appendix B

ReSIST Student Seminar Advance Program
and
Extended Abstracts divided into Sessions

Student Seminar Advance Program

September 5	**08.30 – 09.00**	**Welcome and Seminar presentation**
	09.00 – 11.00	**Session on Security** Chair: Christiaan Lamprecht, University of Newcastle upon Tyne, UK
	09.00 – 09.20 An Analysis of Attack Processes Observed on a High-Interaction Honeypot – Eric Alata, LAAS-CNRS, France **09.20 – 09.40 Identifying the Source of Messages in Computer Networks - Marios S. Andreou, University of Newcastle upon Tyne, UK** **09.40 – 10.00 Vulnerability Assessment Through Attack Injection - João Antunes, University of Lisbon, Portugal** **10.00 – 10.20 Adaptive Security – Christiaan Lamprecht, University of Newcastle upon Tyne, UK** **10.20 – 10.40 ScriptGen: Using Protocol-Independence to Build Middle-Interaction Honeypots - Corrado Leita, Institut Eurecom, France** **10.40 – 11.00 Early Warning System Based on a Distributed Honeypot Network - VanHau Pham, F. Pouget, M. Dacier, Institut Eurecom, France**	
	11.00 – 11.20	**Open discussion on the Session** Moderator: Christiaan Johan Lamprecht, University of Newcastle upon Tyne, UK
	11.20 – 11.50	**Coffee break**
	11.50 – 12.50	**Panel on Security Issues** Organizers: Eric Alata, LAAS-CNRS, France, Birgit Pfizmann, IBM Research, Switzerland Panelists: TBD
	12.50 – 14.20	**Lunch**

14.20 – 16.20	**Session on System Modelling** Chair: Benedicto Rodriguez, University of Southampton, UK
14.20 – 14.40 A Multi-Perspective Approach for the Design of Error-tolerant Socio-Technical Safety-Critical Interactive Systems - Sandra Basnyat, University Paul Sabatier, France **14.40 – 15.00 ReSIST Knowledge Architecture: Semantically Enabling Large-scale Collaborative Projects - Afraz Jaffri, Benedicto Rodriguez, University of Southampton, UK** **15.00 – 15.20 Introducing Hazard and Operability Analysis (HazOp) in Business Critical Software Development - Torgrim Lauritsen, University of Newcastle upon Tyne, UK** **15.20 – 15.40 Enterprise Compliance at Stake. Dependability Research to the Rescue! - Samuel Müller, IBM Zurich Research Lab, Switzerland** **15.40 – 16.00 Fault Modelling for Residential Gateways – Sakkaravarthi Ramanathan, France Telecom, France** **16.00 – 16.20 Modelling Dependence and its Effects on Coincident Failure in Fault-Tolerant, Software-based Systems – Kizito Salako, CSR, City University, UK**	
16.20 – 16.40	**Open discussion on the Session** Moderator: Benedicto Rodriguez, University of Southampton, UK
16.40 – 17.10	**Coffee break**
17.10 – 18.10	**Panel on System Modelling** Organizers: Kizito Salako, City University, UK, Brian Randell, University of Newcastle upon Tyne, UK Panelists: TBD
18.10 – 20.00	**Free gathering and informal discussions**
20.30	**Dinner**

September 6	**08.30 – 10.10**	**Session on Model-based Verification** Chair: Paolo Masci, University of Pisa, Italy
	08.30 – 08.50 Detecting Data Leakage in Malicious Java Applets - Paolo Masci University of Pisa, Italy **08.50 – 09.10 Handling Large Models in Model-based Testing - Zoltán Micskei, Budapest University of Technology and Economics, Hungary** **09.10 – 09.30 Testing Applications and Services in Mobile Systems – Nguyen Minh Duc, LAAS-CNRS, France** **09.30 – 09.50 Behavior-Driven Testing of Windows Device Drivers – Constantin S rbu, Technical University of Darmstadt, Germany** **09.50 – 10.10 On Exploiting Symmetry to Verify Distributed Protocols – Marco Serafini, Technical University of Darmstadt, Germany and Péter Bokor, Budapest University of Technology and Economics, Hungary**	
	10.10 – 10.30	**Open discussion on the Session** Moderator: Paolo Masci, University of Pisa, Italy
	10.30 – 11.00	**Coffee break**
	11.00 – 12.00	**Panel on Model-based Verification** Organizers: Constantin S rbu, Technical University of Darmstadt, Germany, Istvan Majzik, Budapest University of Technology and Economics, Hungary Panelists: TBD
	12.00 – 12.40	**Session on Diversity** Chair: - Ilir Gashi, CSR, City University, UK
	12.00 – 12.20 Potential for Dependability Gains with Diverse Off-The-Shelf Components: a Study with SQL Database Servers - Ilir Gashi, CSR, City University, UK **12.20 – 12.40 Improvement of DBMS Performance through Diverse Redundancy - Vladimir Stankovic, Peter Popov, CSR, City University, UK**	
	12.40 – 13.00	**Open discussion on the Session** Moderator: - Ilir Gashi, CSR, City University, UK
	13.00 – 14.30	**Lunch**

14.30 – 15.50	**Session on Mobile Systems** Chair: Ludovic Courtès, LAAS-CNRS, France
14.30 – 14.50 Storage Mechanisms for Collaborative Backup of Mobile Devices - Ludovic Courtès, LAAS-CNRS, France **14.50 – 15.10 Towards Error Recovery in Multi-Agent Systems - Alexei Iliasov, University of Newcastle upon Tyne, UK** **15.10 – 15.30 Increasing Data Resilience of Mobile Devices with a Collaborative Backup Service - Damien Martin-Guillerez, IRISA/ENS-Cachan, France** **15.30 – 15.50 Random Walk for Service Discovery in Mobile Ad hoc Networks - Adnan Noor Mian, University of Rome “La Sapienza”, Italy**	
15.50 – 16.10	**Open discussion on the Session** Moderator: Ludovic Courtès, LAAS-CNRS, France
16.10 – 16.40	**Coffee break**
16.40 – 17.40	**Panel on Mobile Systems** Organizers: Adnan Noor Mian, University of Rome “La Sapienza”, Italy, Marc-Olivier Killijian, LAAS-CNRS, France Panelists: TBD
17.40 – 19.30	**Free gathering and informal discussions**
20.30	**Banquet at Belvedere**

September 7	**08.30 – 10.50**	**Session on Distributed Systems** Chair: Paulo Sousa, University of Lisbon, Portugal
	08.30 – 08.50 Quantitative Evaluation of Distributed Algorithms - Lorenzo Falai, University of Florence, Italy **08.50 – 09.10 From Fast to Lightweight Atomic Memory in Large-Scale Dynamic Distributed Systems - Vincent Gramoli, IRISA - INRIA Rennes, CNRS, France** **09.10 – 09.30 Dependable Middleware for Unpredictable Environments – Odorico M. Mendizabal, António Casimiro, Paulo Veríssimo** **09.30 – 09.50 A Language Support for Fault Tolerance in Service Oriented Architectures - Nicolas Salatge, LAAS-CNRS, France** **09.50 – 10.10 Challenges for an Interoperable Data Distribution Middleware - Sirio Scipioni, University of Rome "La Sapienza", Italia** **10.10 – 10.30 Proactive Resilience - Paulo Sousa, University of Lisbon, Portugal** **10.30 – 10.50 A Mediator System for Improving Dependability of Web Services - Yuhui Chen, University of Newcastle upon Tyne, UK**	
	10.50 – 11.10	**Open discussion on the Session** Moderator: Paulo Sousa, University of Lisbon, Portugal
	11.10 – 11.30	**Coffee break**
	11.30 – 12.30	**Panel on Distributed Systems** Organizers: Vincent Gramoli, IRISA - INRIA Rennes, CNRS, France, Paulo Verissimo, University of Lisbon, Portugal Panelists: TBD
	12.30 – 13.30	**General discussion and wrap-up**
	13.30 – 15.00	**Lunch**
	15.00	**End of Student Seminar and bus back to Pisa Airport**

Session on Security

Chair: Christiaan Johan Lamprecht,
University of Newcastle upon Tyne, UK

An analysis of attack processes observed on a high-interaction honeypot

E. Alata

LAAS-CNRS

7, Avenue du Colonel Roche

31077 Toulouse Cedex - France

Tel : +33/5 61 33 64 53 - Fax: +33/5 61 33 64 11

Introduction

During the last decade, the Internet users have been facing a large variety of malicious threats and activities including viruses, worms, denial of service attacks, phishing attempts, etc. Several initiatives, such as Sensor project [3], CAIDA [11] and DShield [6], have been developed to monitor real world data related to malware and attacks propagation on the Internet. Nevertheless, such information is not sufficient to model attack processes and analyse their impact on the security of the targeted machines. The CADHo project [2] in which we are involved is complementary to these initiatives and it is aimed at filling such a gap by carrying out the following activities:

- deploying and sharing with the scientific community a distributed platform of honeypots [10] that gathers data suitable to analyse the attack processes targeting a large number of machines connected to the Internet.
- validating the usefulness of this platform by carrying out various analyses, based on the collected data, to characterize the observed attacks and model their impact on security.

A honeypot is a machine connected to a network but that no one is supposed to use. If a connection occurs, it must be, at best an accidental error or, more likely, an attempt to attack the machine. Two types of honeypots can be distinguished depending on the level of interactivity that they offer to the attackers. Low-interaction honeypots do not implement real functional services. They emulate simple services and cannot be compromised. Therefore, these machines can not be used as stepping stones to carry out further attacks against third parties. On the other hand, high-interaction honeypots offer real services to the attackers to interact with which makes them more risky than low interaction honeypots. As a matter a fact, they offer a more suitable environment to collect information on attackers activities once they manage to get the control of a target machine and try to progress in the intrusion process to get additional privileges.

Both types of honeypots are investigated in the CADHo project to collect information about the malicious activities on the Internet and to build models that can be used to characterize attackers behaviors and to support the definition and the validation of the fault assumptions considered in the design of secure and intrusion tolerant systems.

The first stage of the project has been focused on the deployment of a data collection environment (called Leurré.com [1]) based on low-interaction honeypots. Several analyses carried out based on the data collected so far from such honeypots have revealed that very interesting observations and conclusions can be derived with respect to the attack activities observed on the Internet [2][8][7][9][10]. The second stage of the CADHo project is aimed at setting up and deploying high-interaction honeypots to allow us to analyse and model the behavior of malicious attackers once they have managed to compromise and get access to a new host, under strict control and monitoring. We are mainly interesting in observing the progress of real attack processes and the activities carried out by the attackers in a controlled environment.

We describe in this paper our implementation choices for the high-interaction honeypot as well as its configuration to capture the activity of the intruder. Then we summarize some of the results of our analyses, mainly 1) global statistics to give an overview of the activity on the honeypot 2) the description of the intrusion process that we observed, 3) the behavior of the intruders (once they have broken into the honeypot) in order to analyse whether they are human or robots, whether they are script kiddies or black hats, etc.

1. Architecture of our honeypot

In our implementation, we chose to use VMware software [13] and a virtual operating system upon VMware. When an intruder succeeds in breaking into our honeypot, all the commands that he uses are captured and are transferred to a database for post-processing. Our honeypot is a standard Gnu/Linux installation, with kernel 2.6 and the usual binary tools (compiler, usual commands, etc). So that the intruder can break into our honeypot, we chose to use a simple vulnerability: weak passwords for user accounts. Of course, remote *ssh* connections to the virtual host with *ssh* are authorized (and only these ones) so that the attacker can exploit this vulnerability. In order to log all the login and passwords used by the intruders, we modified the source code of the *ssh* server running on the honeypot. In the same way, we made some modifications of the virtual operating system kernel to log all the commands used by the intruders.

The activities of the intruder logged by the honeypot are preprocessed and then stored in an SQL database. The raw data are automatically processed to extract relevant information for further analyses, mainly 1) the IP address of the attacking machine, 2) the login and the password tested, 3) the date of the connection, 4) the terminal associated (*tty*) to each connection, 5) each command used by the attacker.

2. Architecture of our honeypot

2.1. Overview of the activity

The high-interaction honeypot deployed has been running for 131 days. Overall, 480 IP addresses were seen on the honeypot. As illustrated on Figure 2.1.1, 197 IP addresses performed dictionary attacks and 35 carried out real intrusion on the honeypot (see next section). The 248 IP addresses left were used for other activity such as scanning, etc. Among the 197 IP addresses that made dictionary attacks, 18 succeeded in finding passwords (see Figure 2.1.1).

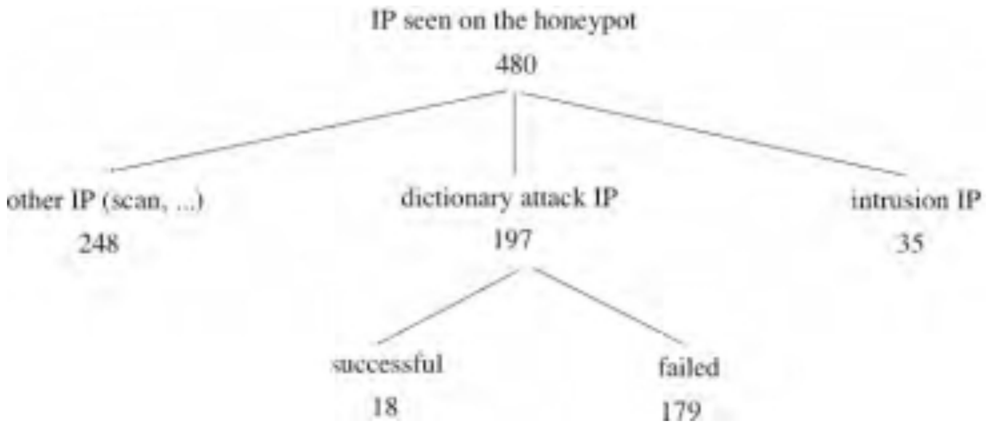


Figure 2.1.1: Classification of IP addresses seen on the honeypot

The number of *ssh* connection attempts to the honeypot is 248717 (we do not consider here the scans on the *ssh* port). This represents about 1900 connection attempts a day. Among these 248717 connection attempts, only 344 were successful. The total number of accounts tested is 41530. The most attacked account is of course the *root* account.

After an observation period, we have decided to create 16 user accounts. Some accounts belong to the most attacked accounts and some do not. We analyse in detail how these accounts were broken and what the intruders did once they have broken into the system.

2.2. Intrusion process

The experiment revealed that the intrusion process is always exactly the same, in two steps.

The first step of the attack consists in dictionary attacks. In general, these attacks succeed in discovering our weak passwords in a few days. By analysing the frequency of the *ssh* connections attempts from the same attacking IP address, we can say that these dictionary attacks are performed by automatic scripts. Furthermore, some finer analyses highlight that the addresses that executed dictionary attacks did not try other attacks on our honeypot.

The second step of the attack consists in the real intrusion. We have noted that, several days after the guessing of a weak password, an interactive *ssh* connection is realized on our honeypot by a human being. These intrusions come from 35 IP addresses, and, surprisingly, these machines, for half of them, come from the same country.

Further analyses revealed that these machines did not make any other kind of attack on our honeypot, i.e, no dictionary attack. It is noteworthy that none of the 35 IP addresses has been observed on the low-interaction honeypots deployed through the Internet in the CADHo project (approximately 40). This is interesting because it shows that these machines are totally dedicated to this kind of attack. They only target at our high-interaction honeypot and only when they know at least one login and password on this machine.

We can conclude for these analyses that we have to face two distinct groups of machines that communicate with each other. The first group is composed of machines that are specifically in charge of making automatic dictionary attacks on some machines. The information collected is then published somewhere and used by the second group of machines to perform real intrusions. This second group of machines is driven by human being, and is analysed in more details in the next section.

2.3. Behavior of attackers

We tried to identify who are the intruders. Either they are humans, or they are robots which reproduce simple behaviors. During our observation period, for 24 intrusions upon 38, intruders have made mistakes when typing commands. So, it is very likely that such activities are carried out by humans, rather than robots. When an intruder has not make any mistake, we analysed how the data were transmitted from the attacker machine to the honeypot on the TCP connection. Thanks to this method, we also concluded that intruders are human being.

In a general way, we have noted three main activities of the intruders. The first one is launching *ssh* scans on other networks from the honeypot. The honeypot is thus used as a rebound to start new attacks. The second type of activity is launching *irc* botnet. Some examples are *emech* [12] and *psyBNC*. The binaries of these software were regularly changed in *crond* or *inetd* which are well known binaries on Unix systems, in order to dissimulate them. A small part of the intruders also tried to become root. They used rootkits that unfortunately failed on our honeypot.

Intruders can be classified in two main categories. The most important one is relative to *script kiddies*. They are inexperienced *hackers* who use programs found on the Internet without really understanding how they work. The next category represents intruders who are more dangerous, named ``black hat", that can make serious damage on systems because they are security experts. The intruders we have observed are most of the time of the first category. For example, many of them don't seem to really understand the Unix file access rights and some of them try to kill the processes of other users. Some intruders do not even try to delete the file containing the history of their commands or do not try to deactivate this history function. Upon 38 intrusions, only 14 were cleaned by the intruders (11 have deactivated the history function and 3 have deleted the history file). This means that 24 intrusions left behind them a perfectly readable summary of their activity within the honeypot. No intruder has tried to check the presence of VMware software, which may be a sign that the intruder is observed.

None of the publicly known methods to identify the presence of VMware software [5][4] was tested. This probably means that the intruders are not experienced hackers.

Bibliography

[1] Home page of Leurré.com: http://www.leurre.org.

[2] E. Alata, M. Dacier, Y. Deswarte, M. Kaaniche, K. Kortchinsky, V. Nicomette, Van Hau Pham and Fabien Pouget. Collection and analysis of attack data based on honeypots deployed on the Internet. In *QOP 2005, 1st Workshop on Quality of Protection (collocated with ESORICS and METRICS), September 15, 2005, Milan, Italy, Sep 2005*.

[3] Michael Bailey, Evan Cooke, Farnam Jahanian and Jose Nazario. The Internet Motion Sensor - A Distributed Blackhole Monitoring System. In *NDSS, 2005*.

[4] Joseph Corey. Advanced Honey Pot Identification And Exploitation. In *Phrack, Volume 0x0b, Issue 0x3f, Phile #0x01 of 0x0f*, 2004, http://www.phrack.org.

[5] T. Holz, and F. Raynal. Detecting honeypots and other suspicious environments. In *Systems, Man and Cybernetics (SMC) Information Assurance Workshop. Proceedings from the Sixth Annual IEEE*, pages 29-36, 2005.

[6] http://www.dshield.org. Home page of the DShield.org Distributed Intrusion Detection System, http://www.honeynet.org.

[7] M. Dacier, F. Pouget, and H. Debar. Honeypots: practical means to validate malicious fault assumptions. In *Dependable Computing, 2004. Proceedings. 10th IEEE Pacific Rim International Symposium*, pages 383-388, Tahiti, French Polynesia, 3-5 March 2004.

[8] F. Pouget. Publications web page: http://www.eurecom.fr/~pouget/papers.htm.

[9] Fabien Pouget, Marc Dacier, and Van Hau Pham. Understanding threats: a prerequisite to enhance survivability of computing systems. In *IISW'04, International Infrastructure Survivability Workshop 2004, in conjunction with the 25th IEEE International Real-Time Systems Symposium (RTSS 04) December 5-8, 2004 Lisbonne, Portugal*, Dec 2004.

[10] Fabien Pouget, Marc Dacier, and Van Hau Pham. Leurre.com: on the advantages of deploying a large scale distributed honeypot platform. In *ECCE'05, E-Crime and Computer Conference, 29-30th March 2005, Monaco, Mar 2005.*

[11] CAIDA Project. Home Page of the CAIDA Project, http://www.caida.org.

[12] EnergyMech team. EnergyMech. Available on: http://www.energymech.net.

[13] Inc. VMware. Available on: http://www.vmware.com.

Identifying the Source of Messages in Computer Networks

Marios S. Andreou

University of Newcastle upon Tyne

Abstract

A brief overview of the mechanisms relevant to the creation and use of network messages carrying forged address data, with a synopsis of current proposals for countering this threat (IP traceback). A proposal is made for the investigation of message traceback in the context of other network environments, typical to intranets, (such as switched Ethernet), as an extension of current work

1. Introduction

Modern computer networks are predominantly built with "packet switched" (aka "connectionless") delivery services; the Internet Protocol for example provides a connectionless service. The nature of a such services is that each message is routed independently of other messages forming part of the same exchange. No information regarding a message is stored on the relay machine; instead, the complexity of the delivery service is pushed to the edge of the network (and specifically to the layered network services of the sending and receiving machines). In other words, the delivery service itself is *stateless*.

Thus, assuming there is correct delivery of the message, and restricting our view only to the delivery service, the recipient has no means with which to verify that the source address of a message actually belongs to the originating machine. Specially crafted messages that contain false address data are sometimes termed "spoofed" messages, and they can facilitate a number of malevolent objectives.

A fabricated message introduced to the delivery service will not automatically contain the higher level data expected by the recipient network service(s). Instead, the attacker must make an informed guess as to what data each of the receiving machine's networking layers expects and fabricate the message accordingly. The magnitude of this challenge depends on the scenario in which the spoofed message is to be used. Generally, we can consider three cases; a single initial message, a single message as part of an ongoing exchange, and multiple successive messages. The second and third cases pose a bigger problem for the attacker, as she will need to fashion the forged message appropriately for the receiving machine to accept it as "legitimate".

We can also distinguish between *blind* and *non – blind* attacks. Typically, in any forged message scenario, three entities exist; the attacker, the victim, and the tertiary (masqueraded) host. When a forged message is replied to, it is sent to the source address reported in the original "spoofed" message. In a "blind" scenario, the attacker cannot trivially access these replies, complicating his task of fabricating a subsequent message. If however the attacker is on the same subnet as the machine whose address was used (i.e. the tertiary masqueraded host) then she can more easily access the victim's responses *to that* machine, and so we have "non blind spoofing" (assuming a traditional shared medium network, such as "classic" Ethernet).

2. Consequences

Whilst obscuring the source of potentially incriminating messages is advantageous in itself, IP packets with forged addresses have been used to facilitate a number of malevolent objectives. Spoofed messages are mainly associated with Denial of Service (DOS) attacks. These attacks *deny service* by specifically targeting a vulnerable process so that the services provided by that process (or by other, dependant processes) are discontinued or at least hindered. A slight variation is a *Distributed* DOS (DDOS), in which the source of the attack is not a single machine but any number of machines under the direction of the attacker. There are a number of illustrative examples where spoofed messages have been used to mount DOS attacks, such as Land [Lan97], La-Tierra [Huo98], Smurf [Smu97], ARP [Spu89], ICMP [Bel01], and of course SYN Flood [Syn00]. An important point to be made is that all these attacks fall at the "easier" end of the scale (from the attacker's point of view) as they are all instigated by a single initial spoofed packet. Spoofed messages can and have also be used for other purposes. An example is the elaborate "idle scan" method which allows an attacker to scan for open ports on a potential victim without sending any packets containing its true source I.P. address [San98].

3. Solutions

Varied suggestions have resulted from work in this area. "Defeating" spoofed messages can mean a number of things and proposals typically fall into the categories of prevention, detection and deterrence.

Perhaps the best preventive measures are to improve and strengthen the affected network protocols themselves [Bel89]. However, the difficulties associated with homogeneous deployment and implementation (e.g. agreeing on new standards, or the costs involved in upgrading/exchanging hardware) assure their practical infeasibility in the shorter term.

There are many proposals for implementing systems that facilitate the determination of the “true” source IP address of a packet (or stream of packets), possibly to a known degree of accuracy. These are commonly referred to as “IP Traceback” systems and with respect to “defeating” spoofed messages are regarded as a promising area of research. A traceback system capable of accurate and timely discovery of the true source address of a given packet could be used as a deterrent of malicious network activity. Lee et al envisage IP traceback having a number of applications, such as attack reaction, attack prevention, and establishment of liability (even prosecution) [Lee01].

4. Current and Future Plans

A major shortcoming of current proposals is that they can only trace as far back as the first router connecting to the attacker’s subnet hardware. Typically, the network hardware from which an attack originated may not even use IP (e.g. switched Ethernet). Oe et al consider the problem with implementing IP traceback to be the composition of the Internet, as interconnected autonomous systems (AS)[Oe03]. Each connected network is independently and uniquely architected, implemented and administered. They thus propose a two-part solution to the IP traceback problem, modelled after the Internet routing hierarchy (i.e. Exterior and Interior gateway protocols). The two part traceback architecture is further explored and developed by Hazeyama et al [Haz03], who distinguish between *inter*domain and *intra*domain IP traceback. This decoupling allows the expression (and subsequent satisfaction) of the unique requirements for intradomain traceback (i.e. within an autonomous system). For instance, “precision” requirements are greater within the AS (in that you ultimately aim to discover a single originating machine address). Hazeyama et al make the observation that in the case of DOS style attacks, traffic flows are typically greater around the victim than around the attacker. Thus, approaches such as link testing or probabilistic approaches such as packet marking and messaging, are unsuitable for tracing *within* the originating AS. Their proposal extends traceback to level 2 of the OSI network stack, which stores identifiers together with packet audit trails on gateway (or *leaf*) routers [Haz03].

The inter/intra network traceback approach is very interesting and lends a greater degree of credibility to the ultimate goal of tracing spoofed messages to their actual source. This approach realises the limitations of current inter-network traceback proposals, whilst recognising their importance in determining the source *network*.

Given the heterogeneity of private networks connected to the Internet, an intranet traceback system will need to be tailored to that network (or at least to that network *type*). This intranet traceback system will need to provide an interface to an inter-network traceback system, with the two systems together providing the end to end message traceback. Aside from tracing network attacks, such as system would be a useful added security measure for an organisation, and in legal terms affords the ability to establish individual liability. It is typical that some form of security related audit on network traffic is already implemented in most existing networks, though they may not currently maintain state regarding traffic origins.

Two switched Ethernet computer clusters at the School of Computing Science will serve as the subject for the exploration of intranet message traceback. The *Rack* and *Mill* clusters each consist of two CISCO switches connected by a gigabit Ethernet link. The sixty odd machines in each cluster are provisioned across the two switches (the *Rack* has Windows boxes and the *Mill* Linux). A dedicated linux machine has been installed on one of the Mill switches, to serve as the spoofing node. One of the ultimate objectives of this investigation is to achieve a better understanding of the traceback problem through its generalisation (i.e. consider *message* traceback and not just *IP packet traceback*).

5. References

[Bel89] Bellovin, S., *Security Problems in the TCP/IP Protocol Suite*, Computer Communication Review, 19(2), p.32-48, 1989.

[Bel01] Bellovin, S., Leech, M., Taylor, T., *ICMP Traceback Messages*, IETF Internet Draft, October 2001.

[Bur00] Burch, H., Cheswick, B., *Tracing Anonymous Packets to Their Approximate Source*, Proceedings 14th Sys. Admin Conference (LISA 2000), p.319-327, December 2000.

[Haz03] Hazeyama, H., Oe, M., Kadobayashi, Y., *A Layer-2 Extension to Hash Based IP Traceback*, IEICE Transactions on Information and Systems, E86(11), p.2325, November 2003.

[Huo98] Huovinen, L., Hursti, J., *Denial of Service Attacks: Teardrop and Land*, http://users.tkk.fi/~lhuovine/study/hacker98/dos.html

[Lan97] “m3lt”, *The LAND Attack (IP DOS)*, http://www.insecure.org/sploits/land.ip.DOS.html, 1997.

[Lee01] Lee, S.C., Shields, C., *Tracing the Source of Network Attack: A Technical, Legal and Societal Problem*, Proceedings 2001 IEEE Workshop on Information Assurance and Security, p.239-246, June 2001.

[Oe03] Oe, M., Kabobayashi, Y., Yamaguchi, S., *An Implementation of a Hierarchical IP Traceback Architecture*, Workshop Proceedings IEEE Symposium on Applications and the Internet 2003, p.250, January 2003.

[San98] Sanfillipo, S., *New TCP Scan Method*, http://seclists.org/lists/bugtraq/1998/Dec/0079.html, 1998.

[Sav01] Savage, S., Wetherall, D., Karlin, A., Anderson, T., *Network Support for IP Traceback*, IEEE/ACM Transactions on Networking, 9(3), p226-237, 2001.

[Smu97] "Tfreak", *Smurf multi broadcast ICMP attack*, http://seclists.org/lists/bugtraq/1997/Oct/0066.html, 1997.

[Sno01] Snoeren, A.C., Partridge, C., Sanchez, L.A., Jones, C.E., Tchakountio, F., Schwartz, B., Kent, S.T., Strayer, W.T., *Single Packet IP Traceback*, IEEE/ACM Transactions on Networking, Vol 10(6), p.721-734, December 2002.

[Spu89] Spurgeon, C., *Broadcast Storms and Packet Avalanches on Campus Networks*,http://madhaus.utcs.utoronto.ca/cns/ftp/doc/introductory_documents/storms.txt, 1989.

[Sto00] Stone, R., *Centertrack: An IP Overlay Network for Tracking DOS Floods*, Proceedings 9th USENIX Security Symposium, US

[Syn00] CERT, *CERT® Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks*, http://www.cert.org/advisories/CA-1996-21.html, November 2000.

Vulnerability Assessment Through Attack Injection

João Antunes

jantunes@di.fc.ul.pt

University of Lisboa, Portugal

Abstract

Our reliance on computer systems for everyday life activities has increased over the years, as more and more tasks are accomplished with their help. The increasing complexity of the problems they address also require the development of more elaborated solutions. So, applications tend to become larger and more complex. On the other hand, the ever present tradeoff between time to market and thorough testing puts pressure on the quality of the software. Hence, applications tend to be released with little testing, so software bugs are continuously detected afterwards, resulting in security vulnerabilities that can be exploited by malicious adversaries and compromise the systems' security. The discovery of security vulnerabilities is then a valuable asset in the development of dependable systems. AJECT is presented as a new tool for vulnerability assessment, without requiring access to the source code or any updated database vulnerability. Preliminary experimental results in IMAP servers showed that AJECT was able to discover not only all known vulnerabilities, but also a previously unknown one.

1. Overview

Software evolved over the years. It became more useful and easier to make, but on the other hand it also became more complex, requiring bigger development teams. This increasing complexity has lead to the creation of larger applications with much more lines of code. Also, the competitive software market requires applications to be deployed with full functionality as soon as possible. Hence, applications tend to be released with little testing, so software bugs are continuously detected afterwards. These software bugs have also evolved and are more sophisticated. There are now many new and different ways in which software could be exploited.

Also, the nature of the software and reliance we place in it makes us vulnerable to any deviation of its correct behaviour, such as in safety-critical systems or e-banking. Dependability in this systems is of paramount importance and a security compromise potentially catastrophic.

If we could develop error-free software, vulnerabilities would not exist and dependability would surely be increased. Actually, without vulnerabilities applications could not be exploited. So, theoretically, if we could devise methods and means to remove these vulnerabilities or even prevent them to appear in the first place, we should be able to create dependable software with inviolable security properties.

We propose attack injection with extended monitoring capabilities as a method for detecting vulnerabilities. In this method we try to identify invalid software states just like an attacker would – trial and error, by consecutively attacking its target. This means we are not dependent on a database of known vulnerabilities, but rather we rely on a more generic set of tests. Through careful and automated monitoring we can later analyze the results of the attacks and pinpoint the exact vulnerability. This allows us to detect known and unknown vulnerabilities in an automated fashion.

2. Attack Injection

The AVI (attack, vulnerability, intrusion) composite fault model introduced in [1, 6] helps us to understand the mechanisms of failure due to several classes of malicious faults. This specialization of the well-known sequence of *fault Æ error Æ failure* applied to malicious faults, limits the fault space of interest to the composition (*attack* + *vulnerability*) *Æ intrusion*. *Attacks* are malicious activities perpetrated by an adversary with the objective of violating the system's security properties. Coding or configuration errors are other type of faults which may insert *vulnerabilities* in the system. These faults can be introduced accidentally or deliberately, with or without malicious intent, but they only compromise the system if an attack successfully activates a vulnerability, hence leading to a *intrusion*. This further step towards failure is normally succeeded by the production of an erroneous state in the system (e.g., a root shell, or new account with root privileges), and if nothing is done to process the error, a failure will happen.

We propose to emulate an attacker's behaviour in order to activate the existing vulnerabilities, therefore detecting them. This idea was materialized in an attack injection tool called *AJECT*. This tool emulates the behaviour of an external adversary attempting to cause a failure in the target system. First, it generates a large number of attacks which it directs against the target's interface. These attacks are expected to be deflected by the validation mechanisms implemented in the interface, but if some of them succeeds in exploiting a vulnerability, an attack/vulnerability combination was found. Some conditions can amplify the attack's probability of success, for example: a correct understanding of the interaction protocol employed by the target facilitates the creation of more efficient attacks (e.g., reduces the number of random tests); and a good knowledge about what type of vulnerabilities appear more frequently also helps to prioritize the attacks.

3. Attack Monitoring

Though attack injection is one of the most important aspects of the AJECT tool, it also has some strong monitoring capabilities. One does not only needs to inject attacks in the target's system, it must also observe and record its reaction. This is essential for the purpose of automating an effective detection of vulnerabilities.

Thus, while attacks are being carried out, AJECT monitors how the state of the target system is evolving, looking for errors or failures, tracing the process execution, or its resource usage. Whenever one these problems is observed, it indicates that a new vulnerability has potentially been discovered. Depending on the collected evidence, it can indicate with more or less certainty that a vulnerability exists. For instance, there is a high probability of the presence of a vulnerability if the system crashes during (or after) the attack – this attack at least compromises the availability of the system. On the other hand, if what is observed is an abnormal resource usage, such as the creation of a large file, or an increasing memory allocation, though it might not be a vulnerability it still needs to be further investigated.

4. Attack Tests

The success of the injections depends greatly on the attacks themselves. If one had access to the source code, one could generate carefully crafted attacks, in order to get a maximum test coverage [8, 7, 4]. But that would be human dependent and an intensive task to undertake. One could also inject previously created attacks for known vulnerabilities from a knowledge database, such as the vulnerability scanners [2, 3, 5]. Only that would merely enable us to detect known vulnerabilities, leaving the application susceptible to new ones. It would also require a continuously updated database.

Since AJECT is neither dependent on the application's source code, nor on a database of known vulnerable versions and respective exploits, it is not limited to previously detected vulnerabilities and can be used with virtually any server application. Then how is the attack generation performed? There's almost an infinite number of combinations that can constitute an attack. One can always stress testing each and every protocol message, with random and almost infinite permutations. But how can one generate a more manageable number attacks, maintaining a large overall coverage? AJECT relies its detection capabilities on a more generic set of tests, such as value, syntax, or information disclosure tests. This tests are specific enough so that we don't go to the point of testing all possible combinations (i.e., only experiment with a careful subset of malicious strings as opposed to all possible character combinations), but are also very generic as to create a large number of different attacks, thus achieving a larger coverage (i.e., test the syntax of the protocol by injecting invalid messages, permutating, adding, or removing its fields).

5. Conclusions and Future Work

AJECT was experimented against different IMAP servers in order to evaluate its method for vulnerability assessment. The servers were chosen from the reported vulnerabilities in several bugtraq and security sites. There is at least one attack that can exploit each of these vulnerable servers, which can be manually confirmed by the respective posted exploit. The objective of the experiment was that from all the attacks generated by AJECT's tests, at least one should be able to replicate the exploit's effects, or some server's reaction that could point out the existence of a vulnerability. AJECT successfully detected the 2005's vulnerabilities of the following products: MailEnable Professional and Enterprise Edition, GNU Mailutils, TrueNorth eMailServer, Alt-N MDaemon, University of Washington Imap, Floosietek FTGate, and Qualcomm Eudora WorldMail Server. The vulnerabilities found ranged from *buffer overflows* to *information disclosure*.

As an unplanned secondary objective, AJECT was also able to discover a new and unknown vulnerability in one of the servers, thus confirming its usefulness. A specific IMAP command in the last version of TrueNorth eMailServer identified a buffer overflow vulnerability.

Currently, we are investigating new ways in generating efficient attacks and enhancing its monitoring capabilities. Perhaps access to the server's source code could leverage AJECT's capabilities, allowing us to reckon, and improve, its test coverage.

References

[1] A. Adelsbach, D. Alessandri, C. Cachin, S. Creese, Y. Deswarte, K. Kursawe, J. C. Laprie, D. Powell, B. Randell, J. Riordan, P. Ryan, W. Simmonds, R. Stroud, P. Veríssimo, M. Waidner, and A. Wespi. *Conceptual Model and Architecture of MAFTIA. Project MAFTIA deliverable D21*. Jan. 2002. http://www.research.ec.org/maftia/deliverables/D21.pdf.

[2] D. Farmer and E. H. Spafford. The COPS security checker system. In *Proc. of the Summer USENIX Conference*, pages 165–170, June 1990.

[3] FoundStone Inc. FoundStone Enterprise, 2005. http://www.foundstone.com.

[4] E. Haugh and M. Bishop. Testing C programs for buffer overflow vulnerabilities. In *Proc. of the Symposium on Networked and Distributed System Security*, Feb. 2003.

[5] Tenable Network Security. Nessus Vulnerability Scanner, 2005. http://www.nessus.org.

[6] P. Veríssimo, N. F. Neves, and M. Correia. The middleware architecture of MAFTIA: A blueprint. In *Proceedings of the Third IEEE Information Survivability Workshop*, Oct. 2000.

[7] J. Viega, J. T. Bloch, Y. Kohno, and G. McGraw. ITS4: A static vulnerability scanner for C and C++ code. In *Proc. of the 16th Annual Computer Security Applications Conference*, Dec. 2000.

[8] D. Wagner, J. S. Foster, E. A. Brewer, and A. Aiken. A first step towards automated detection of buffer overrun vulnerabilities. In *Proc. of the Network and Distributed System Security Symposium*, Feb. 2000.

Adaptive Security

Christiaan Lamprecht

School of Computing Science

University of Newcastle upon Tyne

C.J.Lamprecht@ncl.ac.uk

Security that can change itself, or the environment, in response to system changes or threats has the potential to be a more effective security solution than current static solutions. This paper shows that context aware security is at the forefront of such developments and in particular focuses on resource aware security. It also proposes considerations for future research.

Introduction

The research area of Adaptive Security recognises the fact that “one size fits all” security solutions fail to address the complexity of modern day computing systems and the environments in which they operate. The solutions lack the desired expressiveness to take relevant environmental factors into account, the desired flexibility to accommodate for various runtime environmental changes and the ability to remain accurate and relevant as the security requirements of the system change over time. Appropriate and timely security adaptation based on these factors are the concern of Adaptive Security.

In this paper we will first provide an overview of current Adaptive Security research, then examine our particular focus area and finally provide some details on future work and considerations.

1. Context aware adaptive security

Adaptive Security aims to provide timely responses in anticipating, effectively controlling the effects and initiating appropriate countermeasures to changes in the system or its environment. Research solutions in the area endeavours to provide effective automation of this process by considering contextual information that can be exploited to deal with the demanding requirements of the domain of application.

Examples of such research include adaptive access control policies where policies incorporate application specific information in policy decisions [Bez2002], adaptive intrusion detection systems which allow individual trust management to conserve processor resources [Venk1997], adaptive agents where the system itself moves between different domains and has to detect and adapt to various malicious scenarios [Alam2003], adaptive security in resource constrained networks where appropriate security protocols are selected at runtime based on the current network conditions [Chigan2005][Schn1998] and threats [Hinton1999], adaptive security infrastructures (ASI) where the ASI consists of many security systems which cooperate to ensure minimal policy conflicts [Marcus2003][Shnitko2004] and many more.

2. Resource aware adaptive security

Resource availability, in particular processing resources, is a contextual factor which is often overlooked during the adaptation process. Changing the security mechanisms deployed can have a significant impact on the performance of a system. This is often true as these mechanisms are typically computationally intensive and consequently resource hungry processes. This is of particular concern for systems with real-time constraints such as; stock trading or online bidding services where the nature of the data accessed imposes real-time constraints on the transaction, military systems where the timely arrival of essential data is a significant cost factor in deciding the level of security required or embedded safety critical systems (such as cars, airplanes, etc.) where task deadlines are of particular concern. In such systems the demand security places on the system resources need to be quantified and traded off against the available resources to provide acceptable functionality as well as an appropriate level of security.

Bearing in mind the aforementioned complexity of modern day computing systems we believe that such a trade-off could also provide significant benefits for everyday systems which do not necessarily have such rigorous real-time constraints. Though correct system functionality is not at stake, security still has a significant impact on the performance of the system. This is of particular concern as it affects, among other things, the quality of service (QoS) provided as well as the extent of service exploitation in terms of the maximum number of customers that can be supported at any one time.

Adaptive security, based on a security-performance trade-off, shows potential in providing adequate security based on the current system and client needs. By varying the level of security appropriately benefits could potentially include consistent QoS provided to customers even at peak system loads, support for more customers at peak loads, support for clients on various resource constrained devices, improved security when system load is low and additional system resilience against unanticipated changes in the environment (e.g. Denial of Service (DoS) attacks, resource failure and changes in system load) This is in sharp contrast to pre-deployment decisions which, considering the complexity of the system, often fail to take into account all the environmental factors that will influence the system and so either burden the system with unneeded security or expose it through inadequate security measures.

Care has to be taken however that adapting security solely based on system performance does not overlook the principle purpose of security in that it keeps the system secure! An attacker could mount a preliminary DoS attack to burden the system and thus, in order to maintain QoS levels, force the system to lower the security and therefore potentially assist the eventual attack. Additional steps need to be taken in order to restrict the extent of adaptivity based on, for example, minimum security requirements, data worth, data channel exposure, client requirements, threats etc… Further considerations are mentioned in the Research Issues section.

3. Current research

Though many areas of research do consider performance as an important factor and so try to make security mechanisms more efficient, they do not explicitly consider its cost and the effect it has on the system or service as a whole. Examples of research which address some of these issues include [Schn1998] which investigates how various authentication heuristics influences the timely signing/verifying of streamed data over a network for clients with varying resource availabilities. [Cho2005] models, using stochastic Petri nets, an intrusion detection system in a wireless setting and investigates how often intrusion detection should occur by considering the mean time to security failure and the cost of changing the group key when malicious nodes are evicted. [Son2000] considers a few levels of authentication and confidentiality guarantees and how to most effectively adapt the application of these to meet transaction deadlines for real-time transactions. [Chigan2005] explores the cost of using particular security protocol sets (at different layers of the stack), chosen due to threats, under different node mobility patterns in a simulated ad hoc network environment.

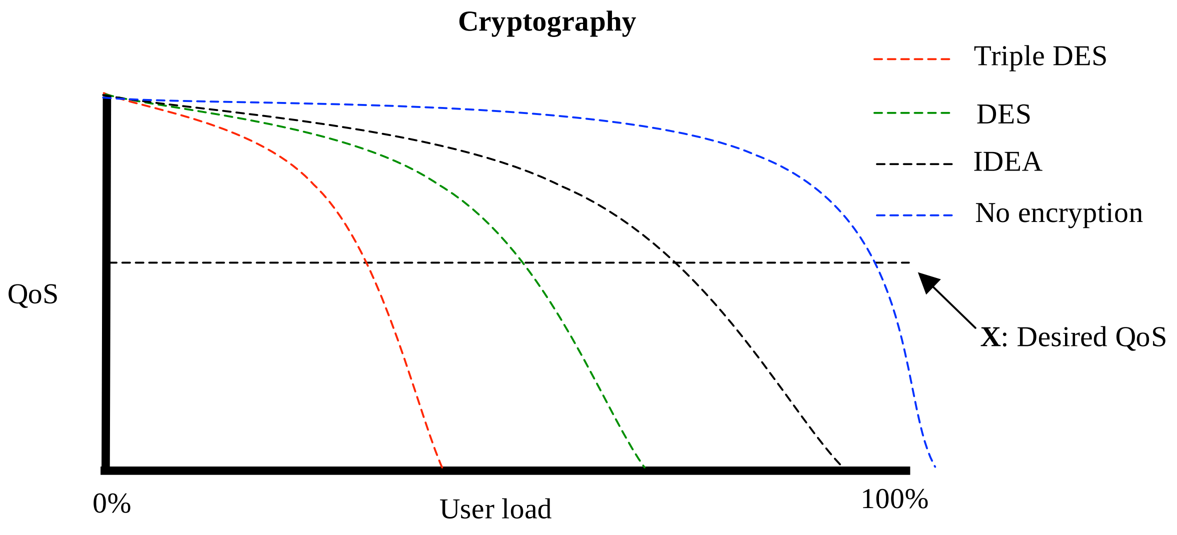
The above mentioned research each address a small subset of the related issues to various degrees and depths. Much potential research still remains.

Focus

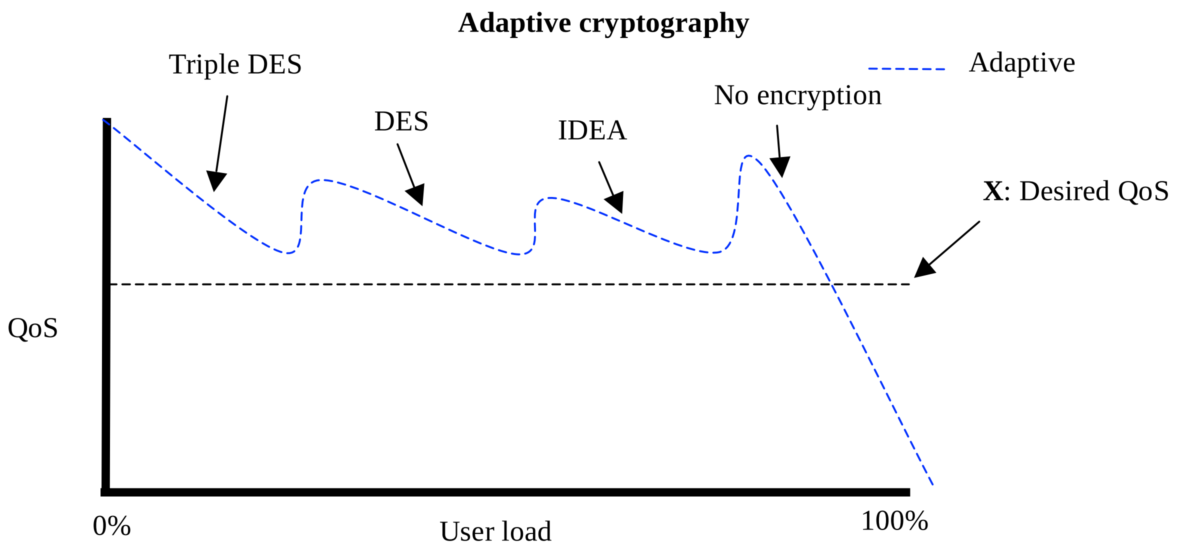
Our particular focus will be on explicitly reasoning about the effect security has on the system in order to aid in satisfying some higher level QoS goals. This will be achieved through quantifying some security-performance trade-off.

High level example;

"Provide an appropriate QoS level to clients whilst maintaining security at the maximum level the system can afford." In this example average client QoS, i.e. response time, will be maintained above X by varying the security level, i.e. encryption level, in response to user load.



The graph above shows how transaction response time, under three encryption protocols and no encryption, would typically be affected by user load. Triple DES is the most secure and the most resource hungry whilst "no encryption" provides no security but requires no additional resources.



The above graph shows how a suitable average client QoS can be maintained by changing the encryption to one which is requires less resources before the QoS guarantee is violated.

Of course it would be desirable not to include "no encryption" in the set of protocol choices. The research issues section below highlights some issues which require careful consideration.

4. Research issues

To successfully address all the issues relating to our research focus, several categories of research issues need to be considered:

- Trade-off metrics

    Metrics for security

    Metrics for; system resource availability, value of data, real-time constraints, client resource availability, threats, etc…

- System architecture

    Seamlessly changing the security protocol at runtime is non-trivial and also incurs an additional, though temporary, resource cost which need to be considered.

    Performance of particular security protocol implementations might bear little resemblance on their expected cost. [Lamp2006]

- Modelling formalisms will play a large role in the types of deductions that can be made about the system being modelled.

    Probabilistic modelling

    Formal modelling

References

[Alam2003]	Alampalayam S.P., Anup Kumar, An adaptive security model for mobile agents in wireless networks, *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, Volume 3, 1-5 Dec. 2003 Page(s):1516 - 1521 vol.3, Digital Object Identifier 10.1109/GLOCOM.2003.1258491
[Bez2002]	Konstantin Beznosov, Object Security Attributes: Enabling Application-specific Access Control in Middleware, *In proceedings of the 4th International Symposium on Distributed Objects & Applications (DOA)* pp. 693-710, Irvine, California, October 28 - November 1, 2002
[Chigan2005]	Chunxiao Chigan, Leiyuan Li and Yinghua Ye, Resource-aware self-adaptive security provisioning in mobile ad hoc networks, *Wireless Communications and Networking Conference,* 2005 IEEE Volume 4, 13-17 March 2005 Page(s):2118 - 2124 Vol. 4, Digital Object Identifier 10.1109/WCNC.2005.1424845
[Cho2005]	Jin-Hee Cho and Ing-Ray Chen, On design tradeoffs between security and performance in wireless group communicating systems, *Secure Network Protocols, 2005. (NPSec). 1st IEEE ICNP Workshop*, 6 Nov. 2005 Page(s):13 - 18 Digital Object Identifier 10.1109/NPSEC.2005.1532047
[Hinton1999]	Hinton H., Cowan C., Delcambre L. and Bowers S., SAM: Security Adaptation Manager, *Computer Security Applications Conference, 1999. (ACSAC '99) Proceedings. 15th Annual,* 6-10 Dec. 1999 Page(s):361 - 370 Digital Object Identifier 10.1109/CSAC.1999.816047
[Kep2003]	Jeffrey O. Kephart, David M.Chess, The vision of Autonomic computing, *IBM Thomas J. Watson Research Center, Published by the IEEE Computer Society,*0018-9162, pages 41-50, Jan 2003
[Lamp2006]	C. Lamprecht, A. van Moorsel, P. Tomlinson and N. Thomas, Investigating the efficiency of cryptographic algorithms in online transactions, *International Journal of Simulation Systems, Science & Technology, Special issue on Performance Engineering*, Vol.7 No. 2, pages 63-75, February 2006.
[Marcus2003]	Marcus L., Local and Global Requirements in an Adaptive Security Infrastructure, *In Proceedings of International Workshop on Requirements for High Assurance Systems*, September 2003.

[Schn1998] Schneck P.A. and Schwan K, Dynamic authentication for high-performance networked applications Quality of Service, *(IWQoS 98) 1998 Sixth International Workshop*, 18-20 May 1998 Page(s):127 - 136, Digital Object Identifier 10.1109/IWQOS.1998.675229

[Shnitko2004] Shnitko A, Practical and Theoretical Issues on Adaptive Security, *Proceedings of FCS'04 Workshop on Foundations of Computer Security, Workshop on Logical Foundations of an Adaptive Security Infrastructure*, June 2004.

[Son2000] Son S.H., Zimmerman R and Hansson J., An adaptable security manager for real-time transactions, *Real-Time Systems, 2000. Euromicro RTS 2000. 12th Euromicro Conference*, 19-21 June 2000 Page(s):63 - 70, Digital Object Identifier 10.1109/EMRTS.2000.853993

[Venk1997] Venkatesan, R.M. and Bhattacharya, S., Threat-adaptive security policy, *Performance, Computing, and Communications Conference, 1997. IPCCC 1997., IEEE International*, 5-7 Feb. 1997 Page(s):525 – 531, Digital Object Identifier 10.1109/PCCC.1997.581559

ScriptGen: using protocol-independence to build middle-interaction honeypots

Corrado Leita - Institut Eurécom

Introduction

One of the most recent and interesting advances in intrusion detection consists in the honeypot technology. As the word suggests, honeypots aim at attracting malicious activity, in order to study it and understand its characteristics. L.Spitzner in [1] defined a honeypot as "a resource whose value is being in attacked or compromised". From a practical point of view, honeypots are hosts not assigned to any specific function inside a network. Therefore, any attempt to contact them can be considered as malicious. Usually, traffic observed over any network is a mixture of "normal" traffic and "malicious" one. The value of a honeypot resides in the ability to filter out the first kind, allowing to perform an in-depth study of the latter.

The main design issue in deploying honeypots is defining the degree of interaction allowed between the honeypot and the attacking client. A possible choice might consist in allowing full interaction, using a real operating system eventually running on a virtualization system such as VMware[1] (high interaction honeypots). Such a solution allows the maximum degree of verbosity between the attacker and the honeypot. However, this solution leads to a number of drawbacks. Since the honeypot is a real host, it can be successfully compromised by an attack. After a successful exploit, an attacker can even use this host as a relay to perform attacks against other hosts of the network. This underlines the maintenance cost of the high interaction honeypots: these hosts have to be kept under control in order to avoid being misused. Also, this solution is resource consuming: even using virtualization softwares, the resource consumption of each deployed honeypot is high.

[1] www.vmware.com

A less costly solution consists in using much simpler softwares than a whole operating system, mimicking the behavior of a real host through a set of responders (low interaction honeypots). An example of these softwares is honeyd [2]. Honeyd uses a set of scripts able to react to client requests and provide an appropriate answer for the client. These scripts are manually generated, and require an in-depth knowledge of the protocol behavior. This process is therefore tedious, and sometimes even impossible: the protocol specification may not be publicly accessible. As a result, not many scripts are available and their behavior is often oversimplistic. This significantly decreases the amount of available information. For many protocols, no responder script is available and therefore the honeypot is only able to retrieve the first client request, failing to continue the conversation with the client. Many known exploits send the malicious payload only after a setup phase that may consist of several exchanges of request/replies inside a single TCP session. Therefore in these cases low-interaction honeypots do not collect enough information to be able to discriminate between the different malicious activities.

1. The ScriptGen idea

The previous section clearly identifies a trade-off between cost and quality of the dialogs with the client. Both high interaction and low interaction solutions provide clear advantages and not negligible disadvantages. For this reason, we tried to find a hybrid solution able to exploit some of the advantages of both approaches. This led to the development of the ScriptGen approach.

The main idea is very simple: to observe the behavior of a real server implementing a certain protocol, and use these observations as a training set to learn the protocol behavior. In order to be able to handle also those protocols whose specification is not publicly available, we target a powerful and apparently ambitious objective: that is, we aim at complete protocol independence. We do not make any kind of assumption on the protocol behavior, nor on its semantics, avoiding to use any kind of additional context information. We parse as input an application level dialog between a client and a server as a simple stream of bytes, and we build from this a representation of the protocol behavior, performing an inference of some of the protocol semantics.

It is important to notice that we do not pretend to be able to learn automatically the whole protocol language: this would be probably impossible. Our goal is more modest, and consists in correctly carrying on conversations with attack tools. This dramatically simplifies the problem since the requests are generated by deterministic automata, the exploits. They represent a very limited subset of the total input space in terms of protocol data units, and they also typically exercise a very limited number of execution paths in the execution tree of the services we want to emulate.

ScriptGen is a framework composed of two different functional parts:

- An offline analysis, that consists in inferring the protocol semantics and behavior starting from samples of protocol interaction.
- An online emulation, that uses the informations inferred in the preceding phase to carry on conversations with the clients in the context of a honeypot. ScriptGen is also able to detect in this phase deviation from the already known behaviors, triggering alerts for new activities and reacting to them.

2.1 The stateful approach and the need for semantic inference

Starting from samples of conversation with a real server implementing the protocol, ScriptGen must produce a representation of the protocol behavior to be used in the emulation phase. ScriptGen follows a stateful approach, representing this information through a finite state automata. Every state has a label and a set of outgoing edges, leading to future states. Each of these edges is labeled too. During emulation, when receiving a request the emulator tries to match it with the transition labels. The transition that matches will lead to the next future state, and state's label will be used to generate the answer to be sent back to the client. The state machine therefore represents protocol's language as observed in the samples.

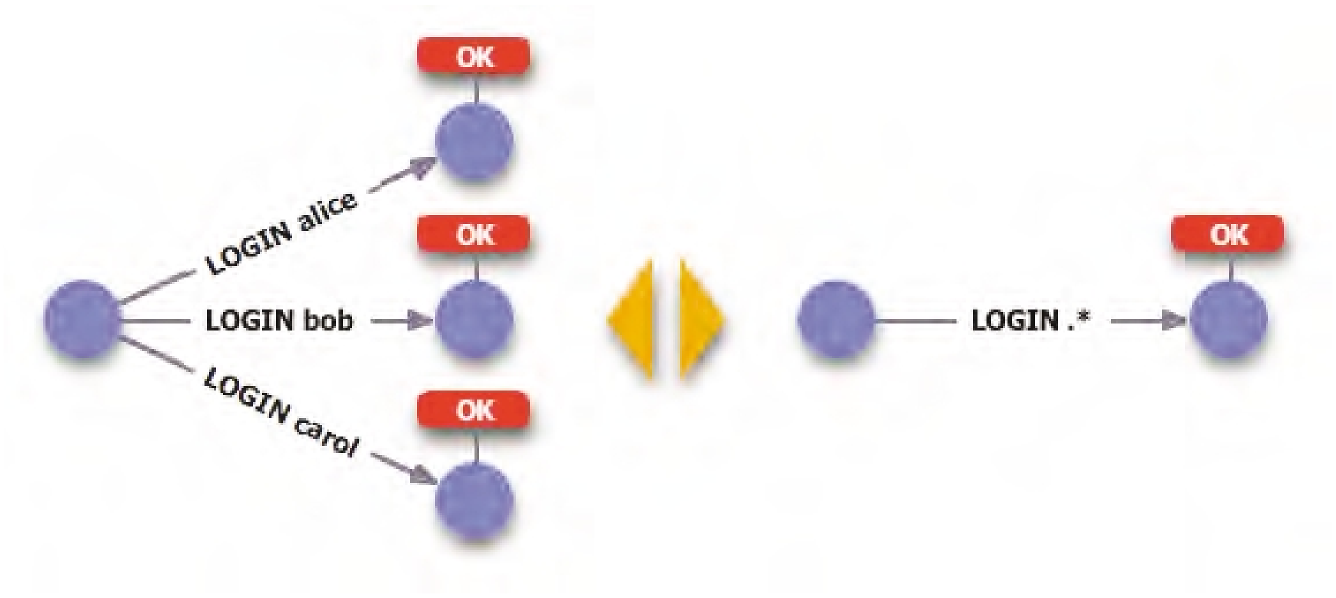


Figure 1 – Semantic abstraction

Building such a state machine without any knowledge about semantics would not generate useful emulators. The requests would be seen as simple streams of bytes, and every byte would receive the same treatment. Referring to the example shown in figure 1, two login requests with different usernames would lead to two different paths inside the state machine. During emulation, a login request with a username never seen before would not find a match.

The previous example is just one of the examples that show the need to rebuild some of the protocol semantics, exploiting the statistical variability of the samples used for training. This is possible through the region analysis algorithm, a novel algorithm that we developed and that is detailed in [3]. Region analysis allows, through bioinformatics algorithms, to aggregate the requests into clusters representing different paths in the protocol functionality. For each cluster, the algorithm produces a set of fixed and mutating regions. A fixed region is a set of contiguous bytes in the protocol request whose content can be considered as discriminating from a semantic point of view (referring to the previous example, the "LOGIN" command). A mutating region is a set of contiguous bytes whose value has no semantic value (for instance, the username). The succession of fixed and mutating regions generates regular expressions used as labels on the transitions to match the incoming requests, taking into consideration a simple notion of semantics.

2.2 Intra-protocol dependencies

The semantic abstraction produced through the region analysis algorithm allows to detect mutating fields that do not add semantic value to the client request. Some of those fields, however, incorporate a more complex semantic value that cannot be detected by looking at the samples of client requests. For instance, many protocols contain cookie fields: the value of these fields is chosen randomly by either the client or the server, and must be repeated in the following messages of the protocol conversation. Two situations can be identified:

1. The client sets the cookie in its request, and the value must be reused in the server answer. In this case the emulator must be able to retrieve the value from the incoming request and copy it in the generated answer.

2. The server sets the cookie in its answer, and the client must reuse the same value for the following requests to be accepted. From the emulator point of view, this does not generate any issue. The server label will contain a valid answer extracted from a training file, using a certain value for the field. The corresponding field in the client requests will be classified as mutating. This leads only to two approximations: the emulator will always use the same value, and it will accept as correct any value used by the client without discarding the wrong ones. These approximations might be exploited by a malicious user to fingerprint a ScriptGen honeypot, but can still be considered as acceptable when dealing with attack tools.

In order to deal with the first case, we introduced a novel protocol-agnostic correlation algorithm, that allows to find content in the requests that always repeats in the answers. This process takes advantage of the statistical diversity of the samples to identify cookie fields in the requests and consequently produce correct answers.

2.3 Inter-protocol dependencies

The protocol knowledge represented through the state machines mentioned before is session-specific: a path along the state machine corresponds to the dialogue between a client and a server in a single TCP session. Therefore, this choice builds a definition of state whose scope is bounded inside a single TCP session. This might be oversimplistic.

For instance, many exploits consist in sending a set of messages towards a vulnerable port (A), taking advantage of a buffer overflow to open another port (B) usually closed and run, for instance, a shell. Before sending the messages to port A, these exploits often check if port B is open. If port B is already open, there is no reason to run the exploit since probably somebody else already did it in a previous time. Limiting the scope of the state to a single TCP session, the ScriptGen emulator would be forced to choose between two policies:

1. Port B is left always closed. In this case, the exploit will always fail and probably the attacker will consider the system as patched, and therefore not interesting.

2. Port B is left always open. In this case, the exploit will never send the malicious payloads on port A, and the honeypot will loose valuable information on the attack process.

Therefore, in order to maximise the amount of interesting information, port B should be opened only after having seen a successful exploit on port A. To do so, ScriptGen infers dependencies between different sessions observing the training samples. For instance, knowing that port B is normally closed, a SYN/ACK packet on that port will generate a dependency with the previous client request on port A. That is, every time that the emulator will reach the final state on port A that characterizes the exploit, will open port B mimicking the behavior of a real host. This is achieved through an inference algorithm that, based on a set of simple rules such as the one described before, finds these dependencies and represents them through signalling primitives between the various states.

3 ScriptGen potential

The potentials of the ScriptGen approach are manifold.

- Increasing the length of the conversations with clients, ScriptGen allows to better discriminate between heterogeneous activities. As mentioned before, often different activities share an initial portion of the protocol functional path and therefore it is impossible to discriminate between them without carrying on the conversation long enough. Thanks to the protocol independence assumption, ScriptGen allows to achieve an high level of interaction with the client for any protocol in a completely automated and protocol-agnostic way.
- ScriptGen is able to precisely identify deviations from the known protocol behavior. If the ScriptGen emulator receives a request that does not match any outgoing transition for the current state, it can classify it as a new activity, raising an alert: in fact, this means that it is something that was never seen before. Also, if ScriptGen was able to build a training set for this activity, it would be able to refine the state machine, thus its knowledge of the protocol behavior, learning how to handle this new activity. This is indeed possible through the concept of proxying: when facing a new activity, ScriptGen can relay on a real host, acting as a proxy and forwarding to the host the various client requests. These proxied conversations are then used to build a training set, and refine the state machine in a completely automated way. Also, thanks to the semantic inference, ScriptGen can also generate regular expressions to fingerprint the new activities, automatically generating signatures for existing intrusion detection systems.

From this short introduction, it is clear that ScriptGen is a completely innovative approach and a promising solution for the development of the honeypot technology. This technology covers a variety of different domains, ranging from bioinformatics, to clustering and data mining, to computer networks, to end up in intrusion detection. It is therefore a vast field, most of which still needs an in-depth exploration leaving room to a number of improvements.

Bibliography

[1] L. Spitzner, *Honeypots: Tracking Hackers*. Boston: Addison-Welsey, 2002.

[2] N. Provos, “A virtual honeypot framework,” in *Proceedings of the 12th USENIX Security Symposium*, pp. 1-14, August 2004.

[3] C. Leita, K. Mermoud, and M. Dacier, “Scriptgen: an automated script generation tool for honeyd,” in *Proceedings of the 21st Annual Computer Security Applications Conference*, December 2005.

Early Warning System based on a distributed honeypot network

V.H. Pham, F. Pouget, M. Dacier
Institut Eurecom
2229, route des Crêtes, BP 193
06904, Sophia-Antipolis, France
{pham,fabien,dacier}@eurecom.fr

Recently, the network security field has witnessed a growing interest for the various forms of honeypot technology. This technology is well suited to better understand, assess, analyze today's threats [1-4]. In this paper, we will present a high level description of an Early Warning System based on the Leurre.com project (a distributed honeypot network). The author continues the work of a former PhD student in the context of his own thesis. The rest of the text is organized as follows: the first section aims at presenting the Leurre.com project. Section 2 describes a clustering algorithm used to provide aggregated information about the attack tools used by hackers. Section 3 offers some insight on the current work we are performing on top of this seminal work in order to build a novel and lightweight Early Warning System.

1. Leurre.com project

The Leurre.com project is a worldwide honeypot network. We have deployed sensors (called platforms) at different places on the Internet. Each platform is made of three virtual machines. These machines wait for incoming connection and reply. Trafic generated is captured. At the beginning we had only one platform running in France. By now, the system has grown up to 35 platforms in 20 different countries. The data captured then are sent daily to a central server hosted at Eurecom Institut where we apply several techniques to enrich the data set. Leurre.com is a win-win partnership project. Each partner provides us an old PC and four routable IP addresses. On our side, we provide the software and the access to the whole data set captured during 3 years. Both sides sign an NDA (Non Disclose Agreement) that ensures that neither the name nor the IP addresses of ours partners or of the attackers are published.
The advantage of our approach is that it allows us to observe network traffic locally. This is not the case neither of Dshield [15] nor of Network telescope [14] where people take log files of big firewall or monitor traffic in a large range of IP addresses. The value of those systems resides in their capacity to observe worm propagation, DoS...but they cannot tell the local characteristics of the network traffic. We have showed in [17] the usefulness of such system in comparison with other approaches. For conciseness, we do not mention all similar projects here. Interested reader can find it in [10].
There are several partners who are working on our data set for their research. More information can be found on http://www.leurrecom.org.

2. Existing Work

So far, we have collected data for more than 3 years. We have applied several methods to analyze data set and we have had some early interesting results. In the following paragraphs, we will give a survey on it.

2.1 Clustering algorithm

Data captured by Leurre.com network is in raw format. The packets by themselves do not contain much information. This leads us to search for a more meaningful and easier to manipulate represention. Indeed, raw data are reorganized into different abstracts levels by grouping packets according to their origin and destination. Then we enrich the data set by adding geographical information thanks to the Maxmind database [7] and Netgeo [6], domain name and OS fingerprinting of each sources. To determine the OS of a source, we use passive OS fingerprinting tools: ettercap [5], Disco [9] and p0f [8]. At this stage, we have the answers for questions such as who attacks who? Which services are most attacked? Which domains hackers belong to.... However, we have no clear idea of which tools are mostly used. The answer for this question would be very interesting. For this reason, we developed the clustering algorithm, which aims at identifying attack tools used by hackers. The algorithm is presented in detail in [11]. Here we just give a very high level of description. First of all, we define the fingerprint of an attack by using a few parameters:

- The number of targeted virtual machines on the honeypot platform
- The sequence of ports
- The total number of packets sent by the attacking source
- The number of packets sent by an attacking source to each honeypot virtual machine
- The duration of the attack
- Ordering of the attack
- The packet content (if any) sent by attacking source.

Then we use certain techniques to group all sources that share the same fingerprint into a cluster (see [12] for more information about this).

2.2 Correlative analysis

In the previous paragraphs, we group all the sources that share the same fingerprint. By digging into the database, we found some relationships between these activity fingerprints. This led us to carry out some knowledge discovery approach that enables us to automatically identify important characteristics of set of attacks. To make a long story short, we summarize this lengthily process by the few following steps:

- *Manual identification of potentially interesting features.*
  E.g. manual inspection seems to reveal that some attacks originate from some specific countries while others do not.

- *Matrix of similarity of clusters*
  The underlying purpose of this step is to calculate the level of similarity between clusters w.r.t a specific feature. In our context, we have applied some distance functions (Euclidean, peak picking [12], SAX [12]...) to define the notion of similarity. The output of this step is a matrix M expressing similarities between clusters. The values of M(i,j) represents the similarity level between cluster i and cluster j.

- *Extraction of dominant set*
  We see this matrix as an undirected edge-weighted graph G. A vertex V corresponds to a cluster. The motivation for this step is to use existing algorithms to extract group of similar clusters. In our case, we have applied the dominant-set extraction [13] algorithm to discover set of ***cliques.*** Each clique C is a complete sub graph of G. It means that all vertices in C are connected to each other. Because of the space limitation, we do not present how the algorithm works. The interested reader can find this in [13].

3. Ongoing work

Although we have had some early interesting results but there are also open problems and work to be improved. We will cite the most important open questions here and discuss the possible solutions we are currently working on.

We begin by the small cluster problem. By applying the clustering algorithm, we have identified interesting clusters. But besides that, we have also a lot (around 50000) small clusters. This means that the fingerprint of these clusters is very specific. This can be due to several factors that we need to analyze:

Network disturbances

If packets are reordered or lost on their journey to the victim host, this cause the involved source to be misclassified or to be stored in a cluster on its own. As these phenomenons are quite frequent (above 10% sometimes) they can lead to the generation of a large amount of meaningless clusters.

Bad definition of *source* notion:

In our database, we define the notion of source. A source S is bound to an IP address, I, which is seen sending packets to at least one of our platforms. All packets sent by that IP, I, are linked to source S as long as no more than 25 hours elapse between received packets. Packets arriving after a delay larger than 25 hours, will be bound to another source S'. We have cases where attacking sources regularly send packets to our honeypots. The list of destination ports targeted by the hacker, in this case, forms a repeated pattern. It would be more reasonable if we say that repeated pattern is the signature of this kind of attacks. We are working on discovering repeated patterns of long port lists. With this, we hope to be able merge sources that have the same repeated pattern in port list but belong to different-small-clusters.

4 Long-term goals

The final purpose of the author PhD thesis is to construct an early warning system based on the above notion of clusters. The general model will look like **figure 1.** The system consists of a centre server and several platforms. The functionality of each components are described as following:

Platform consists of three subcomponents :

Cluster detector aims at detecting attacks from hackers. The output will be a cluster identification (if known cluster) or a new cluster. The output then is transferred to *Platform Profiler*.

Based on the attack profile of the platform and the volume of the attacks, *Platform Profiler* may decide to raise an alert if the observed type, or volume, of attacks is abnormal.

Alert Manager sends regularly new cluster information to correlation center and receives update profile information from correlation center.

Correlation Center gathers alerts from several platforms and take appreciate actions.

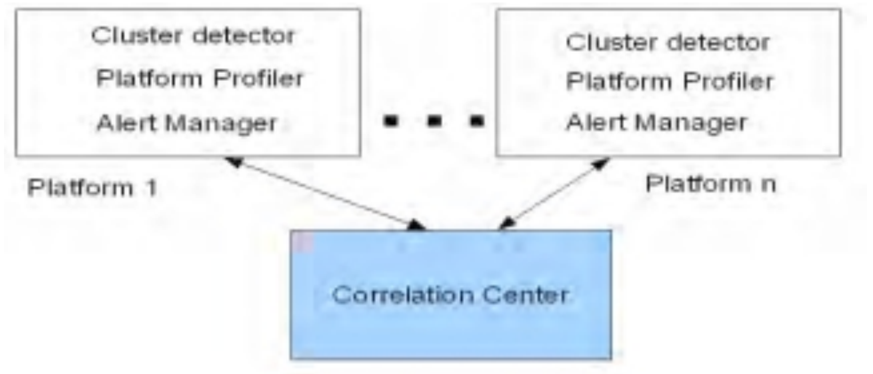


Figure 1

Reference

[1] David Dagon, Xinzhou Qin, Guofei Gu and Wenke Lee, HoneyStat: LocalWorm Detection Using Honeypots, Seventh International Symposium on Recent Advances in Intrusion Detection (RAID '04), 2004.
[2] Laurent Oudot, Fighting Spammers with Honeypots, http://www.securityfocus.com/infocus/1747, 2003.
[3] Lawrence Teo, Defeating Internet Attacks Using Risk Awareness and Active Honeypots, Proceedings of the Second IEEE International Information Assurance Workshop (IWIA'04), 2004.
[4] Nathalie Weiler, Honeypots for Distributed Denial of Service Attacks, Proceedings of IEEE WET ICE Workshop on Enterprise Security, 2002.
[5] Ettercap NG utility home page: http://ettercap.sourceforge.net.
[6] CAIDA Project. Netgeo utility -the internet geographical database.
URL:http://www.caida.org/tools/utilities/-netgeo/.
[7] MaxMind GeoIP Country Database Commercial Product. URL:http://www.maxmind.com/app/products.
[8] p0f Passive Fingerprinting Tool. URL:http://lcamtuf.coredump.cx/p0f-beta.tgz.
[9] Disco Passive Fingerprinting Tool. URL:http://www.altmode.com/disco.
[10] F. Pouget, M. Dacier, V.H. Pham. “Understanding Threats: a Prerequisite to Enhance Survivability of Computing Systems”. In Proceedings of the International Infrastructure Survivability Workshop (IISW 2004), Dec. 2004, Portugal.
[11] F. Pouget, M. Dacier. “Honeypot-based Forensics”. In Proceedings of the AusCERT Asia Pacific Information Technology Security Conference 2004 (AusCERT2004), May 2004, Australia.
[12] F.Pouget. « Système distribué de capteur Pots de Miel : Discrimination et Analyse Corrélative des Processus d'Attaques ». PhD thesis, Jan 2005, France.
[13] M.Pavan and M. Pelillo, « A new graph-theory approach to clustering and segmenation », in Proceding of the IEEE Conference on Computer Vision and Pattern Recognition, 2003.
[14] CAIDA, the Cooperative Association for Internet Data Analysis web site: http://www.caida.org
[15] DShield Distributed Intrusion Detection System. URL:http://www.dshield.org.
[16] F. Pouget, T. Holz, « A Pointillist Approach for Comparing Honeypots », in Proc. Of the Conference on Detection of Intrusions and Malware & Vulnerability Assessment. (DIMVA 2005), Vienna, Austria, July 2005.
[17] F. Pouget, M. Dacier, V-H Pham. “Leurre.com: On the advantages of deploying a large scale distributed honeypot platform”. In Proceedings of E-Crime and Computer Conference (ECCE'05), Monaco, March 2005.

Session on System Modelling

Chair: Benedicto Rodriguez, University of Southampton, UK

A multi-perspective approach for the design of error-tolerant socio-technical safety-critical interactive systems

Sandra Basnyat
University Paul Sabatier, LIIHS-IRIT
118, route de Narbonne 31062 Toulouse, Cedex 4, France
basnyat@irit.fr

Introduction

This article summarises the concept of the PhD research on a multi perspective method for the design of safety critical interactive systems. The goal is to propose model based design techniques that allow to take into account and manage erroneous human and system behaviour. We present an integrated modelling framework to attain this goal

One of the main characteristics of interactive systems is the fact that the user is deeply involved in the operation of such systems. More specifically, a safety critical interactive system is one in which any failure or design error has the potential to lead to loss of life, that is to say the cost of failure outweighs the cost of development. The design of a usable, reliable and error-tolerant interactive safety-critical system is a goal that is hard to achieve because of the unpredictability of the humans involved, but can be more closely attainable by taking into account information from previous known situations. One such usually available and particularly pertinent source is the outcome of an incident or accident investigation.

While designing interactive systems, the use of a formal specification technique is of great help because it provides non-ambiguous, complete and concise notations. The advantages of using such a formalism is widened if it is provided by formal analysis techniques that allow to prove properties about the design, thus giving an early verification to the designer before the application is actually implemented.

Model-based development (MBD) is a developing trend in the domain of software engineering [7] advocating the specification and design of software systems from declarative models [11]. It relies on the use of explicit models and provides the ability to represent and simulate diverse abstract views that together make up a 'system', without the need to fulfill its implementation. Many types of models are used based on the given requirements to contribute to the overall design. For example, task model, user model, environmental model, platform model, system model, presentation & layout model, design model. However, formal specification of interactive systems often does not address the issues of erroneous behaviour that may have serious consequences for the system.

Our research focuses mainly on task and system modelling, which like other models are generally developed for normal behaviour without taking into account human or system-related "errors". What is more, task modelling and system modelling are often performed by experts with different competencies. It is unlikely that a human factors specialist will develop the task model and continue to develop the system model. A computer scientist will also in general be incapable of collecting the necessary information for task modelling.

We have developed an approach that supports the integration of information relating to human and system-related erroneous behaviour in models (principally the task and system models). We believe this perspective extends the general boundaries of model based development (MBD), by taking into account additional information relating to previous experiences of failure. The ultimate goal is improvement of the design process with the aim of producing safer safety-critical interactive systems.

1. Motivation and Objectives

Human error plays a major role in the occurrence of accidents in safety-critical systems such as in aviation, railways systems, or nuclear power plants [12]. It has been claimed that up to 80% of all aviation accidents are attributed to human 'error' [6]Yet in practice, most accidents stem from complex combinations of human 'errors', system 'failures' and managerial issues (of employees, of the company…). Interactive systems, and in particular safety-critical interactive systems need to be designed while taking into account the eventuality of human "error". These aspects should be taken into account in early phases but also throughout the deign process.

2. The Approach

Our approach aims to bring coherence between the task model and system model while taking into account human and system-related error in order to minimise the occurrence of erroneous events (also known as deviations) in safety-critical interactive systems. The approach detailed in the "integrated modelling framework" presented in **Error! Reference source not found.**.

The approach results from our research on two case studies. The first concerns a fatal mining accident involving the waste fuel delivery system of a cement plant. The second is an interactive application (MPIA) embedded in an interactive cockpit and compatible with the ARINC 661 standard [1].

Our fundamental idea is to gather, within a single framework, principle issues of User Centred Design (UCD) and of safety critical interactive systems.

In figure 1, there are four modeling phases, requirements modeling, system modeling, task modeling and safety modeling. The PhD research is based on these four models. Since the case studies we have been working on are existing systems, we will not discuss the requirements modelling here.

Part 1 : system modelling. For the system modelling we have used the 'Petshop' environment [8] and the Interactive Cooperative Objects (ICO) formalism [PUT NAVARRE 2003]. The phase is divided into two parts, the modeling of functional behaviour and of safety-related behviour. The modeling of functional behaviour has been demonstrated using the first case study [3]. After having modeled the system, we have performed a technical barrier analysis on the system to eventually reduce the possibility of erroneous events. The barriers are also modeled using the ICO notation and Petshop environment. This coupling of barriers and system and what it brings to increase the reliability of an interactive system has been presented in [13].

Part 2 : task modelling. This phase is divided in two, the first describes the predicted behaviour of the user and results in an analysis of user tasks and activities. The second concerns the modeling of user deviations with respect to predicted behaviour. For this phase, we use the ConcurrentTaskTree (CTT) [10]. The CTT tool,

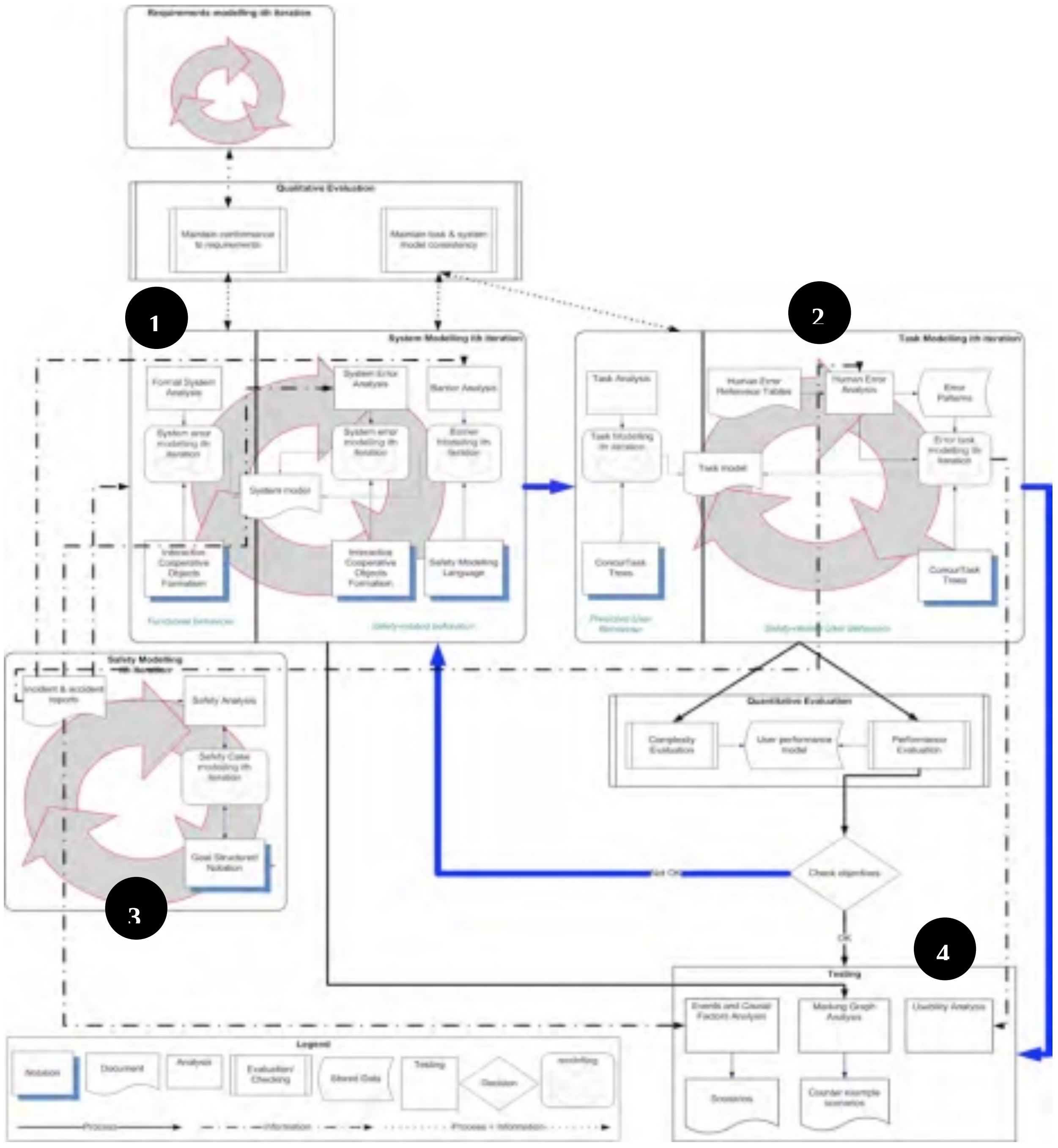


Figure 1: Integrated Modelling Framework

CTTE allows the task models to be simulated in order to study different possible paths of interaction. The notation is based on four types of task (abstract tasks, user tasks, system tasks and interaction tasks) as well as several temporal operators. To take into account erroneous user behaviour, our task modeling is extended by reusable subtasks (called error patterns) resulting from human error reference tables. The method for integrating "standard" task models and task models representing potential deviations is described in [9] [3].

Part 3 : Safety modelling. In this phase, we propose a safety analysis by exploiting the “safety case” notation using the “goal structuring notation” GSN. Safety Cases are documentation often required by regulatory agencies before they will grant approval for the operation of complex command and control systems. These cases describe the arguments that a company will use to convince the regulator that their proposed application is ‘acceptably safe’[5].

Phase 3 also exploits further information including incident and accident reports. It is clear that this information is central to our proposed approach since a significant aspect of safety is to make sure successive versions of a system do not allow the same incidents or accidents to be reproduced. The use of safety cases in the proposed approach is described in [3].

Part 4: Testing. This part concerns the verification of prototypes and models. We have used incident and accident reports to perform events and causal factors analyses. This technique is often used in accident investigation to identifier the path of events and contributing factors that led to an accident. We have also ICO used system models to perform marking graph analyses. The graphs are used to systematically explore all of the possible scenarios leading to an accident which described in terms of states in the system model. The techniques serve two purposes. Firstly, to ensure that “modified” system model does not allow the same sequence of events that led to the accident and secondly, our approach can be used to reveal further scenarios that could eventually lead to non-desirable system states. We have illustrated this concept on the first case study in [2].

3. PhD Progress

The research presented corresponds to the 3rd year of the PhD which is envisaged to end by December 2006. The PhD is co-directed by Philippe Palanque (University Paul Sabatier, Toulouse, France) and Chris Johnson (University of Glasgow, Scotland, UK)

4. Acknowledgements

The research is supported by the EU funded ADVISES Research Training Network, GR/N 006R02527.http://www.dcs.gla.ac.uk/advises/

5. References

1. "ARINC 661 Cockpit Display System Interfaces to User Systems. ARINC Specification 661." Airlines Electronic Engineering Committee 2002.
2. Basnyat, S, Chozos, N, Johnson, C and Palanque, P. "Redesigning an Interactive Safety-Critical System to Prevent an Accident From Reoccurring." 24th European Annual Conference on Human Decision Making and Manual Control. (EAM) Organised by the Institute of Communication and Computer Systems, Athens, Greece. (2005)
3. Basnyat, S, Chozos, N and Palanque, P. "Multidisciplinary Perspective on Accident Investigation. " Special Edition of Elsevier's Reliability Engineering and System Safety Journal (2005)
4. Bastide, R, Sy, O, Palanque, P and Navarre, D. " Formal Specification of CORBA Services: Experience and Lessons Learned." ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'2000), Minneapolis, Minnesota USA. ACM Press (2000)
5. Bloomfield, R., P Bishop, C Jones and P Froome. Available Online at: Http://Www.Adelard.Co.Uk/Resources/Papers/Pdf/Dsn2004v10.PDF (Accessed 12 December 2004). 2004.

6. Johnson, C. W . Failure in Safety-Critical Systems. A Handbook of Accident and Incident Reporting. University of Glasgow Press, Glasgow, Scotland (2003).
7. MDA Guide version 1.0.1. "OMG Document number omg/2003-06-01." Web page, 2003. Available at http://www.omg.org/cgi-bin/doc?omg/03-06-01.
8. Navarre, D, Palanque, P and Bastide, R. "A Tool-Supported Design Framework for Safety Critical Interactive Systems." Interacting With Computers 15/3 (2003) 309-28.
9. Palanque, P, and Basnyat, S. "Task Patterns for Taking into Account in an Efficient and Systematic Way Both Standard and Erroneous User Behaviours." HESSD 2004 6th International Working Conference on Human Error, Safety and System Development, Toulouse, France (within the IFIP World Computing Congress WCC 04) (2004)
10. Paterno, F. Model Based Design and Evaluation of Interactive Applications. Berlin, Springer Verlag (1999)
11. Puerta, A. R . "Supporting User-Centered Design of Adaptive User Interfaces Via Interface Models." First Annual Workshop On Real-Time Intelligent User Interfaces For Decision Support And Information Visualization, San Francisco. (1998)
12. Reason, J. Managing the Risks of Organizational Accidents., Aldershot, UK: Ashgate (1997).
13. Schupp, B, Basnyat, S, Palanque, P and Wright, P. "A Barrier-Approach to Inform Model-Based Design of Safety-Critical Interactive Systems." 9th International Symposium of the ISSA Research Section Design Process and Human Factors Integration: Optimising Company Performances, Nice, France. (2006)

ReSIST Knowledge Architecture: Semantically Enabling Large-Scale Collaborative Projects

Afraz Jaffri, Benedicto Rodriguez

Dependable Systems and Software Engineering Group
School of Electronics and Computer Science
University of Southampton, Southampton S017 1BJ, UK

Introduction

The ReSIST project is aimed at providing a framework towards resilience for survivability in IST systems. One of the aims of the project is to collate as much knowledge as possible about resilient systems and technology. This includes organisations that are researching resilient systems; researchers interested in resilient systems; papers associated with resilient systems; research issues; faults, errors and failures that have occurred on IST systems; and resilient systems research topics. All the knowledge will be stored in a knowledge base as RDF. In order to describe the concepts within the domain of resilient systems, there needs to be an ontology that can accurately describe the relationships between these concepts and act as a controlled vocabulary for the project.

This extended abstract briefly describes the ontology design process and the problems that are faced when trying to model a domain. Our solutions to these problems, with reference to the ontology of dependable and secure computing within ReSIST, are outlined and directions for future monitoring of the ontology are given.

1. Issues in Ontology Design

Even though the main components in ontology design are well known and are formally defined, there are still significant issues preventing ontology design from becoming a systematic activity. Mainly because there is no one correct way to model a domain and there are always viable alternatives [2].

There has been however different efforts to formalize the process of ontology design in order to minimize its subjectivity and convert it into a well-defined engineering activity [3].

Regardless of the methodology employed, the designer will have to identify how the concepts of the domain being modelled fit into the main components of the ontology, which are:

- Classes: The domain concepts, usually represented in a hierarchy.
- Relations (also known as properties or slots): The defined links between concepts in the domain.
- Axioms: Act as restrictions and control the use of certain relations.
- Instances: Actual member of classes.

Based on how the designer populates these ontology components with the domain vocabulary will determine the success of the ontology in the knowledge base.

2. Issues in the Design of a Dependable and Secure Computing Ontology

In order to fit the ReSIST dependable and secure computing ontology for purpose, the first thing to consider in the design process is the requirements or use cases for which the ontology will be used [2]. This involves, creating a vocabulary of the terms or concepts within the domain that are trying to be modelled [3]. In this case there is such a vocabulary already available and it is based on a paper by the IEEE describing a taxonomy of dependable and secure computing [1]. The taxonomy is extremely complex and is not a straight forward hierarchy that can easily be turned into an ontology.

The classification of faults is particularly difficult to turn into an ontology, as the classification provides a lot more information than can be modelled by a straight class hierarchy. There are eight elementary fault classes and three other categories that overlap with the elementary classes: physical faults, interaction faults and development faults. With so many dimensions it is unclear which concepts to model as classes and which to model as properties. In a situation such as this, the best way to model the domain is to create an instance of one of classes that will be modelled.

The Risk's Digest is a forum for reporting faults and errors on all types of system. One of these faults from the latest issue is summarised below:

Excel garbles microarray experiment data

When we were beta-testing [two new bioinformatics programs] on microarray data, a frustrating problem occurred repeatedly: Some gene names kept bouncing back as "unknown." A little detective work revealed the reason:
... A default date conversion feature in Excel ... was altering gene names that it considered to look like dates. For example, the tumor suppressor DEC1 [Deleted in Esophageal Cancer 1] was being converted to '1-DEC.'"

This fault could be described and modelled in the otology in a number of ways. If one was to create a class for every variety of fault type, there would be 31 classes. With this approach the fault could be described, in RDF, as follows:

```
<resist:Develpoment-Internal-Human-Made-Software-Non-Malicious-Non-Deliberate-Accidental-Persistent-
Fault rdf:about="http://fault-uri">
    <resist:fault-type rdf:resource=&resist;Development-Fault/>
    <akt:has-pretty-name> Excel garbles microarray experiment data
    </akt:has-pretty-name>
</resist:Develpoment-Internal-Human-Made-Software-Non-Malicious-Non-Deliberate-Accidental-Persistent-
Fault>
```

However, with this approach the names of the faults become very long and cumbersome and some of the information is lost. A second way of modelling the hierarchy would be to create a generic fault class and then create eight properties corresponding to the eight elementary fault classes. The domain of these properties will be the 16 fault types given in the paper. An example of this for the same instance data is shown below:

```
<resist:Fault rdf:about="http://fault-uri">
        <resist:phase-of-creation rdf:resource="&resist;Development-Fault/>
        <resist:system-boundary rdf:resource="&resist;Internal-Fault"/>
        <resist:phenomenological-cause rdf:resource="&resist;Human-Made-Fault"/>
        <resist:dimension rdf:resource="&resist;Software-Fault"/>
        <resist:objective rdf:resource="&resist;Non-Malicious"/>
        <resist:intent rdf:resource="&resist;Non-Deliberate"/>
        <resist:capability rdf:resource="&resist:Accidental-Fault"/>
        <resist:persistence rdf:resource="&resist:Persistent-Fault"/>
        <resist:fault-type rdf:resource=&resist;Development-Fault/>
        <akt:has-pretty-name> Excel garbles microarray experiment data </akt:has-pretty-name>
</resist:Fault>
```

This approach leads to clearer formatting and does not lose any of the information from the hierarchy. The argument about the presentation of the RDF could be said to be irrelevant since it is intended for machine processing. However, if queries were to be made on the data, it would be much easier to get results if the instances were created according to the second method. For example, if somebody wanted to get the names of all Software faults stored in the knowledge base, they could make the following SPARQL query:

```
PREFIX  resist:       <http://www.resist.eu/ontology/resist#>
PREFIX  akt:          <http://www.aktors.org/ontology/portal#>
SELECT  ?y
WHERE
(?x resist:dimension resist:Software-Fault .
?x     akt:has-pretty-name    ?y . )
```

The same query could not be made with the data made with the first method because we do not know or care whether the fault is development, malicious, human-made, etc. In other words the class hierarchy creates a high level of coupling and dependency between the faults which is not the case with the second method.

3. Other Ontologies in the ReSIST Knowledge Base

The initial building block of the ReSIST Knowledge Base (RKB) is the set of ontologies needed in the application.

Obviously the key ontology is the one in dependable and secure computing, but because of the requirements of the ReSIST project, there are additional sub-ontologies or utility ontologies that will be needed in the knowledge base. These ontologies are described below:

- Dependable and Secure Computing Ontology: Main RKB ontology which model the key concepts of the RKB.
- Computer System Applications Ontology: Includes a model for the computer systems, fields, or applications, to which the concepts of dependability and security are being applied.
- Academia Domain Ontology: Provides the model for the academic domain, institutions, people, publications, research interests, etc.
- Syllabus and Courseware Ontology: Provides the model for syllabus and courseware in the dependability and security field. The definition of such educational material is another goal of the RKB.

With the exception of the dependability and security ontology, which is being designed from scratch, the rest of the ontologies will be reused from publicly available ontologies if they are available. Some possible ontologies have already been identified for this purpose.

In the case of the ontology for the academic domain, the AKT Reference Ontology [4] is already being used for ReSIST.

For a syllabus and courseware ontology some candidates are being evaluated. It does not seem to be an ontology readily available although the IEEE Learning Technology Standards Committee (LSTC) Learning Object Metadata (LOM) [5] specification could be an excellent starting point to meet the RKB's needs.
Lastly, the ontology about computer systems applications presents the problem of a domain being too generic and not well specified. The list of systems where the concepts of dependability and security can be applied to is potentially endless. Taxonomies such as DMOZ Open Directory Project [6] could address this issue.

Figure 1 illustrates the main components in the RKB architecture and where the ontologies described above fall into that architecture:

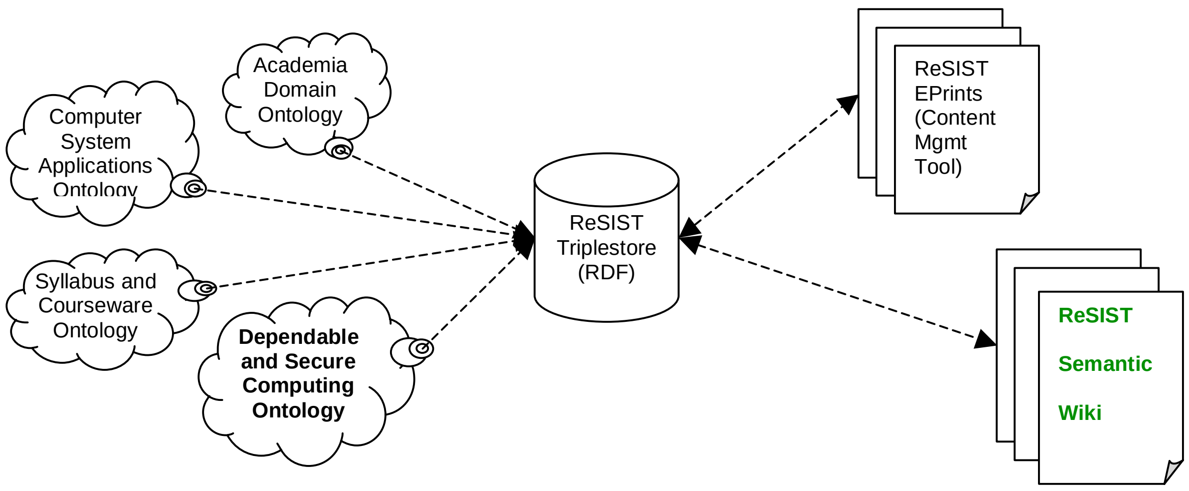

Academia Domain Ontology
Computer System Applications Ontology
Syllabus and Courseware Ontology
Dependable and Secure Computing Ontology
ReSIST Triplestore (RDF)
ReSIST EPrints (Content Mgmt Tool)
ReSIST Semantic Wiki

Figure 1 - Ontologies in the ReSIST Knowledge Base.

The ontologies provide the "semantic knowledge" as an Ontology Web Language (OWL) [7] deliverable that will be stored in the Triplestore component as Resource Description Framework (RDF) [8]. All the data in the Triplestore will provide a closed vocabulary for the RKB with semantic relations. The interface for users of the RKB is provided via the ReSIST Semantic Wiki application. This is a customisation for the ReSIST project of the open-source Semantic MediaWiki extension [9] that is publicly available.

The ReSIST Semantic Wiki will be able to use the RDF data from the Triplestore and create content dynamically using the semantic relations in the RDF data, as well as modify or add new semantic knowledge into the Triplestore based on the actions performed by the users of the RKB as they create, edit, or delete ReSIST Wiki pages.

The last component in the architecture diagram is the ReSIST EPrints [10] application. A content management tool provided to manage the repository of publications relevant to the ReSIST project. It will also be synchronised with the Triplestore in the sense that the metadata of relevant ReSIST publications should be consistent with the closed vocabulary in the Triplestore.

Another of the challenges in the implementation of the RKB is to maintain the integrity of the metadata stored in the Triplestore as RDF, with the ReSIST Semantic Wiki and the ReSIST Eprints repository.

4. Conclusion and Future Work

Creating the ReSIST ontology has shown that in order to model complex domains that are not well understood, the applications for which the ontology will be used should be looked at. In particular, creating instances and trying queries on the knowledge that is likely to be used will aid and improve the ontology, reducing the number of unused classes that could have been made without looking at use cases.

It remains to be seen how the ontology will be used and how the ontology will be extended. The ontology evolution will be documented and monitored and it is expected that these results will be presented at the Seminar.

Additional challenges faced in the implementation of the RKB include the creation and / or reuse of additional sub-ontologies, and the algorithms to preserve referential integrity of semantic data within the Triplestore and its applications, such as the ReSIST Semantic Wiki or the ReSIST EPrints repository.

5. References

[1] Jean-Claude-Laprie, Brian Randell, Carl Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing", in IEEE Transactions on Dependable & Secure Computing. Vol. 1, No. 1, pp. 11-33.

[2] N. Noy and D. L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05 and SMI-2001-0880, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics, March 2001.

[3] Fernandez, M., Gomez-Perez, A., & Juristo., N., METHONTOLOGY: From Ontological Art to Ontological Engineering. *In Workshop on Knowledge Engineering: Spring Symposium Series (AAAI'97)*, pages 33-40, Mellow Park, Ca, 1997. AAAI Press.

[4] The AKT Reference Ontology. http://www.aktors.org/publications/ontology/

[5] Learning Technology Standards Comittee of the IEEE: Draft Standard for Learning Objects Metadata IEEE P1484.12.1/D6.412. June 2002). http://ltsc.ieee.org/doc/wg12/LOM 1484 12 1 v1 Final Draft.pdf/

[6] DMOZ. Open Directory Project. http://dmoz.org/

[7] Mcguiness, D. & Harmelan, F. V. Eds., 2003. *OWL Web Ontology Language Overview*, W3C Recommendation [Online] Available from: http://www.w3.org/TR/owl-features/ [Accessed 4 April 2006].

[8] Lassila, O & Swick, R., 2002. RDF Model and Syntax Specification, (W3C Recommendation), [Online], Available from: http://www.w3.org/TR/REC-rdf-syntax/ [Accessed 5 July 2005].

[9] Semantic MediaWiki. http://sourceforge.net/projects/semediawiki/

[10] Eprints. Supporting Open Access (OA). http://www.eprints.org/

Introducing Hazard and Operability Analysis (HazOp) in business critical software development

Torgrim Lauritsen - Newcastle University

Introduction

I am evaluating the effect of introducing safety analysis techniques in business critical software development environment. The goal is to find techniques that can help software developers to develop software that is more business safe than today. I have designed and executed an experiment with the Hazard and Operability Analysis (HazOp) technique that have been used with success for safety-critical systems like avionics, train, nuclear plants and for the chemical process industry for many years. Since safety critical systems deals with the same issues that business critical software deals with, I believe that it can have the same effect in business critical software development.

1. Hazard and Operability Analysis (HazOp)

HazOp is a formal, methodical and critical investigating technique that can be used to analyse UML diagrams, such as class and sequence diagrams.

The goal of a HazOp analysis is to identify possible problems that can arise during the operation and maintenance of the software. The results from the analysis can then be used as a basis for further development and tests. Guidewords are used during the analysis to identify possible problem areas - see bullet points below for some guidewords and their interpretations for message passing:

- No / None - Message not sent when it should be
- Other than - Message sent at wrong time
- In addition to - Message sent at correct time and also an incorrect time

- More than - Message sent later/more often than intended
- Less than - Message sent earlier/less often than intended
- Earlier - Message sent earlier within message sequence than intended
- Later - Message sent later within message sequence than intended

The guide words should be used on the study nodes – points in the system where we want to focus. Usually study points are points where the system interacts with its environment – e.g. uses input or two or more parts of the system exchange information – e.g. a network connection

Other issues to analyze are event, action, action for a software control system, states, relationships, classes, attributes, message destination, message conditions, etc.

The results from the HazOp analysis are documented in a HazOp table:

Guide word	Study node	Causes	Consequences	Possible solution
Other than	Buyer sends an item request	Mismatch of what is perceived and what is ordered	Wrong items will be sent to the customer	Insert check to ensure correct information
Other than	Buyer signs for order	Could be a imitation of the signature -> fraud	Company will lose money	Insert barrier that check signature
No	Buyer pays invoice	Company receives no money for the items sold.	Company do not make profit from the sale.	Implement a credit check and log who placed the order
….	…..	……	………………	……

Table 1.0

2. HazOp Experiment in four Newcastle companies

I am now finished with the experiments and the data collection from four software development companies in Newcastle. I am going to give you the results from the analysis at this student seminar.

In the analysis of the experiment I will answers research questions like:

- “Will the introduction of the HazOp technique lead to software that is more business safe?”
- “Does the HazOp technique find more hazards than ad hoc “brainstorming” techniques that are in use in today’s software development?”
- “Will the software developers see the benefit of using safety analysis techniques in their daily work?”
- “Can the results (e.g. safety requirements) be included in test suites?”
- “Can the use of safety analysis techniques compensate for a reduction in time spent on testing?”

> Enterprise Compliance at Stake < Dependability Research to the Rescue!

Samuel Müller

IBM Zurich Research Lab
sml@zurich.ibm.com

1 Introduction

Large enterprises are confronted with an increasing amount of new and constantly changing regulatory requirements. In the light of this development, successfully being and effectively remaining compliant with all relevant provisions poses a big challenge to affected companies. In this context, research findings in dependability, that is, in the trust that can be placed upon a system to deliver the services that it promises, may come to the rescue. In this paper, we shall delineate how dependability research can contribute to successful compliance management. We shall also demonstrate how the classical dependability and security taxonomy can be generalized and extended to suit the needs of this evolving field.

1.1 Dependability and Compliance?

While the technical benefit and the respective merits of dependability are undoubted in research, investments in dependability are often hard to justify in practice. Unless a system is absolutely critical to the business, the importance of dependable and secure systems is frequently overlooked or simply ignored.

The advent and importance of enterprise compliance obligations has the potential to change this conception. As mentioned above and described in [1], organizations are confronted with an accretive amount of increasingly complex and constantly evolving regulatory requirements. For instance, as a result of the famous Sarbanes-Oxely Act, CEOs and CFOs now face personal liability for the occurrence of material weaknesses in their internal control systems for financial reporting. Furthermore, companies risk paying considerable implicit (e.g., decrease in market valuation or customer base) and explicit (e.g., monetary fine) penalties, if they fail to attain and prove compliance with various regulations, provisions, and standards. Hence, particularly large enterprises are well-advised to carefully check their regulatory exposure and to ensure overall compliance.

Given the complexity of today's IT-supported business operations, attaining overall enterprise compliance is by no means an easy task. Hence, in order to continually ensure compliance with relevant regulations, companies need a well-defined and comprehensive approach to compliance management. In an attempt to address this need, we have proposed REALM (Regulations Expressed As Logical Models), a well-structured compliance management process, which includes the formal representation of relevant regulatory requirements using a real-time temporal object logic (cf. [1]).

However, there are also a number of strands of existing research that may be in a position to contribute both to an all-embracing approach as well as to provide individual solutions. One promising research area with a seemingly large potential to address many issues central to achieving compliance is dependability research. Through the logical formalized of required properties, potentiating automated transformations and enforcement, the mentioned REALM-based approach to compliance already shares some similarities with certain areas within dependability (e.g., fault prevention/removal and, in particular, static verification). In close analogy, there seem to be many opportunities to employ well-established solutions from a dependability context to addressing enterprise compliance.

1.2 Some Well-known Concepts from Dependability

As defined in [2] and [3], "dependability refers to the trust that can be placed upon a system to deliver the services that it promises". As a result, dependability does not only concern security, but it also includes a number of other system properties, such as reliability, safety, and quality of service.

According to [3] and more recently [4], everything that might go wrong in a computer system can be characterized as being one of the following three types: *fault*, *error*, or *failure*. We speak of a failure when the delivered service no longer complies with its specification. We think of an error as that part of the system state that is liable to lead to a subsequent error. And we use fault to refer to the hypothesized cause of the error.

With an eye on possible applications within compliance management, in the above definition of failure, we could substitute 'service' with 'behavior' or 'system functionality', and 'specification' with 'regulatory requirement'. The definitions of 'error' and 'fault' could stay the same. Hence, we basically introduce compliance as a new and broader notion of dependability, and we specifically introduce a new type of failure, namely, a compliance failure.

Dependability research has also come up with a taxonomy that characterizes the possible approaches to the prevention of failures through preventing and neutralizing errors and faults. In particular, [3] defines four categories: *fault prevention*, *fault tolerance*, *fault forecasting*, and *fault removal.*

Fault tolerance can be further subdivided into *error processing* and *fault treatment.* The former can be done using error recovery (i.e., the substitution of the erroneous state by an error-free one) or error compensation (i.e., delivering of an error-free service using available redundancy). The latter is done by first determining the cause of the errors at hand (fault diagnosis), and by then preventing faults from occurring again (fault passivation).

Fault forecasting, possible using both probabilistic and non-probabilistic techniques, denotes evaluating system behavior ex ante with respect to future fault occurrence or activation.

Finally, fault removal involves three steps: first the we need to check whether faults are present (verification), then the causes of identified faults need to be determined (diagnosis), and finally, they need to be adequately addressed and possibly removed (correction).

2 Dependability to the Rescue

In our paper, we stress the important role of dependability and security in the context of compliance management. Using the classical dependability and security taxonomy introduced above, we can better understand the problems that arise with respect to enterprise compliance, and we can identify existing and new solutions addressing these problems. Furthermore, we will demonstrate how enterprise compliance can be characterized using the various taxonomical concepts and classes and, wherever necessary, we will suggest extensions to the taxonomy.

In particular, our contribution to existing research shall be as follows:

1. First of all, we will delineate in detail how dependability research can come to the rescue of a large set of compliance management problems. Specifically, we will demonstrate how dependability can contribute to attaining enterprise compliance with a broad set of regulatory requirements. Towards this end, we will show how the compliance problem can be better understood by mapping the classical dependability and security taxonomy to compliance-related aspects. For instance, we will explain how fault tolerance is to be understood in the larger context of enterprise compliance and we shall provide examples to illustrate our ideas. Along these lines, we will point out how established techniques and results from dependability can be directly applied to tackling individual compliance problems.
2. Secondly, we will also explain how dependability may be impacted by research on risk & compliance. Specifically,
   (a) we shall argue that compliance provides a useful legitimation and possibly even a business case for many subfields of dependability research.
   (b) we will argue that with respect to [4], compliance may lead to a generalized form of the classical dependability and security taxonomy. Along these lines, we will identify and suggest a number of additional attributes that, in our opinion, need to be added to the existing taxonomy.
   (c) we shall also discuss further additions to the taxonomy such as supplemental fault types and error classes.
   (d) as pointed out by [2], in the context of fault prevention/removal and security, *convincing* has got an important function and deserves to be included as a forth subtask. In close analogy, we identify and stress the crucial role of *auditing* in the context of compliance, and we will give a justification for this.
3. Finally, we will demonstrate how the classical dependability and security taxonomy can be combined with a set of identified regulatory requirements categories. This will give rise to an enhanced and comprehensive classification framework to better analyze the problem at hand and to identify feasible solutions.

References

1. Giblin, C., Liu, A.Y., Müller, S., Pfitzmann, B., Zhou, X.: Regulations Expressed As Logical Models (REALM). In Moens, M.F., Spyns, P., eds.: Proceedings of the 18th Annual Conference on Legal Knowledge and Information Systems (JURIX 2005). Volume 134 of Frontiers in Artificial Intelligence and Applications., IOS Press (2005) 37–48
2. Meadows, C., McLean, J.: Security and dependability: Then and now. In: Computer Security, Dependability, and Assurance: From Needs to Solutions, IEEE Computer Society (1999) 166–170
3. Laprie, J.C.: Dependability: Basic concepts and terminology. Dependable Computing and Fault Tolerant Systems **5** (1992)
4. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.E.: Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing **1** (2004) 11–33

Fault modelling for residential gateways

Sakkaravarthi RAMANATHAN
France Telecom RD/MAPS/AMS/VVT
2, avenue Pierre Marzin, 22307 Lannion Cedex, France
sakkaravarthi.ramanathan@francetelecom.com

Introduction

Traditional communication access methods, which rely on twisted copper pairs for *Public Switched Telephone Network* or coaxial cable for television service, are steadily evolving to a diverse collection of technologies that are capable of providing two-way broadband access to the residence [1]. This new capability presents both an opportunity and a challenge to the access providers. The opportunity arises in the ability to deliver many new services to the customer, while the challenge resides in the necessity of delivering these services economically and with a high level of QoS. High-speed access is evolving from *Integrated Services Digital Network* data rates, to *xDigital subscriber line* data rates that range from 1.5 Mbps to 50 Mbps, to fiber-to-the-home rates that can be in the hundreds of Mbps. Support for broadband access to the home must be able to accommodate new features and services as they develop without rendering the home infrastructure obsolete.

Therefore, a modular solution is required, where access interfaces can be updated without disabling (or even powering down) feature interfaces, and new feature interfaces can be added without requiring a new access infrastructure.

1. Residential gateways

A residential gateway is a centralized intelligent interface between the operator's access network and the home network. It is also the physical interface terminating all external access networks to the home as well as the termination point of internal home networks and enabling platform for residential services to be delivered to the consumer, including both existing services and new ones yet to come.

Recent advances in access technologies are creating several new network elements that enable the delivery of multiple services over high bandwidth access streams. However, these advances are often slowed by the substantial amount of legacy networks and systems, as well as by the lack of a unifying end-to-end architectural vision. The importance of residential gateways is to support emerging access technologies, legacy home networks, and new home network technologies in a seamless, modular way.

Advantages

An access network incorporating residential gateways provides a number of benefits to *consumers*. A single access line can offer many services, so the addition of a new service does not require a maintenance call requiring new wiring to the house. A consumer can have greater choice of service providers for individual services, and can change among them dynamically. Interactions between different in-home networks are possible because they all terminate at the residential gateway. Services that require “always on” functionality (e.g., home security or utility meter access) are readily implemented, since the access network is always active and the gateway cannot be inadvertently turned off. Residential gateways preserve consumer investment in existing in-home networks.

As the telecommunications market evolves from narrowband to broadband delivery, *service providers* also benefit from residential gateways in customer’s premises. The access provider’s risk is reduced, since it needs not predetermine all the possible services to be provided on a new access network. Furthermore, broadband access provides the means of delivering new services that have not yet been developed. A change in the access infrastructure has little impact on customers, since the residential infrastructure does not change when a new access method is installed. The marginal cost of adding a new service is low, since there is no need to add new access infrastructure. Since residential gateways can have substantial processing power, service providers can decentralize their infrastructure.

Considering the needs and advantages, Fig. 1 envisages the perfect residential gateway.

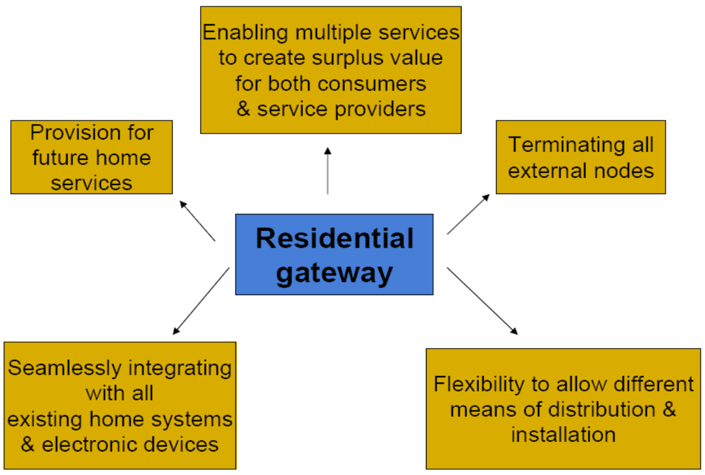

Enabling multiple services to create surplus value for both consumers & service providers
Provision for future home services
Terminating all external nodes
Residential gateway
Seamlessly integrating with all existing home systems & electronic devices
Flexibility to allow different means of distribution & installation

Figure 1: A full fledged residential gateway

2. France Telecom's residential gateway (LiveBox)

Introduced in July 2004, this home gateway plugs into the regular phone jack for broadband connection of all residential communication services. People can surf the Internet**,** watch TV via ADSL**,** make voice-over-IP phone calls or video calls, or play network games**.** Multimedia devices, phones, PCs, TVs, cameras and video-game consoles benefit from security and ease of use. The LiveBox is equipped with an ADSL modem plus Ethernet, WiFi and Bluetooth interfaces to accommodate all types of devices. It becomes a single point of entry for an entire world of high-tech entertainment and communications services. LiveBoxes met with tremendous success right from its launch, with 234,000 units leased in France in 2004, plus sales. The residential gateway was also rolled out in the United Kingdom, the Netherlands and Spain, and, at the end of 2004, was still the only product of its kind in these markets. At the end of December 2005, the target reached up to 1,342,000 and it is estimated that around 2 million are sold out so far (April 2006) in France itself.

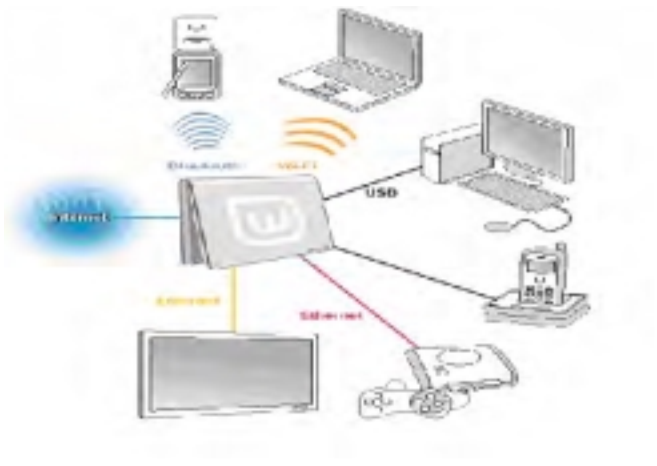


Figure 2: LiveBox's connections to devices

Ultimately, the goal is to develop a versatile residential gateway that will support QoS and paves a way to increase the revenues for service provider as well as satisfying the user needs. As QoS is a major criteria, the challenge here is dependability [2]. It clearly involves availability, security, reliability and usability. Work currently being undertaken includes how failures occur, how systems can be made more reliable and secure, how various issues such as infrastructure, responsibility, heterogeneity, risk and maintainability are addressed within the areas of advanced home technologies.

3. Residential gateway and dependability

The main aim of the work is to minimize regular failures and to avoid catastrophic failures in the future through a pragmatic approach, i.e., the analysis of field failures. We also intend to increase the efficiency of fault diagnostic and correction mechanisms, tasks undertaken by hot line operators. The starting point is to search for accidental faults (software, hardware, and human) and malicious faults (attacks), then to identify and to classify those faults in order to build a fault model.

• Fault model

Fig. 3 shows a simple architectural fault model. Clearly analyzing the field failure data's, we are describing the origin and all the problems and conditions. We also evaluate the cause-effect of operational data, so that it paves a way to build a fault model. Solutions and improvements will be provided to LiveBox actors of various phases.

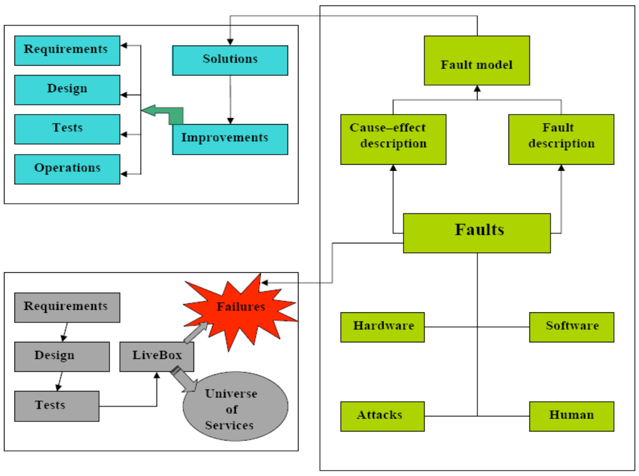


Figure 3: Fault model

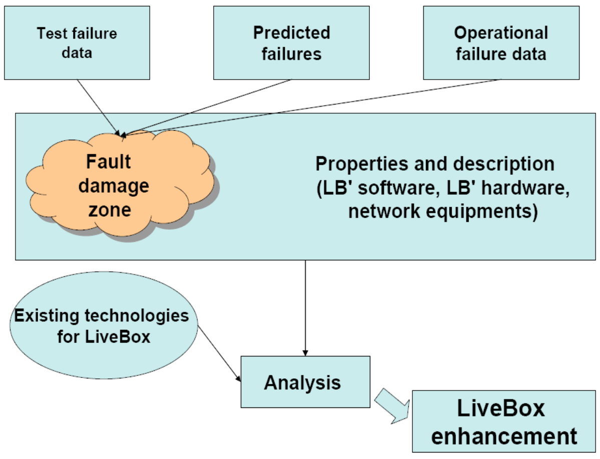

Test failure data
Predicted failures
Operational failure data
Fault damage zone
Properties and description (LB' software, LB' hardware, network equipments)
Existing technologies for LiveBox
Analysis
LiveBox enhancement

Figure 4: Functional diagram

We envisage analyzing the failures using two different techniques depending upon their role and complexity.

Fault tree and event tree analysis

Fault tree analysis (FTA) is a deductive, top-down method of analyzing system design and performance. This approach starts with a top event (catastrophic failure), followed by identifying all the associated elements in LiveBox that could cause the top event to occur. An *event tree analysis* (ETA) is a visual representation of all the faults which can occur in a system. As the number of fault increases, the picture fans out like the branches of a tree. Event trees can be used to analyze LiveBox in which all components are continuously operating, or in which some or all of the components are in standby mode. The starting point disrupts normal LiveBox operation. The event tree displays the sequences of events involving success and/or failure of the LiveBox components.

Fault modelling goals

In Fig. 4, we try to understand the origin of faults, its type, its gravity, its urgency and group them together in a way to:

- reduce the number of individual defects that have to be considered,

- reduce the complexity of the device description ,

- allow analysis for LB enhancement in all steps of LB lifecycle.

Next step

Designing and validating a benchmark for the LiveBox are the next goals of the study. It has various subtasks and each will be clearly worked out. First task would be to develop a benchmark for various dependability metrics (e.g., availability, reliability or security) considering all the desired features. It can be done either by using field failure data, or drawing inspiration from other standards. The validation phase is to observe the responses of the LiveBox to various injected faults. Finally, guarantying the dependability features and ways to increase the security are part of an appropriate design, required to meet customer's needs. Overall, home technological solutions require a dependable base set of criteria to be met as well as the technology is evolving, need is not static; people's relationship to technology in the home is constantly changing.

References

[1] R. McLellan et al., "Residential Gateways in Full Service Access Networks", Bell Labs, Lucent Technologies, Murray Hill, NJ, USA, http://cm.bell-labs.com/cm/cs/who/rae/noc.pdf

[2] G. Dewsbury et al., "Designing Appropriate Assistive Technology for Home Users: Developing Dependable Networks", CIB Working Group, October 2002, Roma, Italy,
http://www.dirc.org.uk/publications/inproceedings/papers/54.pdf

Modelling dependence and its effects on coincident failure in Fault- Tolerant, Software-based systems

Kizito Salako

The Centre for Software Reliability,

City University, London. e-mail:kizito@csr.city.ac.uk

INTRODUCTION

Everyday parlance, such as “Two heads are better than one” or “Belts and Braces”, points to the use of redundancy and diversity as a “common-sense” means of achieving high levels of reliability. In the design of software-based systems the use of diverse redundancy, in a bid to tolerate software faults, is well known [1,2]. These systems, ***Fault tolerant Software-based systems*** (FTS systems), achieve fault tolerance by the implementation of non-identical, functionally equivalent, software components. Examples of some FTS system architectures include N-modular redundancy (e.g. majority voted systems[1]) and N-version, parallel redundancy (1-out-of-N systems)[2] where software components make up individual software channels of the N-version system (so N software channels in total). In particular, 1-out-of-N systems ensure that the reliability of the N-version system can be no worse than the reliability of the individual software components. This is because the software components of the system may not have identical failure behaviour; they may fail on different subsets of the set of possible demands that may be submitted to the N-version system in operation. The software components are said to exhibit a level of ***failure diversity.***

In order to stimulate failure diversity between the software versions the development process of each software channel[3] may be carefully managed by a centralised, management team. This team oversees the N-version system’s development process. There are several, possible management policies. Some of these are

- Perfectly isolated software development teams. The teams cannot communicate with one another. This removes the possibility of a mistake or erroneous “point of view” in development propagating from one team to another. To achieve this physical isolation of the teams may be complemented by the controlled dissemination of information, by the centralised management team, to the development teams for each channel;
- In addition, diversity between the isolated teams in the circumstances surrounding the way they develop their respective software channels, may be forced (***forced diversity***). For instance, the teams may have different budgetary constraints or different design and development methodologies (different programming languages, algorithms, Integrated Development Environments, e.t.c.) for their respective software development efforts;

[1] A majority-voted system is one in which correct system operation is guaranteed if and only if a majority of the software channels operate correctly.

[2] A 1-out-of-N system is one in which the success of the system, in operation, is guaranteed whenever any one of its component software versions succeeds.

[3] Each channel is developed by a unique, development team. Thus, for N software channels there are N development teams.

Both of these scenarios, and their consequences for system reliability, have been studied in literature. Eckhardt and Lee [4] were the first to mathematically model, in particular, coincident failure in an N-version parallel redundant system. Their model considers perfectly isolated software development teams that develop each of the systems N software channels. Among other things their model demonstrates that *independently developed software versions may not fail independently*. Intuitively this is a result of the teams finding similar tasks in development equally difficult despite being perfectly separated from each other. So the teams are likely to make the same mistakes and therefore are correlated in their probabilities of failure. Consequently, their versions are correlated in their failure behaviour; when one version fails on a demand submitted to the system in operation the other versions are also likely to fail on the same demand. However, Littlewood and Miller [5] extended this model by considered forcing diversity between the development processes of isolated, development teams. As a result of forcing diversity between the teams development processes the teams may become diverse in how difficult they find similar development tasks. Thus, the probabilities of failure of their respective software versions could be negatively correlated. Also both of these models also showed that the expected reliability for a 1-out-of-2 system depends on the failure diversity between the versions that make up the system as well as the reliabilities of the individual versions.

Both of these models are based on the requirement of perfect isolation between the development teams. I have extended these models by relaxing this requirement and modelling possible forms of dependence between the development teams. Relaxing "perfect isolation" is useful for the following reasons.

- Perfect isolation is difficult to justifiably achieve in practice. It is important to understand when it is violated (even small departures from it) and what can be expected for system reliability as a result of this;
- The centralised management team, by allowing useful interaction between the development teams, might wish to improve the reliabilities of the versions that make up the FTS system's software channels;
- Economic constraints may require the sharing of tasks and resources by the development teams e.g. subjecting their software versions to the same testing regimes;
- The development process can be very complex and random with many possible dependencies. It may be practically infeasible to know the precise outcome of a given stage in the process. For instance, the space of possible conversations, relevant to developing the FTS system, which may occur between the teams might be exceedingly large. Hence, it will be useful to capture this uncertainty by considering the effect of not knowing which of the possible, relevant conversations has taken place.

In what follows I shall give a brief description of the model extensions.

1. MODELLING THE DEVELOPMENT OF AN FTS SYSTEM

The development process and the operation of a FTS system are ***random processes***. That is, the outcome of the various stages of the development process and the failure behaviour of the resulting FTS system in operation can be modelled as random variables. These include such stages as system design, writing the initial system specification, prototyping, verification, and validation. The possible dependencies between these various events may give rise to complex interactions which are adequately modelled by ***conditional independence relations*** between the random variables.

Graphical models, in particular ***Bayesian Belief Networks*** (BBN), can be used to describe and analyse the various probability distributions over the random variables of interest. The BBN in Fig. 1 is an example of a fairly complex topology that may arise from modelling a system's development process and its operation. Each node represents a random event in the system's development or operation. The edges between the nodes (more precisely, absence of edges) depict the conditional independence relationships between the events. The stages in the development processes of the redundant components/channels that make up the FTS system may have multiple dependencies between them. Each channel is developed by a unique, development team. In the case of perfect isolation between the development teams the nodes I and J in the BBN would not exist. That is, the nodes making up the development of Channel A would be disjoint from the nodes that make up the development of channel B. This is a generalisation of work carried out by Popov and Littlewood [6].

The BBN in Fig. 2 simplifies the one in Fig.1, preserving the dependencies between the two development processes of channels A and B, represented by the nodes *I* and *J*. There is an equivalence class of BBNs that can be transformed into this form. We can, therefore, make general statements about a wide range of possible practical scenarios by restricting our analysis to BBNs like Fig. 2, focusing on dependencies between the two processes.

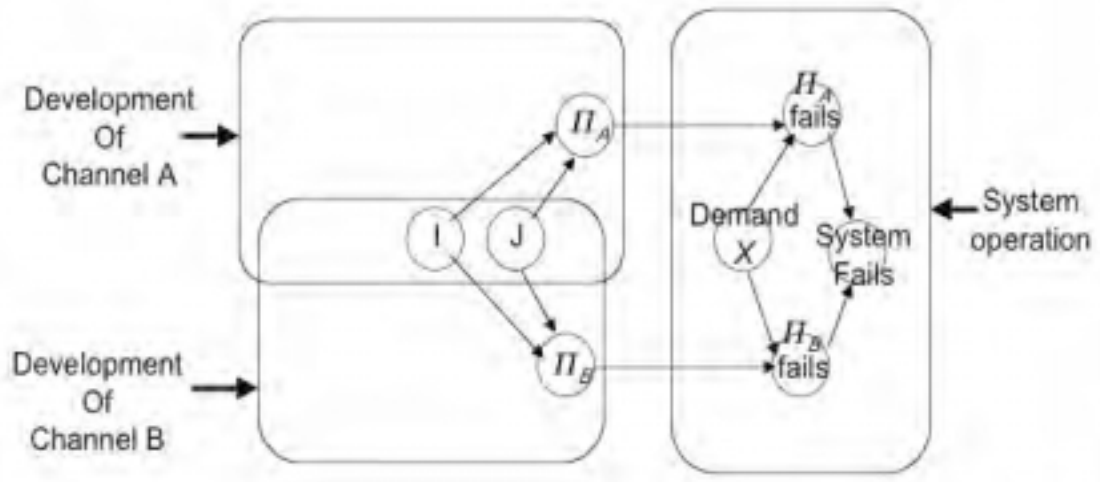


Fig. 2

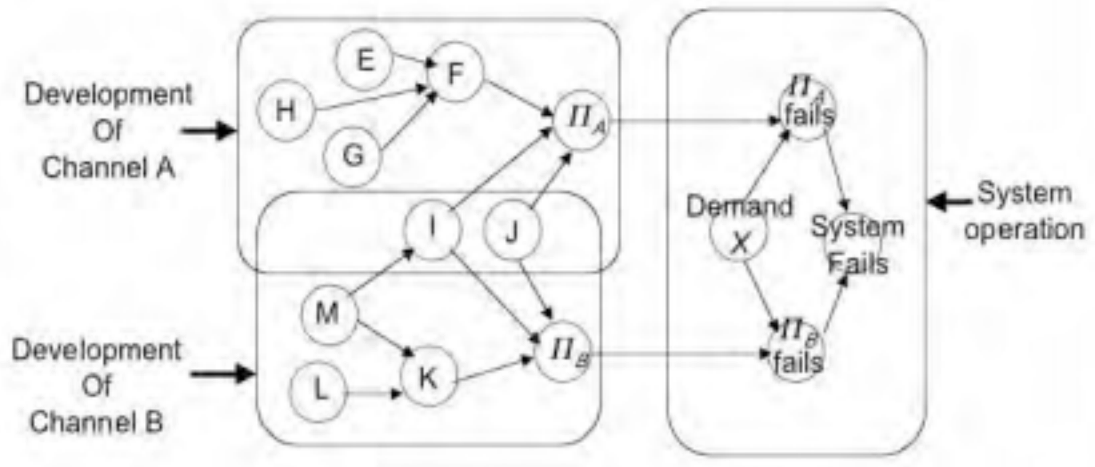


Fig. 1

2.DISCUSSION

There is not complete agreement, in industry and in the literature, on the best way to manage the development of a Fault-tolerant software system. Some practitioners point to the possibility of mistakes or erroneous viewpoints propagating between the development teams as a valid reason for not allowing the teams to communicate with each other [1]. On the other hand, others point to the expected improvement in the reliabilities of the component software versions, as a result of useful inter-team interaction, as a definite benefit for overall system reliability [7]. By modelling dependencies (in the FTS system's development process and operation) as conditionally independent events there are several results that either confirm or refute these claims under various scenarios. Hence, the results provide a formal basis for clearly stating the consequences of preferred practices in a development process. Some of the achievements of my research are

- the enumeration and analysis of many forms of dependence, all of which can be captured by the same probabilistic modelling approach. These include scenarios of dependent, forced diverse events between the teams, e.g. The teams implement the same algorithm (***dependence***) using a choice of different programming languages (in particular, this is a ***forced diverse*** choice of possible control sequences, method definitions and options for data manipulation);
- the identification of scenarios where dependence between the developments of redundant components will always be "a bad thing". That is, in these scenarios the existence of some dependence between the development teams always results in no improvement for system reliability. So keeping the teams isolated is the preferred policy under these circumstances;

- the identification of scenarios where the use of forced diversity always results in system reliability that can be no worse than otherwise. These scenarios include those where the development teams may have some dependence between them.;
- The models, and their implications, ***are useful in reasoning about any random process*** that is ***characterised by common dependencies*** between multiple random events. e.g. complex systems employing human-machine diversity (air-traffic control systems), hardware and/or software diversity (critical infrastructure).

Possible directions for future work include studying the implications, in terms of system reliability, of dependence between the nodes of large networks (e.g. critical infrastructure, ubiquitous systems, e.t.c). These networks, in particular the subsystems thereof, tend to exhibit dependence between their component nodes. As such, it is natural to expect that the probabilistic modelling approach for reasoning about FTS systems will provide useful insight into possible failure correlation between the network nodes and subsystems.

REFERENCES

[1] A. Avizienis, "The methodology of n-version programming," in *Software Fault Tolerance*, M. Lyu, Ed. John Wiley & Sons, 1995, pp. 23–46.

[2] M. Lyu and Y. He, "Improving the n-version programming process through the evolution of a design paradigm," *IEEE Transactions on Reliability*, vol.R-42, pp. 179–189, 1993.

[3] B. Littlewood and L. Strigini, "A discussion of practices for enhancing diversity in software designs," Centre for Software Reliability, City University, DISPO project technical report LS-DI-TR-04,2000.[Online].Available:http://www.csr.city.ac.uk/people/lorenzo.strigini/ls.papers/DISPO2000diversityEnhancing/

[4] D. E. Eckhardt and L. D. Lee, "A theoretical basis for the analysis of multiversion software subject to coincident errors," *IEEE Transactions on Software Engineering*, vol. SE-11, pp. 1511–1517, 1985.

[5] B. Littlewood and D. R. Miller, "Conceptual modelling of coincident failures in multi-version software," *IEEE Transactions on Software Engineering*,vol. SE-15, pp. 1596–1614, 1989.

[6] P. Popov and B. Littlewood, "The effect of testing on the reliability of fault-tolerant software," in *DSN 2004, International Conference on Dependable Systems and Networks*. Florence, Italy: IEEE Computer Society, 2004, pp. 265–274.

[7] Y.C. Yeh, "Design Considerations in Boeing 777 Fly-By-Wire Computers," in Proc. 3rd IEEE International High-Assurance Systems Engineering Symposium, pp. 64, Washington DC, 1999

Session on Model-based Verification

Chair: Paolo Masci, University of Pisa, Italy

Detecting data leakage in malicious Java applets[1]

Paolo Masci, Dip. di Ingegneria dell'Informazione, Università di Pisa

Web applets are programs dynamically loaded and executed inside the Internet browser of users' machines. They are used to extend the functionalities of web pages. Web applets can be associated with specific profiles granting access to users' information. As a consequence, web applets may possibly disclose, intentionally or by error, confidential information on public channels. We propose a technique to analyze the compiled code of web applets before execution. The technique is based on abstract interpretation. Users' data is associated with security levels and an iterative analysis is performed to trace information flows.

Introduction

New generations of Internet services are heavily based on sensitive contexts. E banking, e commerce, data streaming, they all require access to a number of users' data in order to be customizable. Classic html programming language alone is unable to exploit these features. As a consequence, during the last few years web pages have started to embed special objects in order to extend the capabilities of the html language. At the moment, besides simple scripts that can change visual styles, web pages can also include programs written in a number of different languages.

Web applets are programs written in Java [4]. They are one of the most widespread objects embedded inside web pages. Basically, web applets are compiled Java programs that can be executed within the users' browser. They generally have less capabilities than stand alone programs since an access control mechanism, called Java sandbox, is used to limit their access rights to system resources. Nevertheless, the Java platform includes also mechanisms for granting special rights to specific web applets. This necessity arises since useful programs generally need users' data in order to perform a task. For example, a program for money transactions may require access to a certificate or a key. Nevertheless, granting access rights for sensitive information may possibly cause

[1] Joint work with M. Avvenuti, C. Bernardeschi, N. De Francesco

security issues since these information may be released on public channels. In fact certification systems, together with access control mechanisms, are not able to detect leakage of data when programs have already gained access to such data.

The security paradigm of the JDK 1.1 represents an emblematic case of study of a system providing an access control mechanism which is not able to detect leakage of sensitive data [2]. Security for JDK relies on the identity base, that is an object containing a database of certificates and keys. This object is used as a central repository to hold information useful for authentication purposes. Sensitive information, such as private keys, is protected through interface methods, which means that data stored in the identity base can be obtained by programs only through methods of predefined static classes (classes belonging to the java.security package, in this case). This way, programs have access to public information, like public keys and certificates, but they cannot read private keys. Private keys of the identity base can be accessed only by programs belonging to security packages of the Java platform installed on the system. Applets cannot belong to such package. Nevertheless, this paradigm is not secure since the identity base is stored in serialized form in a file, and applets may read this file and send the serialized data to a malicious host through a socket connection. On the receiving host an attacker can easily read private keys by deserializing the file directly on its own machine.

This shows that classic protection paradigms may not cover all possible unauthorized disclosure of data. As a consequence, new mechanisms are needed to protect computer systems. In particular, the information leakage mentioned above can be revealed by analyzing control flows in the programs. The theory that studies these security issues is known in the literature as secure information flow [3].

1. Our contribution

We propose a technique to check secure information flow in Java programs, with specific reference to web applets. The technique is based on a work proposed in [1], where a multilevel security policy that assigns security levels to variables in the program is used. The bytecode is processed by an abstract interpreter through an analysis that is similar to that of the bytecode verifier. The analysis is performed method by method, assuming other methods correct when verifying a method. A context is used to maintain security levels of methods and objects in the heap. The analysis abstracts real values of instruction operands into security levels, which represent the maximum privacy level of data affecting such values. Sensitive data are marked with high levels, and public channels with low levels. Security levels form a lattice, partially ordered by $\sqsubseteq$. A program does not leak sensitive data if, on program end, channels with security level contain data with level $\sqsubseteq$.

The analysis is performed directly on the Java bytecode. Since the bytecode is unstructured, the analysis includes also mechanisms to compute execution paths and scope of implicit flows. Execution paths are identified through the *control flow graph*, that is a set of nodes representing instructions. Instructions are identified by numbers (the position of the instruction in the code). There is an edge from node *i* to *j* if instruction *j* can be executed directly after *i*. The scope of implicit flows is discovered through *control regions*. A control region is a subset of nodes in the control flow graph, and represents the biggest set of instructions control dependent on a given conditional instruction. A control dependence is defined as follows: instruction *j* is control dependent on *i* when *i* is a conditional instruction (`if`, `switch`, ...) that determines whether an execution path will pass through *j*. Control flow graph and control regions can be computed statically.

1.1. Rules of the abstract interpreter

The abstract interpreter uses a set of rules to model the behavior of the Java VM. Implicit flows are represented in the rules by a local environment which reflects the security level of the control regions. There is a rule for each kind of instruction. Each rule computes security levels on the basis of its own local environment and operands.

The rules define a relation $\rightarrow \subset V \times V$, where V is the set of the abstract interpreter states. A state V V has the form <Q, SL, P>, where Q is the set of all machine states of the instructions, SL is a security table containing the security levels of conditional instructions and entry point of a method, P is a security context. The abstract interpreter builds a chain of machine states by applying the rules. The chain ends when no rule can be applied. This corresponds to the end of the analysis, which happens in two cases: i) when a rule detects an illicit flow, ii) when the security context is unchanged after analyzing all methods. In this last case the bytecode has secure information flow, thus the program can be safely executed.

Each rule has a precondition and a postcondition. The precondition specifies the position of the instruction in the bytecode and the state in which the instruction is executed. The postcondition defines how to compute the new state of the nterpreter on the basis of the semantics of the instruction. The bytecode is modeled as an array called B. The element at position i in the array corresponds to instruction at position i in the bytecode. Q_i denotes the state of instruction at position i. Every state Q_i is a pair (M,St), where M is the memory and St is the operand stack of the virtual machine. M(x) is a mapping from local variable x to security levels. St is a sequence of security levels. The standard concatenation operator is · and the value on top of the stack appears on the left hand side of the sequence. The least upper bound operator between security levels is $\sqcup$. The environment of instruction at position i is denoted as env(i). Rule for the bytecode instruction load x is shown here as example. The instruction loads the content of register x on top of the operand stack.

$$\frac{B[i] = \ \text{load } x,\ Q_i = (M,St)}{<Q, SL, P> \quad <Q[Q_{i+1} = Q_{i+1} \sqcup (M, (env(i) \sqcup M(x)) \cdot St)], SL, P>}$$

1.1. Software Tool

We developed a prototype tool that can be installed on users' machines. The tool is composed of two components: the browser agent (BaT) and the verifier tool (SifJVM). Calls to the Java VM are generated by the browser when loading Internet pages. Basically, the agent BaT intercepts these calls, temporary blocks the applet, and dispatches the bytecode to the verifier tool SifJVM. The verifier tool analyzes the information flow in the applet and returns the final result to the agent. The agent enables applet execution only if the analysis succeeds.

The agent is developed through the Java VM Tool Interface (JVMTI), that is publicly available from Sun. JVMTI is a programming interface that notifies events to registered agents and provides mechanisms to control the internal state of the Java VM. Agents are registered through a command line option when starting up the VM. The verifier tool is based on an open source project of the Apache Foundation called ByteCode Engineering Library (BCEL). It is a set of APIs used to reading and writing Java bytecode. The BCEL package includes also a bytecode verifier called JustIce. The two main modifications to BCEL/JustIce were made to handle security levels instead of types, and to implement the semantics for instructions according to our rules.

2. Conclusions

In this paper we presented the key points of an approach to check secure information flow in web applets. The analysis is performed through abstract interpretation of the bytecode, where real values are replaced by security levels of data affecting such values. The abstract interpreter models information flows through the semantic of instructions and discovers illicit flows in a finite number of steps. We presented also the basic structure and functionalities of a prototype tool that can be installed on users' machines.

References

[1] R. Barbuti, C. Bernardeschi, and N. De Francesco. Analyzing Information Flow Properties in Assembly Code by Abstract Interpretation. The Computer Journal, 2004.

[2] J.P. Billon. Security Breaches in the JDK 1.1 beta2 security API. 1997.

[3] D.E. Denning. A lattice model of secure information flow. Communications of the ACM, 1976.

[4] T. Lindholm and F. Yellin. Java Virtual Machine Specification, 2nd edition. Addison Wesley Longman Publishing Co., Inc., 1999.

Handling Large Models in Model-based Testing

Zoltán Micskei

Department of Measurement and Information Systems
Budapest University of Technology and Economics
Magyar Tudósok krt. 2., H-1117 Budapest, Hungary
micskeiz@mit.bme.hu

Introduction

Model-based testing [1] aims to partially automate the labor-intensive tasks in traditional software testing. Creating an *effective test suite* containing relevant tests cases with input and expected output events needs usually a lot of manual work and expert knowledge. Constructing first a model to use it later as a *test oracle* can ease for example the test generation, conformance testing and test suite evaluation subtasks.

In the past years many research paper was published on model-based testing. The participants of the AGEDIS [2] European Union project built their own tool chain supporting the whole testing process starting from annotating the system's UML model to assessing the test execution results. Ammann et al. [3] used mutation analysis techniques to generate test cases from specification with the help of external model checkers. The BZ-Testing-Tool [4] uses an underlying constraint solver to derive test suites according to various testing strategies and coverage criteria. As we can see, numerous methods were proposed for model-based testing.

1. Test Generation using Model Checkers

In our previous work [5] we analyzed the *effectiveness* of different configurations of a model checker tool when used for test generation purpose. Figure 1. depicts our whole tool chain. The model checker was utilized in the test generation phase.

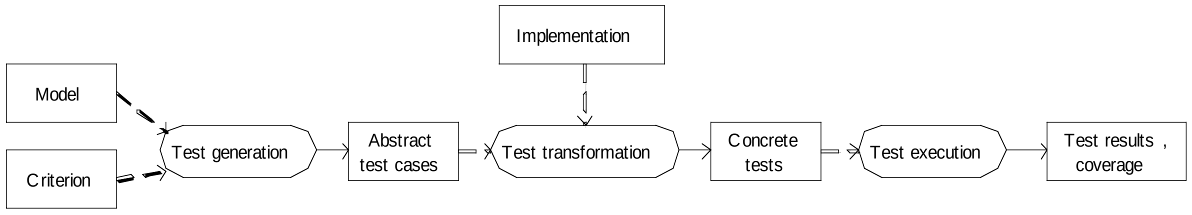


Figure 1. Testing framework

The main issue in using model checker tools is that the default configuration of these tools is optimized for the *exhaustive search* of the full state space while test generation only needs to find a *short counter-example* representing a test case. Fortunately, state-of-the-art model checkers provide a wide variety of options and techniques to tune the behavior of the checker to fit for this special purpose. Our experiments with the SPIN model checker showed that the proper configuration could decrease the execution time with more than 60%, not to mention that the length of the test cases was reduced to 5% of the original test suite (generated by using the default configuration).

The applicability of the framework was measured using a *real-life industrial model*. The main difference compared to the previous experiments was the size of the state space. The statechart model that described a bit synchronization protocol had 31 states and 174 transitions, resulting in a state space with more than $2\ 10^8$ states, which would need approximately 40 GB memory. State compression techniques and reduction of the behavior were effective to make the test generation feasible. After applying all optimization the test suite covering all states was created in 65 minutes. Thus, the framework turned to be applicable for large models as well, but the practical limitations caused by the state space expansion and the long execution time cannot be fully avoided.

2. Techniques for Dealing with Complexity

The previous section showed that the bottleneck of using model-based testing methods is the size of real-life models. Hence, we started to analyze the following methods to handle complexity.

Use of advanced model checking techniques: In the past years, bounded model checking was introduced [6], which finds counter-examples very fast. This property is perfectly suited for test generation. Other benefit of the technique is that it usually needs much less memory than explicit model checking or even symbolic techniques using BDDs.

Combining model checking with other verification methods: A promising direction is to combine model checking with other formal verification methods to generate test cases. Theorem proving is one of the candidates as stated in [7] where the theorem prover was used to reduce the problem to a finite-state form.

Model-based slicing: Slicing [8] has a long tradition in program analysis. The main idea is to focus only on that part of the program, which is relevant for the property currently checked. The other parts are temporally deleted, thus the size of the program in question is significantly reduced. The same method could be used on the level of the model, e.g. when searching for a test sequence satisfying a coverage criterion, only those parts of the model are preserved, which can have influence on the given criterion.

A typical important verification task in a resilient system is to evaluate the protection and defense mechanism of components. Although, these mechanism usually are of manageable size, their testing is not possible without modeling also the complex components they protect. However, when testing a protection mechanism only a small part of the component is relevant, i.e. where the fault occurred. The other parts do not influence the outcome of the tests; hence, the behavior of them can be abstracted. That is why automatic abstraction and efficient state space handling methods mentioned above play a key role in the verification of dependable systems.

3. Conclusion

In this paper we presented the concept of model-based testing and our framework that supports test generation using model checkers. We outlined the results obtained from experimenting with real-life models, it turned out that the huge state space of industrial models require further optimization techniques. A key method to handle complexity can be the use of various abstractions, e.g. model-based slicing and automatic theorem proving, to focus only on relevant parts of the model. The integration of test generation with these abstraction techniques, which proved themselves in many previous research projects, is a challenging open question. However, it is also an opportunity to benefit from the expertise of fellow researchers in the network, and build methods reducing the model-based testing problem to be applicable in everyday engineering use.

References

[1] M. Leucker *et al.* (Eds.), *Model-Based Testing of Reactive Systems*, Advanced Lectures, Series: Lecture Notes in Computer Science, Vol. 3472, 2005, VIII, 659 p., ISBN: 3-540-26278-4, 2005.

[2] A. Hartman and K. Nagin, "The AGEDIS Tools for Model Based Testing," in *Proceedings of ISSTA 2004*, Boston, Massachusetts, USA, July 11-14, 2004.

[3] P. Ammann, P. E. Black, and W. Ding, "Model Checkers in Software Testing," Technical Report, NIST-IR 6777, National Institute of Standards and Technology, 2002.

[4] B. Legeard *et al.*, "Automated Test Case and Test Driver Generation for Embedded Software," in *Proc. of International Conference on Software, System Engineering and Applications*, pp 34-39., 2004.

[5] Z. Micskei and I. Majzik, "Model-based Automatic Test Generation for Event-Driven Embedded Systems using Model Checkers" to appear in *Proc of DepCoS '06*, Szklarska Por ba, Poland, 25-27 May 2006.

[6] E. Clarke *et al.*, "Symbolic Model Checking without BDDs," in *Proc. of Tools and Algorithms for Construction and Analysis of Systems (TACAS '99),* Amsterdam, The Netherlands, March 22-28, 1999.

[7] N. Shankar, "Combining Theorem Proving and Model Checking through Symbolic Analysis", *CONCUR'00: Concurrency Theory*, LNCS 1877, pp 1-16, Springer Verlag, 2000.

[8] M. Weiser, "Program slicing," in *Proceeding of the Fifth International Conference on Software Engineering*, San Diego, CA, USA, pp 439-449, May 1981.

Testing applications and services in mobile systems

Nguyen Minh Duc
LAAS-CNRS, 7 avenue Colonel Roche, 31077 Toulouse France
minh-duc.nguyen@laas.fr

Abstract

Compared to “traditional” applications, mobile applications provide a new set of challenges from the perspective of testing. These challenges have gained only little attention in the literature so far. The aim of this paper is to give an analysis of the new problems raised by testing applications and services in mobile settings, and to propose direction for my PhD work.

1. Introduction

Advances in wireless networking technology have yielded the emergence of the mobile computing paradigm. Generally speaking, mobile computing calls for the use of mobile devices (handset, PDA, laptop…) that move within some physical areas, while being connected to networks by means of wireless links (Blue-tooth, IEEE 802.11, GPRS…). Two of the main characteristics of mobile applications and services are mentioned below:

- High error rates. The movement of mobile devices causes disconnections. As a result, network topology changes frequently. Moreover, limited capacity of mobile device, scarce bandwidth in wireless connection… make the computation more difficult. This means that errors are likely to occur during the operation of mobile systems.
- Context dependency. The context includes any detectable and relevant attribute of a device, of its interaction with other devices and of its surrounding environment at an instant of time. The context is continuously evolving, so that mobile applications have to be aware of, and adapt to, the induced changes.

Such characteristics provide new challenges for the resilience technologies. As regards verification, the development of appropriate technologies has been seldom explored so far, and remains an open issue.

The objective of my work is to develop solutions for the testing of applications and services in mobile systems, with consideration for both functional and robustness requirements. This work has started in November 2005, and is conducted in the framework of the European project HIDENETS (HIghly Dependable ip-based NETworks and Services). HIDENETS aims to develop and analyze end-to-end resilience solutions for distributed applications and mobility-aware services, taking the example of the automotive domain with car-to-car and car-to-infrastructure communication scenarios.

The remainder of this paper is organized as follows: Section 2 discusses the specific issues raised by testing mobile computing systems. Section 3 describes direction for my PhD work. The expected contribution covers both the determination of test platforms to perform controlled experiments, and the definition of a test strategy that combines contextual parameters with inputs liked to the logic of applications. Section 4 gives some concluding remarks.

3. Problems in testing mobile computing systems

Testing [1] is the most widespread verification technique. It consists in executing a system by supplying it with input values. The outputs provided by the system in response to the inputs are then analyzed to determine a verdict. The test activities generally involve several steps, each focusing on a given level: testing at the unit level, integration testing, and system testing.

The problems raised by mobile computing systems are discussed hereafter, by considering both fundamental issues of testing and technological issues. The fundamental issues are:

- The determination of testing levels;
- The test selection problem;

- The so-called *oracle* problem, or how to determine the correctness of the test outputs.

Technological issues concern the platform required to control and observe test experiments.

How to determine the testing levels for mobile-based applications and services?

The determination of a testing level requires the determination of the (sub-)system that will be the target of testing, and of its interface to its surrounding environment. Typically, the testing levels are determined in the framework of an integration strategy that progressively aggregates the system components until a system level test is performed. For mobile applications in ubiquitous communication scenarios, a difficulty is that the notions of system boundaries, and of system components, are not as clear as for traditional applications. Let us take the example of a car application involving both car-to-car and car-to-infrastructure communication. A single car can be the system under test, and the neighbouring cars and infrastructure servers can be the environment. Or a car *and* an infrastructure server can be the target system, with the rest of the world forming the environment. Or the target system can be a *set* of cars traversing a geographical area, and the fixed infrastructure in that area is the environment, etc. Note that, at a given level, alternative system instances can be considered (e.g., instances with sets of cars corresponding to different traffic loads). Moreover the composition of the system may continuously vary during testing. Generally speaking, the determination of the testing levels should depend on the target application and on the target properties to be verified. To the best of my knowledge, there currently exists no practical guidelines to address this problem.

Which test selection technique to be used?

The traditional approaches in software testing can be classified into two categories: white-box selection approaches, and black-box ones.

White-box approaches are based on structural coverage criteria. They are applied for small pieces of software, typically at a unit testing level. An underlying assumption is that the behaviour of the target piece of software is governed by its structure. But in mobile applications, this assumption may not hold. Software running in mobile devices depends on not only the application logic, but also on conditions on contextual parameters that are not explicit in the applicative code. Such a situation is exemplified by [2], where the authors study a context-sensitive middleware-based application. The authors point out that, even for unit testing, a traditional method (*all-du* coverage in this case) is not sufficient because the interesting contextual conditions reside in the middleware, not in the applicative code.

Black-box approaches need a model of the functions of the application. To the best of my knowledge, there is no standard specification language for mobile computing systems. Some authors propose using UML or Mobicharts [3, 4, 5, 6], but the corresponding modelling approaches fail to account for all aspects of mobile computing, such as network bandwidth, neighbouring devices… From the concrete examples of test experiments reported in the literature, it seems that the typical practice is to select test cases manually from the informal application requirements.

Hence, finding effective test selection techniques for mobile applications is still an open problem.

How to solve the oracle problem?

Developing an oracle procedure to compare actual test outputs with the expected ones is a fundamental issue for software testing. The best solution is to have an automated oracle based on a formal specification. However, as mentioned above, the specification of mobile applications is itself a problem.

Moreover, in mobile settings, it may be the case that part of the inputs is uncontrollable and unobservable. For example, if a device is tested in a real environment, neighbouring devices can join and leave network in an unpredictable manner. The resulting inputs for the target device cannot be determined, and the expected outputs cannot be decided. The solution in this situation is to develop *partial* oracles that perform (weak) plausibility checks. Examples of such oracles for mobile applications are provided in [7, 8].

Which test platform to be used?

A test platform must offer facilities to observe and control the transit of input and output events, received or transmitted from each component. This is typically done by means of Points of Control and Observation (PCO). A PCO can be global or local. In centralized test architectures, a global PCO is generally easier to implement since only one PCO is needed. In distributed test architectures, local PCOs need to be synchronized. Compared to traditional systems, mobile computing systems need a higher complexity of

PCOs in both cases, in order to adapt to the high dynamicity due to the mobility of entities and rich contextual parameters.

The platform may be more or less realistic with respect to the operational environment. In practice, it is difficult and expensive to test some mobile applications that require checking the combined logic of the involved hardware and software components. For example, an application running in mobile terminals like PDA, smart-phone… may have to be tested on different types of devices, and on different wireless local networks. Or in automotive domain, realistic platforms involve prototypes that are implemented into real cars (see e.g., [9]). The cost of such platforms, as well as controllability and observability constraints, imply that part of the testing activities may preferably be performed using emulation/simulation facilities.

Generally speaking, there is a trade-off to be found between the need for realism, and the need to support fine-grain control and observation to implement a given test strategy.

3. Contribution of the thesis

My work will focus on testing mobile applications and middleware services in a simulated environment. It will be supported by case studies that are a relevant to the automotive domain. A discussion of the test platform and test strategy to be considered is provided below.

Platform for testing

The test platform will consist of three categories of components: Application execution support, Network simulator, and Context controller as shown in Figure 1. Concrete examples of platforms built according to this generic architecture are provided in [10,11].

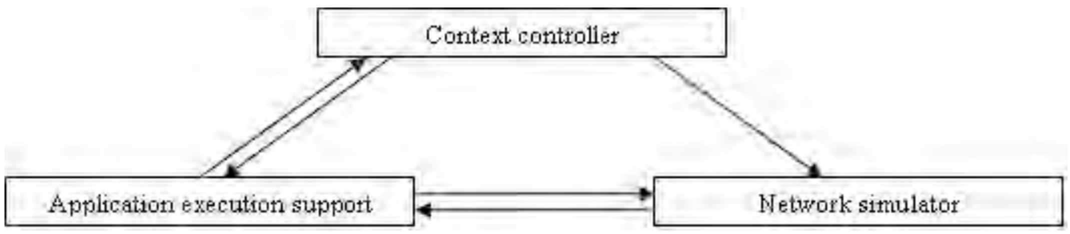


Figure1: Platform for test execution

The Application execution support is needed to emulate the executive support for the applicative code. A requirement is to provide an interface to the context controller and network simulator, so that the application can interact with them as if it would interact with a real environment.

Since applications and services connect over wireless link, the simulated environment must support a model of the underlying network. The network simulator is responsible for simulating the full functionality of a real wireless network: different types of wireless connection, account for location information of mobile devices participating in network, failures of mobile devices or wireless links… Network simulators like ns-2 or GlomoSim can be used.

The context controller is needed to simulate context information. Applications exploit context information to take different actions adaptively. Context is also needed by the network simulator to set up input parameters for imitating the real network characteristics. There have been several toolkits for simulating contexts in recent years. Some of them are built on 3D game engines and serve to simulate a first-person view of the physical environment, like Ubiwise [12]. In our case, we are planning to use either a generic location event simulator, like the GLS tool developed at Cambridge [13], or a microscopic traffic simulator as suggested by [10] for testing car-to-car communications application.

This simulated environment needs PCOs to control and observe the test experiments. Depending on the target application and test strategy developed, PCOs can be integrated in each component or can be realised as a global component added to the architecture.

Test strategy

I will investigate model-based strategies for testing both functional and robustness requirements. As the specification of mobile computing systems is not a mature issue, a first step will be to investigate adequate models for both the application functions, and their environment. The specification of the environment, including the threats that may affect the application, is expected to be the most challenging issue.

As regards test selection, compared to traditional software, there is a shift in the number and nature of input parameters to be considered: contextual parameters and their variation over time have to become first-class citizens. There is a need for methods to combine these parameters with inputs linked to the logic of applications, so as to obtain effective test data at a realistic expense of effort. At LAAS-CNRS, there is a long experience on model-based probabilistic methods for test generation [14]. Such probabilistic methods should be relevant to account for the uncertain characteristics of the environment, and to guide the sampling of meaningful input combinations.

4. Concluding remarks

Testing applications and services in mobile computing systems is a challenging issue that has been little addressed so far. The problems presented in this paper are expected to share commonalities with problems addressed by other resilience technologies. In particular, for the kind of systems that are targeted, the specification problem arise whatever the verification technology, testing or formal verification. Testing and experimental evaluation may share common requirements for test platforms allowing to perform controlled experiments. The student forum should be an opportunity to initiate discussion on these topics.

References

[1] Boris Beizer. *Software Testing Techniques.* Van Nostrand Reinhold, 1990
[2] T.H. Tse, Stephan S. Yau, W.K. Chan, Heng Lu. Testing Context-Sensitive Middleware-Based Software Applications, *Proceedings of the 28th Annual International Computer Software and Application Conference (COMPSAC 2004)*, pp.458-466, IEEE CS Press, 2004.
[3] Satyajit Achrya, Chris George, Hrushikesha Mohanty. Specifying a Mobile Computing Infrastructure and Services, *1st International Conference on Distributed Computing and Internet Technology (ICDCIT 2004)*, LNCS 3347, pp.244-254, Springer-Verlag Berlin Heidelberg, 2004
[4] Satyajit Acharya, Hrushikesha Mohanty, R.K Shyamasundar. MOBICHARTS: A Notation to Specify Mobile Computing Applications. *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)*, IEEE CS Press, 2003.
[5] Vincenzo Grassi, Raffaela Mirandola, Antonino Sabetta. A UML Profile to Model Mobile Sytem, *UML 2004*, LNCS 3273, pp.128-142, Springer-Verlag Berlin Heidelberg, 2004
[6] Hubert Baumeister et al. UML for Global Computing. *Global Computing: Programming Environments, Languages, Security, and Analyisis of Systems, GC 2003*, LNCS 2874, pp. 1-24, Springer-Verlag Berlin Heidelberg, 2003
[7] W.K. Chan, T.Y. Chen, Heng Lu. A Metamorphic Approach to Integration Testing of Context-Sensitive Middleware-Based Applications, *Proceedings of the 5th International Conference on Quality Software (QSIC'05)*, pp.241-249, IEEE CS Press, 2005
[8] Karl R.P.H Leung, Joseph K-Y Ng, W.L. Yeung. Embedded Program Testing in Untestable Mobile Environment: An Experience of Trustworthiness Approach, *Proceedings of the 11th Asia-Pacific Software Engineering Conference*, pp.430-437, IEEE CS Press, 2004
[9] de Bruin, D.; Kroon, J.; van Klaverem, R.: Nelisse, M.. Design and test of a cooperative adaptive cruise control system, *Intelligent Vehicles symposium*, pp.392-396, IEEE CS Press, 2004
[10] Christoph Schroth et al. Simulating the traffic effects of vehicle-to-vehicle messaging systems, *Proceedings of ITS Telecommunication*, 2005
[11] Ricardo Morla, Nigel Davies. Evaluating a Location-Based Application: A Hybrid Test and Simulation Environment, *IEEE Pervasive computing*, Vol.3, No.2, pp.48-56, July-September 2004
[12] J.Barton, V. Vijayaragharan. Ubiwise: A Simulator for Ubiquitous Computing Systems Design, *Technical report HPL-2003-93*, Hewlett-Packard Labs, 2003
[13] Kumaresan Sanmiglingam, Geogre Coulouris. A Generic Location Event Simulator, *UbiComp 2002*, LNCS 2498, pp.308-315, Springer-Verlag Berlin Heidelberg, 2002
[14] P. Thévenod-Fosse, H. Waeselynck and Y. Crouzet, "Software statistical testing", in *Predictably Dependable Computing Systems*, Springer Verlag, pp. 253-272, 1995

Behavior-Driven Testing of Windows Device Drivers

Constantin Sârbu

Department of Computer Science – Technische Universität Darmstadt

Hochschulstr. 10, 64289, Darmstadt, Germany

Email: cs@informatik.tu-darmstadt.de

1. Introduction

Commercial-off-the-shelf operating systems (COTS OSs) increasingly favor adaptability to support diverse application and hardware peripherals in detriment to targeting robustness of OS services. The COTS OSs interface to hardware devices is provided by the *device-drivers*[1]. Furthermore, drivers have themselves become add-on COTS components, enhancing OS's adaptability. Unfortunately, drivers constitute a prominent cause of computer system outages, impacting overall service reliability [5].

We note at least two facts that might explain why the drivers are the weak link in a computer system equipped with a COTS OS. First, drivers are relatively immature pieces of software, exhibiting a higher defect density compared with the OS kernel. One reason is that many hardware producers are forced by fierce competition to push their new peripherals onto the market before they could be tested properly. Additionally, the set of loaded drivers is likely to be different across installations. The difficulty to define all the interactions with other drivers or parts of the system significantly reduces the effectiveness of design-stage driver testing. Therefore, assessing continuous service provision in different environments and system configurations is still a difficult problem to solve.

Related Work: As drivers are COTS add-on components, they are delivered as binaries without source code. Thus the component-user's test campaigns are often limited to a black-box testing strategy, using only the OS driver interface specification. Several approaches have used Fault Injection (FI) techniques [1, 5] to test black-box OS extensions. The location of injection probes and the triggering instance of the actual fault injection

[1] Onwards, we use the general term "drivers" to refer to them

are either empirically or randomly chosen. Though, in the area of defect localization in software, research showed that faults tend to cluster in certain parts of the code [2]. Along the same line, another idea is to focus testing on functionalities that have a high probability of occurrence in the field [7].

In our current work[2] we provide COTS component-users with novel, non-intrusive methods for assessing the level of system robustness for a given HW/SW configuration. Based on ascertaining test progress from a driver-under-test operational state model, our scheme enhances the level of accuracy of existing test procedures.

2. System Model

By *COTS OS* we imply any monolithic operating system, designed for general purpose and not for specialized needs (i.e., not real-time, safety-critical or high-availability systems). The OS is acting both as hardware resource manager and as an execution platform for applications. We have used Microsoft Windows XP as a case study (its OS-driver interface, to be more specific), but porting our approach to other operating systems is part of our ongoing work.

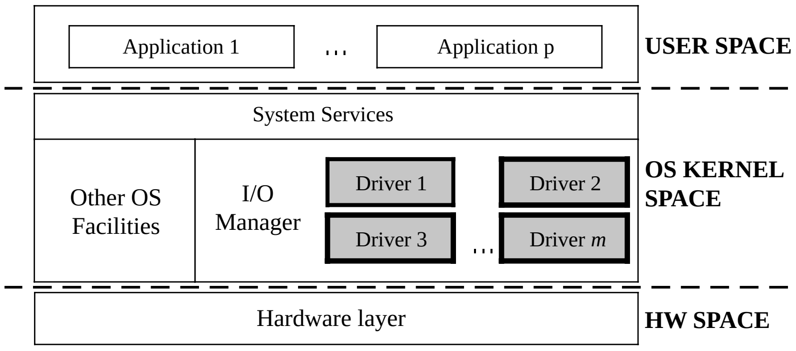


Figure 1. A system equipped with a COTS OS.

Our model is divided into three layers: (a) User space, (b) OS kernel space and (c) Hardware space (see Figure 1). The two dashed lines represent communication interfaces between two neighboring layers of the considered system. For instance, an application wants to access a file stored on the local hard drive. The request will be passed to the OS by means of standardized calling methods defined in the *System Services* layer. Further on, the command is handled by the *I/O Manager*, which transform it into a request to the *device driver* responsible for managing the hard drive. In most COTS OSs available today the drivers are implemented as software running in a privileged address space, i.e., kernel space. They act as a buffer between hardware layer and the rest of the OS kernel. Consequently, testing drivers properly is difficult. If something goes wrong and the driver is not able to properly handle the exceptional situation, the error can propagate to sensitive parts of OS and may cause instability or crash the entire system, reducing system's ability to provide the specified level of service.

[2] This work has been supported, in part, by EC FP6 IP DECOS, NoE ReSIST and also by Microsoft Research

2.1. Windows Driver Model

To standardize and simplify the OS   driver interface inside the kernel space, Microsoft defined a unified communication framework, called *WDM (Windows Driver Model)*[3]. A WDM driver have to comply with certain rules for design, implementation, initialization, plug-and-play, power management, memory access etc. We are particularly interested in the way that I/O Manager communicates with the drivers, which we called *WDM communication interface.*

A driver (as defined in the WDM model) can be assimilated to a toolbox. It contains a set of highly specialized procedures, each one being executed upon receipt of a specific request. Every time such a request is sent to the driver, it will execute the associated procedure.

These commands are issued by the I/O Manager (see Figure 1). They are called *I/O Request Packets* (IRPs) and are OS kernel structures which contain a request code and all the necessary parameters needed by the driver to service the particular IRP request. The driver receives the IRP request and, following its processing, a result of the operation is returned to the caller using the same IRP structure.

2.2. Driver Modes and Mode Graph

From a robustness testing perspective (for instance SWIFI testing) it is important to accurately pinpoint what function the system under test is executing at any moment in time. Therefore, the software testing community is actively researching methods to define the *state* of such a system. In the case of static software systems this task is relatively simple but for operating systems this is non-trivial. OSs are complex software, very dynamic and often entail a low level of determinism.

Assuming a black-box driver-under-test, our work [4] provided state identification methods only with regard to the functionality currently in execution (i.e., as observed from the OS   driver communication interface).

The *driver mode* is defined at a specific instance in time, with respect to the IRP requests the driver is executing at that moment, as a tuple of boolean values, each assigned to one of the *n* IRPs that the driver supports:

$$< P_{IRP1}, P_{IRP2}, \ldots, P_{IRPn} >$$

where P_{IRPi} (1   *i*   *n*) shows the current driver activity with respect to *IRPi*, as below:

P_{IRPi}	**The driver-under-test is**
1	performing the functionality associated with *IRPi*;
0	otherwise.

Knowing *n* we can build the *mode graph*, i.e., the complete set of tuples that represent all the different modes the driver can be in. Since modes are represented by binary strings, there are 2^n distinct modes in the graph (Figure 2). Transitions between modes are triggered either by receipt of an IRP or by finishing the execution of the functionality associated with an IRP.

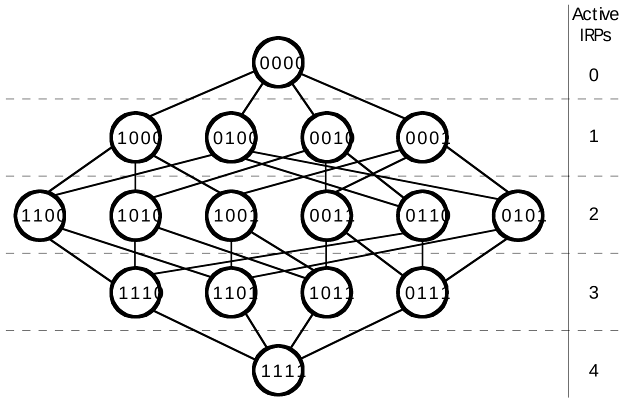


Figure 2. Driver supporting four distinct IRPs.

We assume that the driver receives and finishes the execution related to IRP requests in a sequential manner, i.e., only one bit is flipped at a time. Therefore, a driver can switch only to a mode whose binary string is within Hamming distance of 1 from the current mode.

In [4] we proposed three metrics for testing coverage of the mode graph of the driver-under-test: (a) *Mode Coverage*, (b) *Transition Coverage* and (c) *Path Coverage*. For a complete test coverage of our operational model, a testing process should satisfy all of them at the same time, but the measures are also useful as an indication of the testing progress.

3. Experiments, Results and Issues

To validate the presented approach, we conducted experiments to monitor the operational behavior of the serial driver provided with Windows XP Pro SP2.

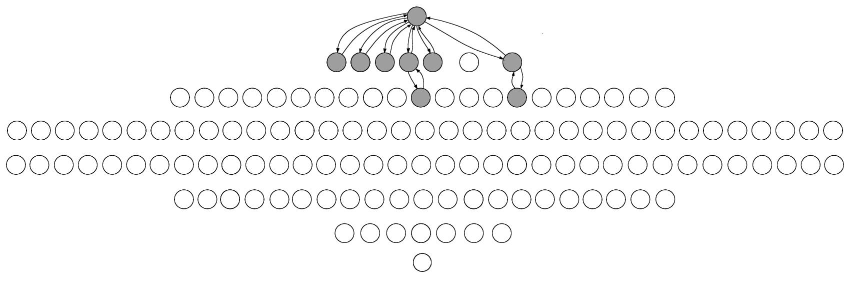


Figure 3. Visited modes and traversed edges.

Small visited mode space size: Using a relevant workload for the serial driver (commercial benchmark applications), we observed that only a very small part of the driver's mode graph is actually visited in the operational phase (Figure 3, shaded part in gray). Some FI testing methods [1, 6] inject errors only on the first occurrence of an event, therefore being limited to few of the modes. Our experiments revealed the modes and transitions having a high likelihood to be reached in the field, indicating the driver functionality that should primarily be targeted by the testing method of choice.

Wave testing: Generating the IRP sequences necessary to put the driver in a mode of interest can be difficult (i.e., for deeper modes), as the receipt and return of an IRP are events occurring in a non-deterministic manner. Therefore, we suggest using the following technique for testing:

1. test all the visited modes, accessing the same traversed edges;
2. identify modes accessible via one-hop from modes tested at step 1;
3. repeat steps above until no visitable modes.

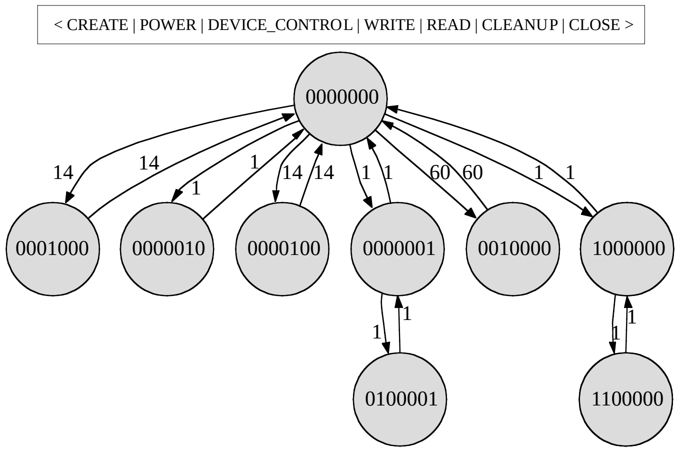


Figure 4. A detail of Figure 3, with respective IRPs.

Focused testing: Figure 4 contains only the details of the visited modes from Figure 3. We counted the number of times each transition was traversed, and observed that some of the modes were preferentially visited under the given workload. A superficial interpretation of this result might be to focus the testing strategy onto these highly accessed modes. Though, the less visited modes should not be ignored by testers, since they are associated with initialization / cleanup of the driver. Therefore, we identified the need to further develop our method to include a semantical ranking within distinct IRPs (and implicitly, within driver modes).

4. Discussion and Future Work

Our contribution provides means that precisely quantifies the test progress of already existing testing methods for COTS drivers. The experimental results are important, indicating relevant locations to focus the testing strategy (i.e., put high priority on testing modes and transitions visited in the field). Additionally, this methodology can be used for application profiling, for instance certain applications can be monitored to build behavioral patterns of driver usage (useful for security attack detection, debugging or dependability assessment).

Next, we will try to monitor the tools provided by Microsoft as driver robustness tests (dc2.exe, included in the WDM package). DC2 stress the target driver by sending billions of IRPs to it, with and without malformed parameters. We intend to identify the modes which are left un-visited by the test and assess their impact on driver robustness.

References

[1] A. Johansson and N. Suri. Error Propagation Profiling of Operating Systems. *International Conference on Dependable Systems and Networks*, 2005.

[2] K.-H. Möller and D. Paulish. An Empirical Investigation of Software Fault Distribution. *International Software Metrics Symposium*, 1993.

[3] W. Oney. *Programming the MS Windows Driver Model*. Microsoft Press, Redmond, Washington, 2003.

[4] C. Sârbu, A. Johansson, F. Fraikin, and N. Suri. Improving Robustness Testing of COTS OS Extensions. *International Service Availability Symposium*, 2006.

[5] M. M. Swift, B. N. Bershad, and H. M. Levy. Improving the Reliability of Commodity Operating Systems. *ACM Trans. Comput. Syst.*, 2005.

[6] T. Tsai and N. Singh. Reliability Testing of Applications on Windows NT. In *International Conference on Dependable Systems and Networks*, 2000.

[7] E. J.Weyuker. Using Operational Distributions to Judge Testing Progress. In *ACM Symposium on Applied Computing*, 2003.

On Exploiting Symmetry To Verify Distributed Protocols [*]

Marco Serafini
DEEDS Group
Department of Computer Science
Technische Universitat Darmstadt (TUDA), Germany
marco@deeds.informatik.tu-darmstadt.de

Péter Bokor
FTSRG Group and DEEDS Group (at TUDA)
Department of Measurement and Information Systems
Budapest University of Technology and Economics
petbokor@mit.bme.hu

Introduction

Amongst varied V&V approaches, formal verification is seeing increased usage and acceptance. One popular techniques is *model checking* that allows for *automated* verification (without user guidance) by performing exhaustive simulation on the model of the system. However, model checking faces the problem of *state space explosion*. Our ongoing research aims at proposing a method to facilitate the model checking of a class of distributed protocols by reducing the size of the associated state space.

Abstraction is a general method to reduce the level of detail in order to reduce the complexity of analysis. We present an abstraction scheme which is aimed at verifying *core fault tolerant protocols of frame-based, synchronous systems* (e.g consensus, atomic broadcast, diagnosis, membership). The idea is that, instead of modeling every single node explicitly, we represent only one correct node (we term this the *one-correct-node abstraction*) that captures overall symmetry as feasible. The rationale for our abstraction is that the protocols under analysis often entail symmetry in their assumptions (A1) *synchronous* and *symmetric communication* among correct nodes, that (A2) every correct node executes the same program, (A3) a *quorum* of correct nodes, stated by the fault assumption, is always present in the system. Furthermore, we are interested in verifying (A4) *non-uniform properties*, i.e., no restrictions are required on the internal state of faulty processors.

Note that even when (A1)-(A3) hold, the internal state of correct processors may differ, e.g. due to asymmetric broadcast of a faulty node. To capture this *asymmetry* we assign *non-deterministic* values to the possibly affected

[*] This research is supported in part by Microsoft Research, FP6 IP DECOS and NoE ReSIST

variables. In spite of that, the amount of correct information symmetrically exchanged by the quorum of correct nodes must be sufficient to guarantee a coordinated behavior of the quorum itself.

Section 1 presents a case study to demonstrate the idea of our abstraction. For any abstraction it is necessary to prove that the abstraction function is sound and complete, i.e., it is *property preserving*. In general the abstraction is said to be property preserving if for any valid formula f^A the following holds: M^A f^A M f, where $M(f)$ and M^A (f^A) stand for the concrete and abstract model (formula), respectively. Section 2 elaborates how to prove the soundness of our abstraction scheme, while Section 3 describes our future work.

1. Interactive Consistency - A Case Study

The problem of distributing data consistently in the presence of faults is variously called interactive consistency, consensus, atomic broadcast, or Byzantine agreement. When a node transmits a value to several receivers, we mandate that the properties of *agreement* (all non-faulty receivers adopt the same value, even if the transmitting node is faulty) and *validity* (if the transmitter is non-faulty, then non-faulty receivers adopt the value actually sent) hold. In this example we consider an interactive consistency protocol consisting of many parallel instances of the classical binary consensus [Lamport et al. 1982] where the maximum number of byzantine faults is *m = 1*. The protocol proceeds through of the following steps: *Step 1* every processor *i* acts as a sender and broadcasts its value v_i, *Step 2* every processor *j* receives and re-broadcasts v_{ij} to every other node (v_{ij} is the value of processor *i* as received by *j*), *Step 3* all nodes receive and adopt the majority value among v_{ij}, j $[1,...,i$ $1, i+1,...,n]$, if such a value exists, otherwise an a priori known default value is adopted.

Table 1 depicts the internal state of a correct node after executing the first two steps of this protocol. In this example the system contains *n = 4 > 3m* nodes and node 3 is byzantine. In the table, v_{ij} refers to the value sent by processor *j* in *Step 1* as received by *i* and re-broadcasted in *Step 2*. The fact that nodes do not re-broadcast their own value in *Step 2* is expressed by v_{ii} = ”   ”. Without loss of generality we consider the case where correct nodes send 0 as their own value. Since node 3 is faulty it may send different values to the other nodes in *Step 1*. Furthermore, in *Step 2* the values received from the other nodes can be rebroadcasted arbitrary. This asymmetry is expressed by the values in {1, 0} in column 3 and row 3 respectively. The majority voting of Step 3 is executed across each column I to obtain the values to adopt for v_i.

Our abstraction scheme models all correct nodes using only *one correct node*. Faulty nodes are not modeled as agreement and validity are non-uniform properties. To model the presence of asymmetric faults we add nondeterminism to the internal state of the one-correct-node: both column 3 and row 3 are assigned non-deterministic values, though with a different semantic. While the 3rd column is consistent for every correct node due to (A1), row 3 may differ from node to node (even if they are correct), as the faulty node can re-broadcast different values to different nodes in *Step 2*.

	Senders			
Re-broadcaster	**1**	**2**	**3**	**4**
1	-	0	1/0	0
2	0	-	1/0	0
3	1/0	1/0	-	1/0
4	0	0	1/0	-

Table 1 - Interactive Consistency: internal state

In general we need to show that the abstract properties hold. Intuitively, for $agreement^A$, we need show that for any arbitrary, non-deterministic value the majority voting gives the same outcome, while $validity^A$ requires that if node *i* is correct (in our case nodes 1, 2 and 4) the value adopted by the one-correct-node is indeed that value sent by node *i* (0 in the example). Table 1 directly shows both properties to hold. For agreement, column 3 is consistent for every correct node, and majority voting over it will result in a consistent outcome. For validity, the fault assumption of [Lamport et al. 1982] ($n > 3m$) guarantees a sufficient quorum of correct values to outvote the non-deterministic values, which become irrelevant. To prove that the abstraction is sound we have to show that $agreement^A \quad agreement$ and $validity^A \quad validity$ hold.

2. Symmetry Reduction

In general it might be cumbersome to prove the soundness of the abstraction for *any* valid formula. We aim at establishing a general abstraction scheme, which poses no restriction to the formula of interest, by proving that our approach is a case of *symmetry reduction* [Clarke et al. 1993]. This would ensure that our one-correct-node abstraction is sound and complete. In fact, symmetry reduction, states that if the state transition graph exhibits defined morphisms, then property preserving abstractions (expressed in form of permutation sets) can be defined.

To assess the applicability of the abstraction two aspects have to be considered. First, we are interested in the *gain* of the approach, i.e. to which extent the abstraction reduces the state space. This also involves experiments comparing the computational complexity of the concrete and abstracted model checking problem run by specific model checkers. Initial measurements performed using the SRI-SAL model checker [de Moura 2004] on a diagnostic protocol [Walter et al. 1977] have shown a considerable reduction.

The other aspect is to define the *class of protocols* for which the abstraction is applicable. So far we considered *synchronous communication* among nodes and we successfully applied the approach to membership and diagnosis algorithms, all of them as specific cases of the *consensus* problem. We would like to extend our approach also to *approximate agreement* protocols, used for clock synchronization, and consensus in *asynchronous* systems and hybrids. Note that, in general, while symmetry is ensured by the synchrony of the frame-based communication scheme, our abstraction, as it is currently defined, can not be directly applied to asynchronous protocols, where correct nodes can also receive different subsets of the values sent by the other correct nodes.

3. Conclusion & Future Work

For distributed protocols, we propose a symmetry based abstraction to decrease complexity of model checking by reducing the size of the state space. Our ongoing research addresses three aspects: (a) soundness of the abstraction by proving that it reduces to a special case of symmetry reduction [Clarke et al. 1993]; (b) explore boundaries of applicability of the approach; (c) quantify effectiveness of the abstraction using different model checking techniques (e.g. symbolic and bounded model checking).

References

[Clarke et al. 1993] Edmund M. Clarke, Thomas Filkorn, Somesh Jha, “Exploiting symmetry in temporal logic model checking”, In CAV ’93: Proceedings of the 5th International Conference on Computer Aided Verification, pages 450–462, London, UK, 1993, Springer-Verlag

[de Moura 2004] Leonardo de Moura, Sam Owre, Harald Rueß, John M. Rushby, Natarajan Shankar, Maria Sorea, Ashish Tiwari: “SAL 2”, CAV 2004: 496-500, 2004

[Lamport et al. 1982] L. Lamport, R. Shostak, M. Pease, “The byzantine generals problem”, ACM Transactions on Programming Languages and Systems, 4(3), 1982

[Walter et al. 1977] Chris J. Walter, P. Lincoln, N. Suri, “Formally verified on-line diagnosis”, IEEE Transactions on Software Engineering, 1997

Session on Diversity

Chair: Ilir Gashi, CSR, City University, UK

Potential for Dependability Gains with Diverse Off-The-Shelf Components: a Study with SQL Database Servers

Ilir Gashi

Centre for Software Reliability, City University London

Email: I.Gashi@city.ac.uk

Introduction

Fault tolerance is often the only viable way of obtaining the required system dependability from systems built out of “off-the-shelf” (OTS) products. In particular, modular redundancy with diversity may be an affordable solution, especially since the individual components may be available at little or no cost. Little empirical evidence however exists about the practical dependability benefits of employing diverse redundancy with OTS components. We have conducted two studies with sample of bug reports from six off-the-shelf SQL servers (from four different vendors) so as to estimate the possible advantages of software fault tolerance - in the form of modular redundancy with diversity - in complex off-the-shelf software. We found that very few bugs cause coincident failures in more than one server. None of the bugs in our study caused failures in more than two different-vendor servers. Therefore, a fault-tolerant server, built with diverse off-the-shelf servers, seems to have a good chance of delivering improvements in availability and failure rates compared with the individual off-the-shelf servers or their replicated, non-diverse configurations.

1. Motivation

When dependability improvements are sought for systems built out of off-the-shelf components fault tolerance is often the only possible way of obtaining them. Various design solutions exists for implementing fault tolerance ranging from simple error detection and recovery techniques [1] to replication with diverse versions of the components, i.e. “diverse modular redundancy”. These solutions are well known in the literature. The question remains however about the actual dependability improvements that can be expected from employing them.

To study these issues for a complex category of off-the-shelf components, we have chosen SQL database servers. Developing an SQL server using design diversity (e.g. several of-the-shelf SQL servers and suitably adapted “middleware” for replication management) requires strong evidence of dependability benefits it can yield: for example empirical evidence that likely failures of the SQL servers, which may lead to serious consequences, are unlikely to be tolerated without diversity. We investigated such empirical evidence. We sought to demonstrate whether design diversity has a potential to deliver significant improvement of dependability of SQL servers, compared to solutions for data replication that can only tolerate crash failures. The only direct dependability evidence that is available for the SQL servers are their fault reports. Therefore a preliminary evaluation step concerns *fault* diversity rather than *failure* diversity. By manual selection of test cases, one can check whether the diverse redundant configuration would tolerate the known bugs in the repositories of bugs reported for the various OTS servers. To this end I conducted two studies with SQL servers from four different vendors, both commercial and open-source. I collected known bug reports for these servers. For each bug, I took the test case that would trigger it and ran it on all four servers (if possible), to check for coincident failures. I found the number of coincident failures to be very low. Results of the first study have been reported here [2]. A paper is in preparation for the results of the second study and I will give a preview of the results observed.

2. Results

Two commercial (Oracle 8.0.5 and Microsoft SQL Server 7 (without any service packs applied)) and two open-source (PostgreSQL 7.0.0 and Interbase 6.0) SQL servers were used in the first study. Interbase, Oracle and MSSQL were all run on the Windows 2000 Professional operating system, whereas PostgreSQL 7.0.0 (which is not available for Windows) was run on RedHat Linux 6.0 (Hedwig). The study was conducted as follows. The known bugs for the OTS servers are documented in bug report repositories (i.e. bug databases, mailing lists etc). Each *bug report* contains the description of what the bug is and the *bug script* (SQL code that contains the failure triggering conditions) required to reproduce the *failure* (the erroneous output that the reporter of the bug observed). In our study I collected these bug reports and *ran* the bug scripts in the servers (we use the phrase “*running a bug*” for the sake of brevity). I collected a total of 181 bug reports. The full results were reported here [2]. The main findings were:

- very few bugs affect two of the four servers, and none affect more than two. Moreover only four of these bugs would cause identical, undetectable failures in two servers. Fault-tolerant, diverse servers therefore seem to have a good chance of improving failure rates and availability.
- it may be worthwhile for vendors to test their servers using the known bug reports for other servers. For example, we observed 4 MSSQL bugs that had not been reported in the MSSQL service packs (previous to our observation period). Oracle was the only server that never failed when running on it the reported bugs of the other servers;
- the majority of bugs reported, for all servers, led to “incorrect result” failures (64.5%) rather than crashes (17.1%) (despite crashes being more obvious to the user). This is contrary to the common assumption that

the majority of bugs lead to an engine crash, and warrants more attention by users to fault-tolerant solutions, and by designers of fault-tolerant solutions to tolerating subtle and non fail-silent failures.

The results of the first study were very intriguing and pointed to potential for serious dependability gains from using diverse off-the-shelf SQL servers. However these results concern only a specific snapshot in the evolution of these products. We therefore decided to repeat the study for later releases of these servers. I collected 92 new bug reports for the later releases of the open-source servers: PostgreSQL 7.2 and Firebird 1.0[1]. We decided not to collect any bug reports for the closed-development servers as the reproduction scripts needed to trigger the fault were missing in most of them (but I still ran these bug scripts on the two commercial servers). We are currently preparing a paper to detail the findings (the full raw data can be found here [3]). The main findings include:

- we again found very few bugs that caused failures in two of the servers and none that caused failures in more than two. From the bugs collected in the second study, only one caused identical non-self evident failures on more than two different-vendor servers and this was only on a few demands. The percentage of crash failures was again observed to be significantly lower (19%) then "incorrect result" failures (65.5%). This further confirmed our observations from the first study.
- I also ran the bugs reported for the new releases on the servers in the older releases of those servers, and vice versa. We observed that for PostgreSQL the gains are quite significant. Most of the old bugs had been fixed in the new version and moreover a large proportion of the newly reported bugs did not cause failure (or could not be run at all) in the old version. This would suggest that a limited amount of dependability improvements can be gained by at least running different releases of a server from the same vendor in a fault-tolerant configuration.

We also studied the mechanism of "data diversity" [4] and its application with SQL servers in aiding with failure diagnosis and state recovery. We defined 14 generic rephrasing rules to be implemented in a 'rephrasing' algorithm which when applied to particular SQL statements will generate logically equivalent statements. We applied these rules to the bugs reported for the open-source servers and found that the rephrased statements were 'workarounds' for a significant number of these bugs.

3. Ongoing and Further Work

The results are intriguing and point to a potential for serious dependability gains from assembling a fault tolerant server from two or more of these off-the-shelf servers. But they are not definitive evidence. Apart from the sampling difficulties caused, e.g. due to a lack of certain bug scripts, it is important to clarify to what extent our observations allow us to predict such gains. We have provided a discussion here [2] of the extent to which we can extrapolate from the count of common bugs to reliability of a diverse server taking into account the various complications that arise due to:

- the difference between fault records and failure records
- imperfect failure reporting

[1] Firebird is the open-source descendant of Interbase 6.0. The later releases of Interbase are issued as closed-development by Borland.

- variety of usage profiles.

Further work in this direction is the extent to which the existing reliability growth models (e.g. the Littlewood model [5]) can be utilised for reliability predictions of a 1-out-of-2 system built out of off-the-shelf components using the observations we have. The work on this has already started.

References

1. Popov, P., et al. Protective Wrapping of OTS Components in 4th ICSE Workshop on Component-Based Software Engineering: Component Certification and System Prediction, 2001, Toronto.

2. Gashi, I., Popov, P., Strigini, L. Fault diversity among off-the-shelf SQL database servers in DSN'04 International Conference on Dependable Systems and Networks, 2004, Florence, Italy, IEEE Computer Society Press, p. 389-398.

3. Gashi, I., Tables containing known bug scripts of Firebird 1.0 and PostgreSQL 7.2. 2005 http://www.csr.city.ac.uk/people/ilir.gashi/Bugs/.

4. Ammann, P.E. and J.C. Knight, Data Diversity: An Approach to Software Fault Tolerance, IEEE Transactions on Computers, 1988, C-37(4), p. 418-425.

5. Littlewood, B., Stochastic Reliability Growth: a Model for Fault-Removal in Computer Programs and Hardware Designs, IEEE Transactions on Reliability, 1981, R-30(4), p. 313-320.

Improvement of DBMS Performance through Diverse Redundancy

Vladimir Stankovic

Centre for Software Reliability, City University, Northampton Square, London EC1V 0HB, UK

http://www.csr.city.ac.uk

V.Stankovic@city.ac.uk

1. Introduction

The most important non-functional requirements for a Database Management System (DBMS) are performance and dependability. Often a trade-off between the two is required to satisfy opposing priorities set for the system. Data replication has proved to be a viable method to enhance dependability and performance of DBMSs. The common assumption that fail-safe failures, i.e. crashes, are the foremost problem to be solved in order to improve system dependability has been refuted in [1]. Hence the diverse redundancy would deliver improvements in failure rates and availability compared with non-diverse redundancy or individual server configurations. A possible architecture for a fault-tolerant server employing (diverse) redundancy is depicted in

Figure **1**.

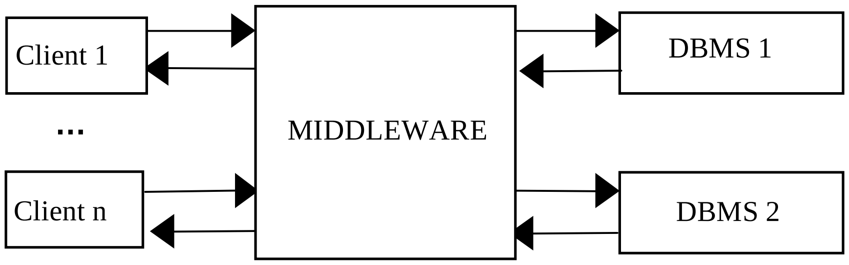


Figure 1. Fault-tolerant server node (FT-node) with two (possibly more) diverse DBMSs. The middleware "hides" the servers from the clients (1 to n) for which the data storage appears as a single DBMS

Can one also obtain performance gains? The response time of SQL (Structured Query Language) statements might vary among diverse servers. For example, the respective execution plans will be different or concurrency control mechanisms will vary. Also systematic differences between the execution times of different transactions might exist when diverse servers are employed. The differences could lead to improved performance. We are particularly concerned with a regime of operation of the FT-node that will favour performance – *optimistic* regime of operation. For this regime the only function of the middleware is to translate the client requests, send them to the servers and as soon as the first response is received, return it back to the client (see Figure 2). If the faster response comes from different servers depending on the SQL statement, then the optimistic regime will give a faster service than the faster of the two servers (provided the overhead of the middleware is not too high compared with the response times of the servers). This happens if the sum of the transport delay to deliver the fastest response to the client, the client's own processing time to produce the next SQL statement, and the transport delay to deliver the next SQL statement to the middleware is longer than the extra time needed by the slower server to complete SQL statement processing (the red dashed rectangle indicates that DBMS 2 would not be ready to start the (n+1)th SQL statement at the same time with DBMS 1). In this case, both (or all) servers will be ready to take the next SQL statement and the race between them will start over. In more dependable set-up a different type of regime can be envisaged. The middleware validates the results of executing each SQL statement. It waits for responses from both servers, checks if the two responses are identical and, in case they differ, initiates recovery.

Data consistency in database replication is usually defined in terms of 1-copy serialisablity between the transactions' histories executed on multiple nodes [2]. On transaction boundaries, the middleware running in *optimistic* regime waits for the slower server in order to preserve data consistency. This approach can be combined with other *eager* approaches that combine non-diverse (homogenous) database replication and group communication primitives in order to satisfy both fault-tolerance and performance requirements [3].

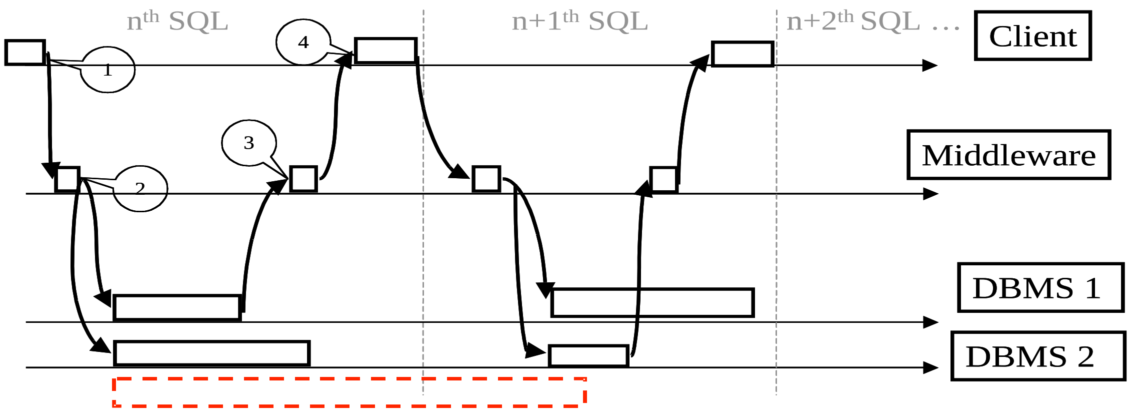


Figure 2 Timing diagram of a client communicating with two, possibly diverse, database servers and middleware running in optimistic regime. The meaning of the arrows is: 1 – the client sends an SQL statement to the middleware, 2 – the middleware translates the request to the dialects of the servers and sends the resulting SQL statements, or sequences of SQL statements, to the respective servers; 3 – the fastest response is received by the middleware; 4 - the middleware sends the response to the client

We introduced a *skip* feature to the middleware to further improve performance. The feature enables the slower server to omit the already executed (by faster server) read (SELECT) statements. The modifying SQL statements (DELETE, INSERT and UPDATE) are executed on all servers. The *skip* feature helps slower server to advance faster during a transaction execution and the overhead introduced by the replication decreases.

2. Experimental Setup

In the empirical study we used an implementation of an industry-standard benchmark for online transaction processing, TPC-C [4], to evaluate potential for performance improvement. TPC-C defines five types of transactions: *New-Order*, *Payment*, *Order-Status*, *Delivery* and *Stock-Level* and sets the probability of execution of each. The minimum probability of execution for New-Order and Payment transactions is 43%, while it is 3% for the other three types. The test harness consisted of three machines. The client machine executed the TPC-C standard implemented in JAVA and the two server machines ran diverse open-source DBMSs, namely InterBase

6.0 and PostgreSQL 7.4.0 (abbreviated to IB and PG in the remainder of the document for brevity). The client and the servers communicate through Java Database Connectivity (JDBC).

To get indications of how the FT-node compares with the single servers and non-diverse server combinations we performed different types of experiments: 1IB, 1PG, 1IB1PG, 2IB and 2PG when only a single client is employed. At the same time we wanted to produce more realistic load on servers but preserve data consistency. Therefore we introduced additional clients (10 or 50) that execute read-only transactions. All clients communicated to the same TPC-C database. We took measurements only for the TPC-C compliant client.

3. Experimental Results

We used the following three measures to compare different server configurations: mean response time per transaction of each type, mean response time for all five transaction types and cumulative transaction time for all five transaction types – experiment duration.

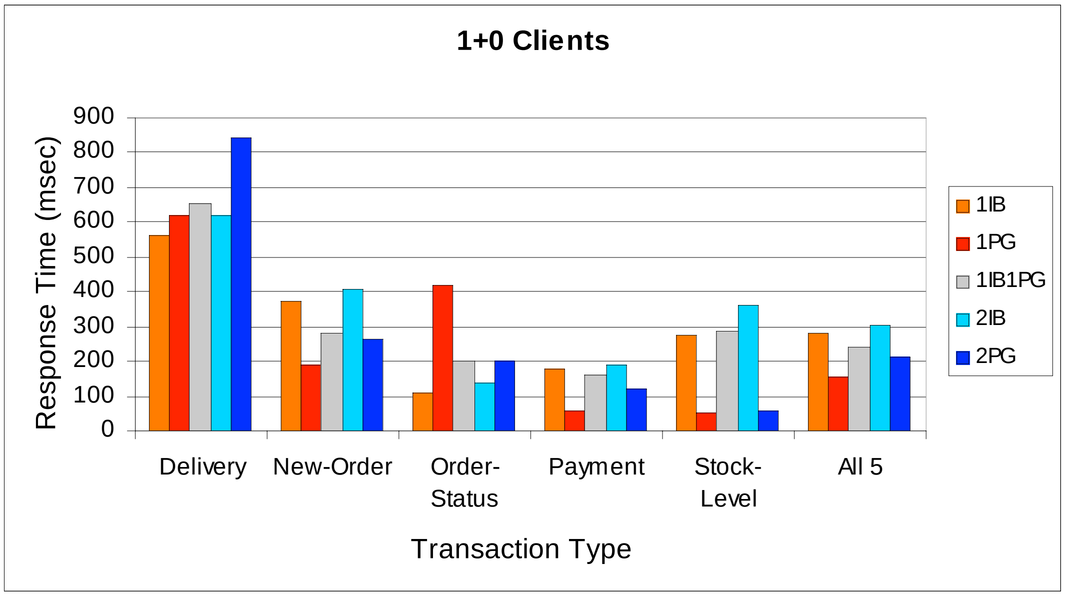


Figure 3 Mean response times of single server combinations (IB, PG), diverse server pair (1IB1PG) or homogenous server pairs (2IB, 2PG) for each transaction type and for all transactions together under a single client load

Figure **3** depicts the response time when only one client communicates with different server configurations. We see that PG is on average the best combination under this load, though transactions of type Delivery and Order-Status are faster on IB. The situation changes when we look at the experiment with the increased load (see Figure 4). Now the fastest combination on average is the diverse pair, albeit not for all transaction types. The figure indicates that the diverse servers “complement” each other in the sense that when IB is slow to process one type of transaction PG is fast (New-Order and Stock-Level) and vice versa (Payment, Order-Status and Delivery). The similar results were obtained under the load with 50 additional clients.

The performance improvement of the diverse pair is more telling when we looked at the cumulative transaction times for each server configuration. The experiment of 10,000 transactions under the ‘lightest’ load (0 additional clients) is the fastest with 1PG. Under increased load, however, the experiment is fastest with the diverse pair, 1IB1PG. The diverse pair is 20% faster than the second best combination (1PG) when 10 additional clients are deployed and more than 25% faster than the second best combination (2PG) when 50 additional clients are deployed.

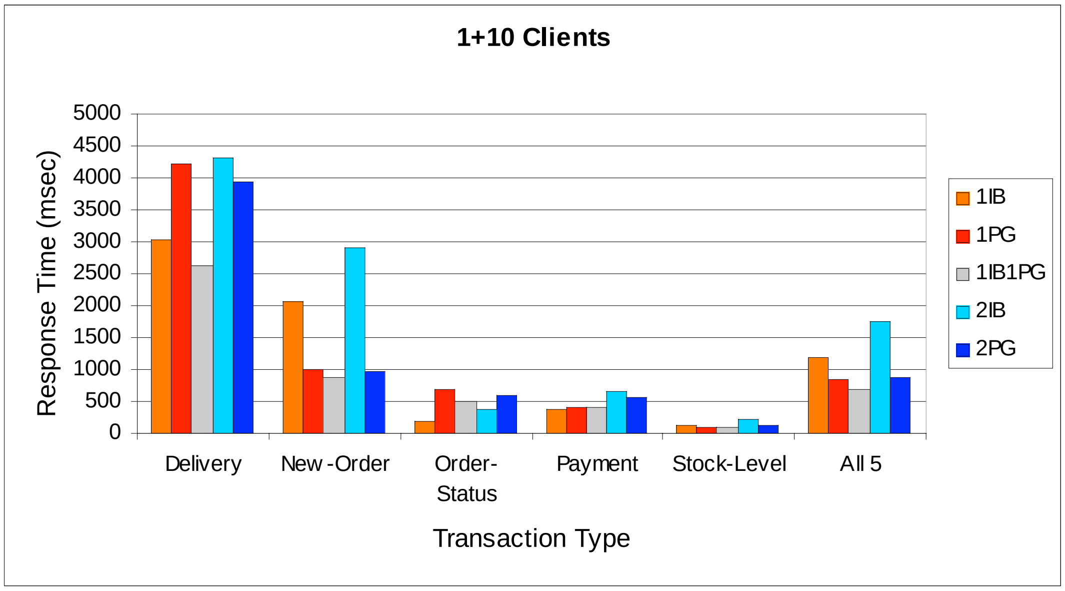


Figure 4 Mean response times of single server combinations (IB, PG), diverse server pair (1IB1PG) and homogenous server pairs (2IB, 2PG) for each transaction type and for all transactions together under an increased load with 10 additional clients

4. Conclusions and Future Work

The results presented substantial performance gain when diverse redundancy is used. Although a synthetic load (TPC-C) was used, the same should be true for a real load, especially in a more read-intensive *environment*. To evaluate performance of a read-intensive application another type of benchmark might be used, such as TPC-H.

The ongoing research includes implementation of an algorithm that ensures data consistency among replicated databases. A promising direction for future development is to implement a configurable middleware, deployable on diverse DBMSs, which will allow the clients to request quality of service in line with their specific requirements for performance and dependability.

5. Bibliography

1. Gashi, I., P. Popov, and L. Strigini, *Fault Diversity among Off-the-shelf SQL Database Servers*, in *International Conference on Dependable Systems and Networks (DSN'04)*, 2004, Florence, Italy, IEEE Computer Society Press, p. 389-398.

2. Bernstain, A., V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*. 1987, Reading, Mass.: Addison-Wesley.

3. Patino-Martinez, M., et al. *Scalable Replication in Database Clusters*. in *International Conference on Distributed Computing, DISC'00*. 2000: Springer.

4. TPC, *TPC Benchmark C, Standard Specification, Version 5.0*. 2002, Transaction Processing Performance Consortium.

Session on Mobile Systems

Chair: Ludovic Courtès, LAAS-CNRS, France

Storage Mechanisms for Collaborative Backup of Mobile Devices

[Extended Abstract]

Ludovic Courtès ludovic.courtes@laas.fr

LAAS-CNRS

7 avenue du Colonel Roche

31077 Toulouse CEDEX 4

France

Introduction

Mobile devices are increasingly relied on but are used in contexts that put them at risk of physical damage, loss or theft. Yet, due to physical constraints and usage patterns, fault tolerance mechanisms are rarely available for such devices. We consider that this gap can be filled by exploiting spontaneous interactions to implement a collaborative backup service. The targeted mobile devices (PDAs, "smart phones", laptops) typically have short-range wireless communication capabilities: IEEE 802.11, Bluetooth, ZigBee. We also assume that those devices also have intermittent Internet access.

The service we envision, which we call MoSAIC[1], aims at providing mobile devices with mechanisms to tolerate hardware or software faults, notably permanent faults such as theft, loss, or physical damage. In order to tolerate permanent faults, our service must provide mechanisms to store the user's data on alternate storage devices using the available communication means.

Considering that wireless-capable mobile devices are becoming ubiquitous, we believe that a collaborative backup service could leverage the resources available in a device's neighborhood. Devices are free to participate in the service. They can benefit from it by storing data on other devices. Conversely, they are expected contribute to it, by dedicating some storage resources, as well as energy.

1 *Mobile System Availability, Integrity and Confidentiality*, *http://www.laas.fr/mosaic/*. MoSAIC is partly financed by the French national program for Security and Informatics (ACI S&I). Our partners are IRISA (Rennes) and Eurécom (Sophia-Antipolis).

This approach is similar to that taken by the commonly used wired peer-to-peer services. It offers a cheap alternative to the costly infrastructure-based paradigms (e.g., UMTS, GPRS), both financially and in terms of energy requirements. Additionally, it may operate in circumstances under which infrastructure access is not available, potentially improving the backup service continuousness. On the other hand, the outcome of using this service will be highly dependent on the frequency of participant encounters.

The next section presents MoSAIC's goals, notably in terms of dependability. New issues raised by this form of service are also highlighted. Section 3 presents our work on the design of storage mechanisms for collaborative backup based on the requirements imposed by the mobile environment. Finally, Section 4 summarizes our findings and sketches future research tracks.

1. Goals and Issues

In this section, we describe the issues that we are trying to solve with this backup service in terms of dependability. Then we describe the functionalities of the service. Finally, we detail new challenges that need to be addressed in this cooperative approach.

2.1. Fault Tolerance for Mobile Devices

In the contexts they are used, mobile devices are subject to permanent faults (e.g., loss, damage, theft), as well as transient faults (e.g., accidental erasure of data, transient software fault). These faults can lead to the loss of data stored on the device. Yet, those devices are increasingly used to produce or capture new data in situations where only infrequent backups are feasible. The development of a collaborative backup service is motivated by the need to tolerate these faults.

The primary goal of our service is to improve the availability of the data stored on mobile devices. Each device should be able to store part of its data on neighboring mobile devices in an opportunistic fashion, using the available short-range wireless communication means. Here the device fulfills the role of data owner, taking advantage of the backup service. Conversely, each participating device must dedicate some of its resources to the service, notably storage, so that others can benefit from the service as well the device acts as a contributor.

We assume that as soon as a contributor gains access to the Internet, it can take advantage of it to transmit the collected data to their data owner. Data owners are then able to restore their data if need be. In practice, transferring data via Internet can be realized in a number of ways: using e-mail, a central FTP server, or some sort of a peer-to-peer durable data store, for instance. However, our work is not concerned with this part of the service. Instead, we focus on the design of the backup service in the mostly-disconnected case.

Our approach is somewhat similar to that of the widely used peer-to-peer file sharing [1] and backup systems [5] on the Internet. However, the mobile environment raises novel issues.

2.2. Challenges

We assume that participants in the backup service have no a priori trust relationships. Therefore, the possibility of malicious participating devices, trying to harm individual users or the service as a whole, must be taken into account.

Obviously, a malicious contributor storing data on behalf of some user could try to break the data confidentiality. Contributors could as well try to modify the data stored. Storage mechanisms used in the backup service must address this, as will be discussed in Section 3.

A number of denial of service (DoS) attacks can be envisioned. A straightforward DoS attack is data retention: a contributor either refuses to send data back to their owner when requested or simply claims to store them without actually doing so. DoS attacks targeting the system as a whole include flooding (i.e., purposefully exhausting storage resources) and selfishness (i.e., using the service while refusing to contribute). These are well-known attacks in Internet-based peer-to-peer systems [1,5,6] which are only partly addressed in the framework of ad hoc routing in mobile networks [3]. One interesting difference in defending against DoS attacks in packet routing compared to cooperative backup is that, in the former case, observation of whether the service is delivered is almost instantaneous while, in the latter, observation needs to be performed in the longer-run. Addressing these issues is a challenge.

Other challenges that need to be solved include the design of an appropriate storage layer which we will discuss in the next section.

2. Storage Mechanisms

In this section we focus on issues raised by the design of appropriate storage mechanisms for the backup service. First, we identify issues raised by the mobile environment and the resulting constraints imposed on the storage mechanisms. Then we present solutions drawn from the literature that are suitable to our context.

3.1. Requirements

The mobile environment raises several specific issues. First, resources on mobile devices (energy, storage space, CPU power) are scarce. Thus, the envisioned storage mechanisms need to be efficient.

In particular, to limit the impact of running the backup service on energy consumption, care must be taken to use wireless communication as little as possible given that it drains an important part of a mobile device's energy [14]. Consequently, the amount of data to transfer in order to realize backups needs to be kept as low as possible. Logically, this goal is compatible with that of reducing the amount of data that needs to be stored.

Encounters with participating devices are unpredictable and potentially short-lived. Thus, users cannot expect to be able to transfer large amounts of data. At the storage layer, this leads to the obligation to fragment data. Inevitably, data fragments will be disseminated among several contributors. However, the global data store consisting of all the contributors' local stores must be kept consistent, in the sense of the ACID properties of a transactional database.

Also, ciphering and error detection techniques (both for accidental and malicious errors) must be integrated to guarantee the confidentiality and integrity of the data backed up. To maximize the chances that a malicious data modification is detected, cryptographic hashes (such as the SHA family of algorithms, Whirlpool, etc.) need to be used, rather than codes primarily design to detect accidental errors such as CRCs.

Finally, given the uncertainty yielded by the lack of trust among participants and the very loosely connected scheme, backups themselves need to be replicated. How replicates should be encoded and how they should be disseminated in order to get the best tradeoff between storage efficiency and data availability must be determined.

3.2. Design Options

The storage-related concerns we described above lie at the crossroads of different research domains, namely: distributed peer-to-peer backup [5,9], peer-to-peer file sharing [7], archival [12], and revision control [10,13]. Of course, each of these domains has its own primary criteria but all of them tend to share similar techniques. File sharing, for instance, uses data fragmentation to ease data dissemination and replication across the network. The other domains insist on data compression, notably inter-version compression, in order to optimize storage usage.

Study of the literature in these areas has allowed us to identify algorithms and techniques valuable in our context. These include fragmentation algorithms, data compression techniques, fragment naming schemes, and maintenance of suitable meta-data describing how an input stream may be recovered from fragments.

We implemented some of these algorithms and evaluated them in different configurations. Our main criteria were storage efficiency and computational cost. We performed this evaluation using various classes of data types that we considered representative of what may be stored on typical mobile devices. The results are available in [4].

As mentioned earlier, backup replication must be done as efficiently as possible. Commonly, erasure codes [16] have been used as a means to optimize storage efficiency for a desired level of data redundancy. Roughly, erasure codes produce n distinct symbols from a k-symbol input, with $n > k$; any $k + e$ symbols out of the n output symbols suffice to recover the original data (is a non-negative integer that depends on the particular code chosen). Therefore, $n \quad (k + e)$ erasures can be tolerated, while the effective storage usage is $\frac{k + e}{n}$. A family of erasure codes, namely rateless codes, can potentially produce an infinity of output symbols while guaranteeing (probabilistically) that still only $k + e$ output symbols suffice [11].

However, an issue with erasure codes is that whether or not they improve overall data availability is highly dependent on the availability of each component storing an output symbol. In a RAID environment where the availability of individual drives is relatively high, erasure codes are beneficial. However, in a loosely connected scenario, such as a peer-to-peer file sharing system, where peers are only available sporadically at best, erasure codes can instead hinder data availability [2,8,15]. Therefore, we need to assess the suitability of erasure codes for our application.

Furthermore, data dissemination algorithms need to be evaluated. Indeed, several data fragment dissemination policies can be imagined. In order to cope with the uncertainty of contributor encounters, users may decide to transfer as much data as possible to each contributor encountered. On the other hand, users privileging confidentiality over availability could decide to never give all the constituent fragments of a file to a single contributor.

We are currently in the process of analyzing these tradeoffs in dissemination and erasure coding using stochastic models. This should allow us to better understand their impact on data availability.

4. Conclusion

We have presented issues relevant to the design of a collaborative backup service for mobile devices, including specific challenges that need to be tackled. In particular, we showed how the constraints imposed by a dynamic mobile environment map to the storage layer of such a service.

Our current work deals with the analytical evaluation of the data availability offered by such a backup service. Our goal is to evaluate the data availability as a function of the rates of participant encounters, Internet access, device failure, and time. This evaluation would allow the assessment of different data dissemination policies. Useful modeling tools to achieve this include Markov chains and generalized stochastic Petri nets.

References

[1] K. BENNETT, C. GROTHOFF, T. HOROZOV, I. PATRASCU. Efficient Sharing of Encrypted Data. *Proc. of the 7th Australasian Conference on Information Security and Privacy* (ACISP 2002), (2384), pages 107120, 2002.

[2] R. BHAGWAN, K. TATI, Y-C. CHENG, S. SAVAGE, G. M. VOELKER. Total Recall: System Support for Automated Availability Management. *Proc. of the ACM/USENIX Symp. on Networked Systems Design and Implementation,* 2004.

[3] L. BUTTYÁN, J-P. HUBAUX. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. ACM/Kluwer Mobile Networks and Applications, 8(5), October 2003.

[4] L. COURTÈS, M-O. KILLIJIAN, D. POWELL. Storage Tradeoffs in a Collaborative Backup Service for Mobile Devices. LAAS Report #05673, LAAS-CNRS, December 2005.

[5] L. P. COX, B. D. NOBLE. Pastiche: Making Backup Cheap and Easy. *Fifth USENIX OSDI,* pages 285-298, 2002.

[6] S. ELNIKETY, M. LILLIBRIDGE, M. BURROWS. Peer-to-peer Cooperative Backup System. The USENIX FAST, 2002.

[7] D. KÜGLER. An Analysis of GNUnet and the Implications for Anonymous, Censorship-Resistant Networks. *Proc. of the Conference on Privacy Enhancing Technologies*, pages 161176, 2003.

[8] W. K. LIN, D. M. CHIU, Y. B. LEE. Erasure Code Replication Revisited. *Proc. of the Fourth P2P*, pages 9097, 2004.

[9] B. T. LOO, A. LAMARCA, G. BORRIELLO. Peer-To-Peer Backup for Personal Area Networks. IRS-TR-02-015, UC Berkeley; Intel Seattle Research (USA), May 2003.

[10] T. LORD. The GNU Arch Distributed Revision Control System. 2005, *http://www.gnu.org/software/gnu-arch/*.

[11] M. MITZENMACHER. Digital Fountains: A Survey and Look Forward. *Proc. of the IEEE Information Theory Workshop*, pages 271276, 2004.

[12] S. QUINLAN, S. DORWARD. Venti: A New Approach to Archival Storage. *Proc. of the First USENIX FAST*, pages 89101, 2002.

[13] D. S. SANTRY, M. J. FEELEY, N. C. HUTCHINSON, A. C. VEITCH, R. W. CARTON, J. OFIR. Deciding when to forget in the Elephant file system. *Proc. of the 17th ACM SOSP*, pages 110123, 1999.

[14] M. STEMM, P. GAUTHIER, D. HARADA, R. H. KATZ. Reducing Power Consumption of Network Interfaces in Hand-Held Devices. *IEEE Transactions on Communications*, E80-B(8), August 1997, pages 11251131.

[15] A. VERNOIS, G. UTARD. Data Durability in Peer to Peer Storage Systems. *Proc. of the 4th Workshop on Global and Peer to Peer Computing*, pages 9097, 2004.

[16] L. XU. Hydra: A Platform for Survivable and Secure Data Storage Systems. *Proc. of the ACM Workshop on Storage Security and Survivability*, pages 108114, 2005.

Towards Error Recovery in Multi-Agent Systems

Alexei Iliasov

Alexei.Iliasov@newcastle.ac.uk

Introduction

The mobile agent paradigm is an attractive approach to developing large-scale, distributed applications. Its flexibility and scalability makes it the technology of choice for highly dynamic and open environments. It also benefits from the advances in wireless networks and wearable computing. Developers, however, still face a number of challenges designing and implementing large agent systems. One of them is the lack of an effective application-level recovery mechanism. This paper summarises these challenges and presents a criticism of an existing exception handling based recovery mechanism and the plans for its adaptation and integration with thr formal agent development methodology.

1. Challenges to Providing Error Recovery in Agent Systems

1.1. Decentralisation and homogeneity

In classical distributed systems a program is designed to be executed concurrently over a number of computing nodes for the benefits of improved performance and better use of resource. Multi-agent systems are composed of a number of independent computing nodes. Concurrency and distribution come more as necessity rather than a design decision. The key distinction is that components of a distributed systems are *orchestrated*, explicitly, by a dedicated entity, or implicitly, through an implemented algorithm, in order to solve a common task while agents *collaborate* to achieve their individual goals. The common approach to error recovery in distributed systems is a hierarchical organisation of processes where recovery is attempted at different levels and exceptions are propagated across the levels. No such organisation is possible for agent systems. Agent are not linked by any relations and have the same privileges, rights and capabilities. Since each agent has only a partial knowledge of a global system state and acquires knowledge about other agent states through communication, collaboration between agents is the only way to jointly recover from an error.

1.2. Weak Communication Mechanisms

Agent systems commonly employ communication mechanisms which provide very weak, if any, delivery and ordering guarantees. This is important from the implementation point as agent systems are often deployed on wearable computing platforms with limited processing power and use unreliable wireless networks for communication. One of the most successful approaches to tolerating connectivity disruptions is the use of decoupled communication. Publisher/Subscriber, Linda and message queues are the most prominent representatives of asynchronous communication languages. Publisher/Subscriber provides a name-space decoupled communication where event producers do not have to know names or number of consumers. Communication in Linda is both time and name-space decoupled. Linda tuples are anonymously deposited in a tuple space from where they can be anonymously retrieved later. Message queue is an example of a time decoupled mechanism. Message producers and consumer have to know the name of a message queue but the do not have to write and read at the same time. Neither of these mechanisms provides a delivery notification or any form of delivery guarantee. This makes it hard to tell between a crash of an agent and a delay in a message delivery. It can be dangerous to assume an agent failure solely on the basis of a timeout. On the other hand, extensive use of timeouts can be highly impractical. Thus a recovery mechanism for should not attempt to make a distinction between network failures and agent crashes unless there is a support for this from a communication mechanism.

1.3. Autonomy

During its live an agent has to communicate with a large number of other agents, developed in decentralised manner by independent developers. This is very different from the situation in classical distributed system where all the system components are part of a closed system and thus fully trusted. Each agent participating in a multi-agent application tries to achieve its own goal. Agent goals, in a general case, may be conflicting. For example, the goal of a seller agent is to sell at the highest possible price while a buyer must buy at the lowest price. For a recovery mechanism this means that no single agent should be given an advantage which may affect outcome of a multi-agent application. Any scenarios where an agent controls or prescribes a recovery process for another agent must be avoided.

1.4. Anonymity

Most agent systems employ anonymous communication where agents do not have to disclose their names to other agents. This has a number of benefits: agents do not have to learn names prior to communication, there is no need to create fresh names and ensure naming consistency in presence of migration and it is easy to implement group communication. Anonymity is also an important security feature - no one can sense an agent presence until it producers a message or an event. It also makes harder to tell which messages are produced by which agent. For a recovery mechanism, anonymity means inability to explicitly address agents which must be involved into a recovery. It may be impossible event to discover the number of agents that must be recovered. Though it may be rather straightforward to implement names exchange, the impact on agent security and problems of names consistency usually outweights the benefits of directed messaging. Thus, a recovery mechanism preferably should not rely on a knowledge about agent names and must be scalable in respect to a number of agents involved into a recovery.

1.5. Message Context

In sequential systems recovery actions are attached to certain regions, objects or classes which define a context for a recovery procedure. There is no obvious counterpart for these structuring units in asynchronously communicating agents. Agent produces messages in a certain order, each being a result of some calculations. When data sent along with a message cause an exception in an agent, the agent may want to notify the original message producer, for example, by sending an exception. When the exception arrives to the message producer it could happen that the agent has proceeded with other calculations and the context in which the message was produced is destroyed. This cannot happen in the case of synchronous communication where a message producer waits for a reaction of a consumer. The problem of a lost context is a major obstacle on the way to implementing application-specific recovery. This is why many existing systems use only asynchronous recovery.

1.5. Message Semantics

In distributed systems developed in a centralised manner semantics of values passed between system components is fixed at the time of the system design and implementation. In open agent system implementation is decentralised and thus the message semantics must be defined at the stage of a multi-agent application design. If agents can also send exceptions the list of exceptions and their semantics must be also defined at the level of an abstract application model. For a recovery mechanism this means that each agent has to deal only with the exceptions it was designed to recover from. New exceptions and new message types cannot appear dynamically during an agent lifetime unless there is way to dynamically extend agent functionality.

2. Exception Propagation in CAMA

This sections presents analysis of an exception handling based error recovery mechanism implemented in the CAMA framework [2, 4]. CAMA introduced a number of concepts which heavily influenced the design of the recovery mechanism. The most prominent ones are scope and role [6]. Scope is a unit of communication isolation. Messages produced in a scope are inaccessible outside of the scope. Agent role is the typing mechanism which prevents communication of incompatible agents. Each scope supports a predefined set of roles and only agents implementing one of these roles can participate in the scope. These rules are supported by the agent development methodology based on the B method [5]. The mechanism has an experimental implementation for the Lime mobile agents middleware and a more mature version for the CAMA framework. The essence of the mechanism is propagation of exceptions among agents which allows agents to implement a form of cooperative recovery. The recovery mechanism is closely integrated with the CAMA scoping mechanism which primary role in recovery is error confinement. When a failure is detected, the isolation provided by a scope guarantees that the agents possibly affected by the failure are only those participating in the current scope. A scope is also a basic structuring unit for a recovery in a multi-agent application. A scope is recovered when all the scope participants recover from a failure. The nesting of scopes introduces a nested hierarchy of recovery units. The mechanism analysis is based on the discussion of how the mechanism addresses each of the challenges presented above. The detailed discussion of the mechanism can be found in [7].

Decentralisation and homogeneity. The mechanism relies on the middleware to control exception propagation among agents. The control is done on the basis of a set of rules jointly produced by interested agents. The problem of ensuring consistency of this rules is can be an obstacle for a decentralised implementation.
Weak Communication Mechanisms. Exceptions are introduced as new type of messages. Implementation-wise this does not introduce dramatic changes to the communication layer however it requires priority of exceptions over normal messages.
Autonomy. Though agents recover cooperatively they do not give up their autonomy during recovery. An agent is never forced to expose its state or to follow commands of another agent.
Anonymity. This is one of the strong sides of the mechanism. An agent name is never disclosed to other agents during a recovery.
Message Context. The mechanism attempts to address the problem of exception context by configuring virtual

regions of recovery. Before leaving a region an agent ensures there are no pending exception associated with this region. When the region is closed all the incoming exceptions associated with it are mapped into a predefined exception. In addition, there are primitives to poll for pending exceptions and wait for exception to appear. They allow programmers to implement their custom recovery strategies, such as asynchronous recovery. *Message Semantics*. The mechanism itself does not address this problem. Instead there is a formal top-down development procedure based on the B Method. Development starts with an abstract model of a multi-agent application. More details are added in a step-wise manner using the refinement technique. The final result is a decomposed specification of a scope along with independent specifications of agent roles. It is possible to formally demonstrate their interoperability for roles constructed in such manner. Since all the constant values and message types were originally defined in an abstract specification they have exactly the same semantics in all the individual roles.

3. Future Plans

The exception propagation mechanism discussed above has a number of weak points. It does not deal very well with decentralisation and message context. Being a rather complex mechanism it is hard to formalise and incorporate into the formal top-down development methodology. The attempt to cover many possible recovery scenarios by providing low-level primitives makes scope-level recovery implementation a hard and an error-prone process. All of these calls for a different style of recovery based on the same principles. The mechanism must be simple enough to be formalised along with agent roles and flexible enough to cover the most important scenarios of application level recovery. The methodology is based on the reactive architecture [3] which fits well the formal development approach of action systems. It also seems to be a reasonable way for introducing recovery actions so that a role specification is a set of reactions implementing normal behaviour and reactions for recovery actions. The open question though is how to enter and leave abnormal code. This problem stems from the fact that internal agent architecture if far more complex than those of a sequential program or a standard concurrent program. Firstly, exceptions may appear both internally and externally. Secondly, the reactive architecture permits an unlimited number of of interleaving activities which are more or less unaware of each other.

4. References

[1] Rigorous Open Development Environment for Complex Systems. IST FP6 STREP project, online at http://rodin.cs.ncl.ac.uk/

[2] A. B.Arief and A.Romanovsky. On Using the CAMA Framework for Developing Open Mobile Fault Tolerant Agent Systems. University of Newcastle. 2006.

[3] N. Busi, A. I. T. Rowstron, and G. Zavattaro. State- and event-based reactive programming in shared dataspaces. In COORDINATION '02: Proceedings of the 5th International Conference on Coordination Models and Languages, pages 111124, London, UK, 2002. Springer-Verlag.

[4] A. Iliasov. Implementation of Cama Middleware. Available online at http://sourceforge.net/projects/cama [Last accessed: 1 June 2006].

[5] A. Iliasov, L. Laibinis, A. Romanovsky, and E. Troubitsyna. Towards Formal Development of Mobile Location-based Systems. Presented at REFT 2005 Workshop on Rigorous Engineering of Fault-Tolerant Systems, Newcastle Upon Tyne, UK (http://rodin.cs.ncl.ac.uk/events.htm), June 2005.

[6] A. Iliasov and A. Romanovsky. CAMA: Structured Coordination Space and Exception Propagation Mechanism for Mobile Agents. Presented at ECOOP 2005 Workshop on Exception Handling in Object Oriented Systems: Developing Systems that Handle Exceptions. July 25, 2005. Glasgow, UK, 2005.
[7] A. Iliasov and A. Romanovsky. Exception Handling in Coordination-based Mobile Environments. In Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC 2005), pages 341350. IEEE Computer Society Press, 2005.

Increasing Data Resilience of Mobile Devices with a Collaborative Backup Service

Damien Martin-Guillerez
IRISA/ENS-Cachan
Campus de Beaulieu, 35 042 Rennes Cedex, FRANCE
dmartin@irisa.fr

1. Introduction

The production of sensible data on mobile devices, such as PDAs or mobile phones, has increased with the use of such devices. The loss of those data can have painful consequences for the user (e.g. loss of phone numbers or meeting notes).

To reduce data loss, many devices have a synchronization mechanism. The main issue in synchronization is the necessary proximity of the user and his computer and thus, there is a time period during which device failure means irreversible data loss. For example, if you take a note on your PDA during a meeting and this PDA gets lost on your way home then the note is definitely lost. However, more and more mobile devices come with wireless connectivity like IEEE 802.11. Thus, using neighbor devices to save data right after its production can decrease data loss. Data can be restored either from a global-scale network like the Internet or directly from the backup device.

We aim at creating a transparent collaborative backup service for mobile devices [5]. Such a service needs to meet specific requirements outlined in section 2. Then, we analyze several specific issues of mobile device data and replication in section 3. Afterwards, we present a way to order replicas in that system in section 4. After presenting existing systems in section 5, we outline works that are still pending and conclude in section 6.

2. Design overview

Our main aim is to design a transparent backup system that can handle high mobility. Thus, it needs to handle two scenarios: (i) when connected to a global network like the Internet, the system must use the opportunity to save data on a resilient server and (ii) when disconnected from the global network, it must use neighbor terminals to backup selected data.

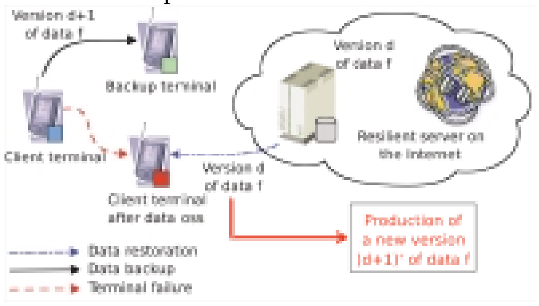


Figure 1: Conflict during a restoration.

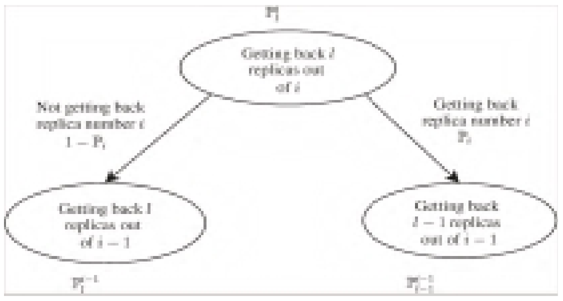

P_l^i
Getting back l replicas out of i
Not getting back replica number i
$1 - P_i$
Getting back replica number i
P_i
Getting back l replicas out of $i-1$
Getting back $l-1$ replicas out of $i-1$
P_l^{i-1}
P_{l-1}^{i-1}

Figure 2: Graphical proof of equation (1).

Depending on data production (e.g. production rate, data importance), the system should adapt the level of replication. Moreover, the system needs to be protected against egoistic participants that backup but do not provide resources to others. Of course, we want the system to be as transparent for the user as possible. That means that very few actions are required from the user during the backup or the restore and that neither prior trust relationship with other peers nor extra hardware is required.

We consider only following backup scenarios: a client terminal can either backup its data to another terminal or to an Internet server. Later, the backup peer can save the data on the Internet server. If a failure occurs, the client terminal can restore the data from the peer or from the server. We do not consider propagating backup through peers because:

- Propagating backups costs energy and others resources that will not be available for further backups.
- Only a full replica can be issued in such situations, contrary to considered backup dispersion schemes.
- Only the client terminal can know when it's necessary to issue a replication. A replication issued by a backup terminal has a good chance to be useless.

3. Data Issues

Mobile device data. We will now look at data produced on classical mobile devices and at their attributes to outline related issues. The size mainly depends on data type (from less than 200 bytes for SMS to hundred of megabytes for movies). The second attribute is data importance, high for notes taken during a meeting to very low for holiday pictures for instance. Dependencies are also important: a SMS may be useless without all preceding SMS in a discussion thread. When a data item depends on preceding data, we call it a *temporal dependency*. On the contrary, when a data item is interdepent with others data, we call it a *spacial dependency*. So, mobile device data are categorized by size, dependencies and importance.

The size affects the backup system in means of (i) resisting to mobility or network problems during a transmission and (ii) avoiding monopolizing one terminal memory. Dependencies affect backup integrity and thus the system needs some version tracking. Finally, we assign a priority for each data item relatively to its importance and try to save data items with highest priority first.

Dispersion of replicas. File fragmentation is imposed by potentially high sized data. Moreover, the risk of a terminal not correctly restoring a replica creates a need for a flexible replication scheme. Courtes et al. [4] have already defined the redundancy and compression methods we use. We consider that all data items with spatial dependencies are agglomerated into one (the priority of the new item is the highest of the agglomerated items). Then, we consider the (n, k) replication scheme that fragments the data item into n pieces where only k are required for reconstruction. We also consider delta-compression which saves only the changes between two versions of the same file. Replication when delta-compressing is made on generated delta.

So, we consider that every data item to save follow this format: n fragments and only k needed to reconstruct the data item, possible temporal dependencies (the priority of old data should be increased if it is lower than the priority of the new data).

Version tracking. Given those propagation and dissemination schemes, some issues can appear regarding arrival of new version of a data item to backup. First, the old version of a data should be kept until all dependencies of the new version have been backed-up to the resilient server. Moreover, conflicts may appear in our system. When you backup the data of a mobile device on a fixed station, no conflict appears since all new versions of a data item are created on this device. But, with our propagation scheme, a conflict may appear (cf. figure 1) when a data item is backed-up on another mobile device and an old version of this data item is located on the Internet server. If a failure occurs in this case, the client may restore the old version from the server and work on it, generating a conflict with the version backed-up on the mobile device. In such a situation, our system must use conflict resolution mechanisms like in Bayou [9].

4. Maximizing backup reliability

We will start studying the *(n, k)* replication scheme. Let P_i be the probability of getting back the replica *i* and P_i^l the probability of being able to get back *l* replicas between the first *i* ones. Then we have (cf. figure 2):

$$P_i^l = (1 + P_i)\ \ P_{i\ 1}^l + P_i\ \ P_{i\ 1}^{l\ 1}$$

So, when backing-up an additional replica, we can estimate the impact on the probability of getting back the entire data item. That is right, of course, only if each replica is saved on a different terminal. We can handle the case of two replicas being backed-up on the same terminal by assuming that they have the same probability P_i. Thus, if we save *m* replicas onto the same terminal at the same time, we have:

$$P_{i+m}^l = (1 \quad P_{i+1})\ \ P_{i\ 1}^l + P_{i+1}\ \ P_i^{l\ m}$$

We assume that a further save of a replica on an already used terminal is an independent event. Finally we must take into account the temporal dependencies. The probability of restoring correctly a new data item that depends on old ones is the probability of restoring the new data item multiplied by the probability of restoring the old data.

We said in section 3 that each data item is associated with a priority given as a desired backup resilience (e.g. a probability). A prior mechanism should have established this priority. Hence we can order data items to be backed-up using a queue ordered by the priority minus the computed probability of successful backup.

Thus, we get a general algorithm to order data packets to save. First, we try to save while the peer is reachable. The first data item that may be saved on this peer is pulled off the queue. We try to save the next packet of this item and recompute the probability of a successful backup. If the probability is too low, the data item is inserted back into the queue. We use (1) and (2) to recompute the probability.

5. Related Works

Usually, data resilience is increased using hardware replication. In network file systems, replication of data uses several data servers [8]. Peer-to-peer file systems have used replication to increase data availability [3] and have paved the way for collaborative backups [1].

In a mobile context, Roam [7] uses peers to replicate data for high availability but can hardly handle high mobility due to the clusterization of the peers. Data replication between clusters is needed when a peer switches clusters. In fact, Roam is not designed for backup recovery but for high availability and data sharing. Moreover, Roam does not exploit opportunistic replication on random mobile peer. AdHocFS [2], another file system focused on high availability and data sharing by transposing peer-to-peer file systems' paradigms to ad hoc networks, does not give support for high mobility. FlashBack [6], a backup system for Personal Area Network (PAN), can efficiently handle data loss. FlashBack is designed for people that wear several wireless mobile devices. Thus, FlashBack suffers from the same limitations as AdHocFS and Roam.

6. Conclusion

We have presented a general design for a backup system that can handle high mobility and that do not rely on pre-established relationships. Issues regarding incentives, confidentiality, high mobility and resource management are still to be resolved.

Indeed, several requirements have been outlined in section 2: the system must be transparent, should not rely on prior relationships nor on specific hardware. The system must automatically assign the priority to the data, should use incentives and confidentiality techniques. The network layer should use classical wireless interface without interference with their classical uses.

A main pending issue is the probability estimation of one packet to be correctly restored (P_i). The main parameter is the reliability of the device itself eventually given by incentives. Other parameters can be battery lifetime, terminal context and memory availability. Resource management implies deletion of replicas to free memory on backup terminals. We need to know which replicas to delete and the impact on the system efficiency. Our future works will concentrate on those issues related to resource management.

References

[1] C. Batten, K. Barr, A. Saraf, and S. Treptin. pStore: A Secure Peer-to-peer Backup System. Technical Report MIT-LCS-TM-632, MIT Laboratory for Computer Science, Dec. 2001.

[2] M. Boulkenafed and V. Issarny. AdHocFS: Sharing Files in WLANS. In *The Second IEEE International Symposium on Network Computing and Applications (NCA'03)*, Apr. 2003.

[3] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *The Workshop on Design Issues in Anonymity and Unobservability*, pages 4666, July 2000.

[4] L. Courtès, M.-O. Killijian, and D. Powell. Storage Tradeoffs in a Collaborative Backup Service for Mobile Devices. To appear, 2006.

[5] M.-O. Kilijian, D. Powell, M. Banâtre, P. Couderc, and Y. Roudier. Collaborative Backup for Dependable Mobile Applications. In *The 2nd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (Middleware)*. ACM, Oct. 2004.

[6] B. T. Loo, A. LaMarca, and G. Borriello. Peer-To-Peer Backup for Personnal Area Networks. Technical Report IRS-TR-02-015, Intel Research Seattle - University of California at Berkeley, May 2003.

[7] D. Ratner, P. Reiher, and G. J. Pope1. Roam: A Scalable Replication System for Mobile Computing. In *The Workshop on Mobile Databases and Distributed Systems (MDDS)*, pages 96104, Sept. 1999.

[8] M. Satyanarayanan. Scalable, Secure and Highly Available Distributed File Access. *IEEE Computer*, 23(5):921, May 1990.

[9] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In *The 15th ACM Symposium on Operating Systems Principles (SOSP'95)*, Dec. 1995.

Random Walk for Service Discovery in Mobile Ad hoc Networks

Adnan Noor MIAN
Dipartimento di Informatica e Sistemistica
University of Rome, “La Sapienza”
Via Salaria 113 Roma, Italy
adnan@dis.uniroma1.it

1. Introduction

Service discovery in mobile ad hoc networks (MANETs) is a challenging issue. As nodes in a MANET offer spontaneous and variable connectivity, the proximity of a given service as well as the number of services vary unpredictably with time. Traditional directory based architectural solutions can hardly cope with such a dynamic environment while a directory-less approach has to resort to network-wide searches. Moreover deploying a service discovery protocol on top of a routing protocol is another source of inefficiency, since similar problems have to be addressed twice by both protocols. In this work we are trying to tackle the problem of service discovery by leveraging on the random walk based search. Here from service discovery we mean finding the location of a service.

2. Related Work

There has been a good amount of work in service discovery regarding the wired networks but the problem has not been addressed very successfully in MANETs. In wired networks four types of architectures have emerged. In the first type, directory-based architecture, some nodes with better computation and memory resources are selected as service coordinators (SC). The service coordinators advertise themselves to other nodes. Service provider nodes register with these SCs. Clients contact these SCs to get the location of service providers. Examples include Jini [1], UDDI [11] and Salutation [12]. This approach is not suitable for MANETs where the topology of the system keeps on changing and it is not easy to form service coordinator node. In the second type, which is the directory-less architecture, there is no service coordinator. Clients contact service provider directly by flooding the service query. This results in a high overhead produced due to flooding. This approach is also not suitable for MANETs as flooding consumes lot of scarce bandwidth resource. Examples include SLP [3] and UPnP [2]. Third type is the hybrid architecture in which servers may either register their services with SCs (if they are available) or wait for the client service query. Client may send a service query to SCs (if they are available) or directly to service providers using flooding. This architecture is also not suitable for MANETs for the reasons mentioned in the previous two architectures. In the forth type of architecture, service discovery can be intergraded with the route discovery as both exploit network-wide broadcasting. Such a cross-layer design principle can improve the performance but there is still the need to resort to network wide searches [13].

Some of the important service discovery protocols proposed for MANETs are following. Kozat and Tassiulas proposed a distributed service discovery architecture for MANETs that relies on a virtual backbone for locating and registering available services within a dynamic network topology [4]. The architecture consists of two independent parts: backbone management (BBM) phase and distributed service discovery (DSD). A similar approach has more recently proposed by Sailhan and Issarny [5] for implementing a scalable directory service for MANET. The architecture is based on homogenous and dynamic deployment of cooperating directories among the network, arranged in a virtual network. Despite the goal of the architecture as achieving scalability, the paper presents performance results only for a few nodes (90 nodes).

Konark [6] is designed specifically for the discovery and delivery of services in multi-hop ad hoc networks. It supports both push and pull modes, with a cache in all devices and uses multicast to advertise and discover services. The only analogy with our proposal is that every node maintains a service registry, where it stores information about its own services and also about services that other nodes provide.

A field theoretical approach to service discovery has been proposed by Lender et al. [7]. A service is modeled by a (positive) point charge, and service request packets are seen as (negative) test charges that are attracted by the service instances. They map the physical model to a mobile ad hoc network in a way where each network element calculates a potential value and routes service requests towards the neighbor with the highest potential, hence towards a service instance. In our approach the estimated proximity could also be thought as the potential of a field. However, our core forwarding protocol is random in nature.

The use of random walk as a mean of searching for an item is a well-established technique in unstructured peer-to-peer (P2P) networks [8]. The topology of a mobile ad hoc network is however structurally different from P2P networks and thus the suitability of random walks have to be studied carefully.

Finally, a technique similar to the one used to measure hints, has been originally proposed in the Associativity Based Routing (ABR) routing scheme, where a counter is used to count the number of "associativity ticks" between two nodes and use this information to select the most stable route [9].

Our approach is based on Random Walk for discovering or locating a service in mobile ad hoc network. This is a probabilistic approach. It does not need to have any centralized repository. It takes advantage of the mobility of the nodes and is also resilient to the topological changes. This work is explained in the next sections.

3. Exploiting Random Walk

Given a graph and a starting point, we select a neighbor of it at random and move to this neighbor. We then select a neighbor of this point at random and move to it and so on. The random sequence of points selected this way is a random walk on the graph. The theory of random walks on a graph has been explained in great detail in [10]. We are trying to exploit this theory for the purpose of service discovery in mobile ad hoc networks. In this approach a client (node that is interested in a particular service) will approach the service randomly moving on the graph formed by the MANET. The advantage of this approach is that there will be no flooding which consumes lot of scarce bandwidth resource in MANET and also the problem of collision of packets can be avoided.

In pure random walk on the graph [10] the neighbor is selected randomly, but we shall try to bias the selection process. In this regard there is an interesting result that we want to exploit. Let us consider a simple case of N+1 nodes arranged in one-dimensional form, as shown in Figure 1.

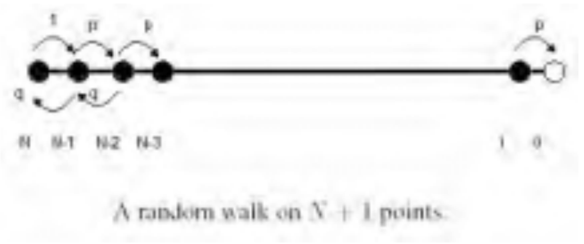


Figure 1

Suppose a node i wants to search a service present in node s. For this purpose, to forward a query message there are always two options available, that are either sending the query message towards the service provider node s or sending the query message away from it. Let the probability p of forwarding the query to the node near to service provider node s is 0.5. Then the probability q of selecting a node away from the node s is also 0.5. We can say that the selection of node for forwarding the query packet is completely unbiased. In this case the mean hitting time is N^2 where the hitting time is the expected number of steps or hops in a random walk before a node s is visited for the first time, starting from node I [10,14]. Let us take another case in which if each node selects a neighbor node towards the node s with probability p=1, that is, the selection is biased. Now there are exactly N hops. Thus we see that biasing the selection process decreases the number of hops of a query message to reach the required service.

For us this result is interesting in the sense that if in a service discovery protocol we can find some way to bias the selection of next neighbor such that the probability is $0.5<p<1$, we can then decrease the number of hops considerably for the discovery of a service. The problem remains to find some metric that can be used to bias the next hop selection for service query message.

It is clear that for next hop, the node that is nearest to the service provider and also in the wireless range of the client should be selected. The most obvious method in determining the nearest node to the service provider is using the Euclidean distance. We can then have a Euclidean metric for biasing the selection process. But this metric requires Global Positioning System (GPS) to determine the relative distances of the nodes from the service provider. GPS is not always possible.

We propose an alternative technique and a metric to bias the node selection process. Our technique is based on finding the radial speed of different nodes with respect to the service provider. Bias is provided by a metric called hint [15]. Hint h_{is} is calculated and stored by a node i for a particular service provider node s. It provides the proximity of node i to node s at time t. In other words, it tells that the node i has some time ago remained in contact with s. No hint will be available in case node i has never been in the vicinity of node s.

To calculate hint, node s sends an advertisement message after every T secs containing the description of the service. This advertisement message is picked up by a mobile neighbor node i, which then updates its service table. The service table at node i, in addition to other information also records the duration of the last wireless link established with s and time elapsed since the link with s was broken. The node i after every T secs calculates hint and while moving sends the hint to the newly encountered neighbors. The nodes that have low values of hints are more probable to be near to the node s.

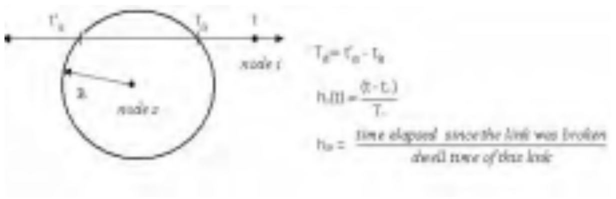


Figure 2

4. Algorithm for Random Walk based Service Discovery

Our Service Discovery Protocol based on biased random walk mainly consists of two phases. In the first phase, moving nodes when pass near service provider nodes calculate hints, as explained in Figure 2, and store these hints. In the second phase, when a client node searches for a service and sends a query message, a forwarding protocol running on each node forwards the query message when it receives it. These two phases are explained below.

Hint creation and distribution

1) A node s advertises a service S by sending an advertisement message containing the description of the service after every T secs.

2) The advertisement message is picked up by a neighbor node i, which then updates its service table. An entry of the table is deleted when it becomes stale.

3) The service table at node i contains the following information:

   i) description of the service S
   ii) ID of the advertising node
   iii) ID of the sending node
   iv) duration of the last wireless link established with s
   v) time elapsed since the link with s was broken

4) After every T secs node i calculates hint and while moving sends the hint to the newly encountered neighbors. The nodes, which have low values of hints, are more probable to be near to the service provider node.

Forwarding Protocol

1) A node wishing to discover a service S generates a request message containing the description of S.

2) If hints are available then that node is selected for next hop, which provides the minimum value of hint.

3) If hints are not available then the selection of the next hop node is at random.

In Figure 3, small values near the nodes are the hints. The query message starts from node s and following a random selection (shown as thin red arrow) and biased selection (shown as thick blue arrow) eventually reaches the node s.

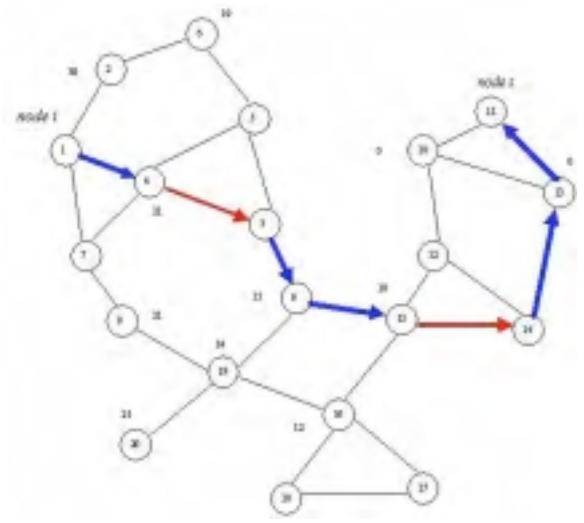


Figure 3

5. Future Work

We are trying to setup a simulation that will simulate our protocol. We are studying the problem from analytical point of view and trying to look for possible chances of utilizing the work done in the field of graph theory, particularly random walks on graphs. Also we are trying to find out other metrics that can be used to bias the selection process. We also plan to compare the message cost in our model with the flooding model. The model and the algorithm presented here assume the mobility of nodes. It will not work if the nodes are stationary. We want to study a model that caters for both a system with moving nodes and also stationary (or slow moving) nodes.

References

[1] Sun Microsystems. Jini network Technology. http://www.jini.org, 1999

[2] Microsoft Corporation. Universal Plug and Play (UPnP) forum. http://www.upnp.org., 1999

[3] PerkinsC., Veizades J. Day M. Guttman, E. Service location protocol, Version 2. RFC 2608, Internet Engineering Task Force (IETF), June 1999

[4] L. Tassiulas and U.C. Kozart. Service discovery in mobile ad hoc networks: an overall perspective on architectural choices and network layer support issues, 2004

[5] V. Issarny F. Sailhan. Scalable service discovery fro manet. *In Proceeding of the 3rd IEEE Int'l Conf. On Pervasive Computing and Communications (PerCom2005)*, Kauai Island, Hi, USA, 8-12 March 2005. IEEE.

[6] V. Verma C. Lee S Helal, N Desai. Konark, a service discovery and delivery protocol for ad hoc networks. *In the proceedings of the third IEEE Conference on Wireless communication Network (WCNC)*, New Orleans, USA, 2003, IEEE.

[7] Martin May Vincent Lenders and Bernhard Platter. Service discovery in mobile ad hoc networks: A field theoretic approach. *Elsevier Pervasive and Mobile Computing*, 2005

[8] Amin Saberi Christos Gkantsidis, Miena Mihail. *Random walks in peer-to-peer networks. In the Proceedings of INFOCOM 2004*, Hong Kong, March 7-11, 2004.IEEE

[9] C. K.G Toh. A novel distributed routing protocol to supported ad hoc mobile computing. *In IEEE 15th Annual International Phoenix Conference on Computers and Communications(IPCCC)*, Phoenix, AZ, USA, March 27-29, 1996, IEEE

[10] L. Lovasz, Random walks on graphs: a survey, In Combinatorics Paul Erdos in Eighty, Vol. 2, Budapest, 1993. J'anos Bolyai Mathematical Society

[11] OASIS Consortium The universal description, discovery and integration (UDDI). http://www.uddi.org.

[12] Salutation Consortium. Salutation architecture specification version 2.1. http://www.salutation.org, 1999.

[13] George C. Polyzos Christopher N. Ververidis. Routing layer support for service discovery in mobile ad hoc networks. In Proceedings of third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'05), Washington, DC, USA, 2005. IEEE.

[14] Gefen Y. Goldhirsch II. Biased random walk on networks. Physical review, Feb. 1987, 35(2).

[15] R. Baldoni R. Beraldi, L. Querzoni. A hint-based probabilistic protocol for unicast communications in manets. Elsevier Ad Hoc Networs, 2005.

Session on Distributed Systems

Chair: Paulo Sousa, University of Lisbon, Portugal

Quantitative Evaluation of Distributed Algorithms

Lorenzo Falai - Università di Firenze

Introduction

The quantitative evaluation of performance and of dependability-related attributes is an important activity of *fault forecasting* ([1]). Quantitative system assessment can be performed using several approaches, generally classified into three categories: *analytic*, *simulative* and *experimental*. Each of these approaches shows different peculiarities, which determine the suitableness of the method for the analysis of a specific system aspect. The most appropriate method for quantitative assessment depends upon the complexity of the system, the development stage of the system, the specific aspects to be studied, the attributes to be evaluated, the accuracy required, and the resources available for the study.

The largeness and complexity of dependability critical systems, together with the necessity of continuous verification and validation activities during all the design and development stages in order to promptly identify deviations from the requirements and critical bottleneck points, call for a composite V&V (verification and validation) framework, where the synergies and complementarities among several evaluation methods can be fruitfully exploited. Comparison of results for a certain indicator obtained through the application of two alternative methods allows *cross-validation* of both. Feeding a system model with parameter values derived through experimental measurement is a central example of *cross-fertilization* among different methods.

A high proliferation of automatic tools supporting a variety of quantitative evaluation methods has been reached till now, and research in this direction is always in progress. In recent years, special attention is being devoted to the analysis of distributed protocols.

In this stream of methodologies and tools to contribute to the V&V of distributed algorithms, a framework has been developed, called Neko, which consists of a simple communication platform that allows to both simulate a distributed algorithm and execute it on a real network, using the same implementation for the algorithm ([2]).

However, Neko permits only to collect *traces of execution*; it does not include support to collect and manage events so as to perform on-line quantitative evaluations, in parallel with the algorithm execution. We worked on the NekoStat extension to Neko; using this extension it is possible to perform simple and powerful quantitative analysis of distributed algorithms, using simulative and experimental approaches. Following the same philosophy of Neko, NekoStat has the ability to perform quantitative evaluations adopting both the simulative and experimental approaches. The main difference between these two kinds of analysis is that in simulations we can make on-line evaluations, whereas in real experiments the quantitative evaluation is performed only at the termination of the distributed system execution, after all the data have been collected.

1. The Neko Framework

Neko ([2]) is a simple but powerful framework that permits the definition and the analysis of distributed algorithms, showing the attracting feature that the same Neko-based implementation of an algorithm can be used for both simulations and experiments on a real network.

The architecture of Neko can be divided in three main components (see Figure 1): *applications*, *NekoProcesses* and *networks*.

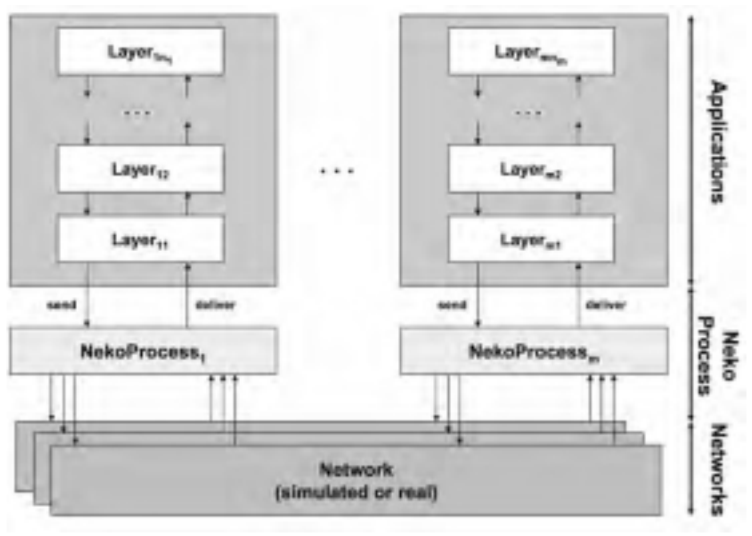


Figure 1: Typical architecture of a Neko-based distributed application

Applications are built following a hierarchical structure based on multiple levels (called *Layers*). Layers communicate using two predefined primitives for message passing.

The Neko communication platform is a *white box*: the developer can use a network available on Neko or he/she can define new network types. Different networks can be used in parallel, and this allows the exchange of different types of message using different networks. Neko *networks* are the lowest level of the architecture of a Neko application. As already mentioned, an implementation of a distributed algorithm can run on top of a real network, as well as on a simulated network, without changing any line of code. In fact, two types of networks are supported by Neko: *real* and *simulated* networks.

A Neko application can be configured through a configuration file, containing information to set up all the involved processes. Then, bootstrapping a Neko application is different for a simulation and a distributed execution. In real executions there is an asymmetry between different processes: there is a *master process*, that coordinates the execution, and *m*-1 *slave processes*.

Although possessing the attractive features exposed so far, the Neko framework lacks any support to quantitative assessments. Therefore, to permit assessment of *quantitative properties* - namely, dependability and performance metrics - we devised, designed and constructed an extension to standard Neko framework.

2. The NekoStat Package

NekoStat extends the V&V analysis features of Neko in the direction of a *statistical dynamic evaluation* of a system, both on simulated and real execution. In Figure 2 a high level view of the structure of a session of analysis using the NekoStat extension is depicted.

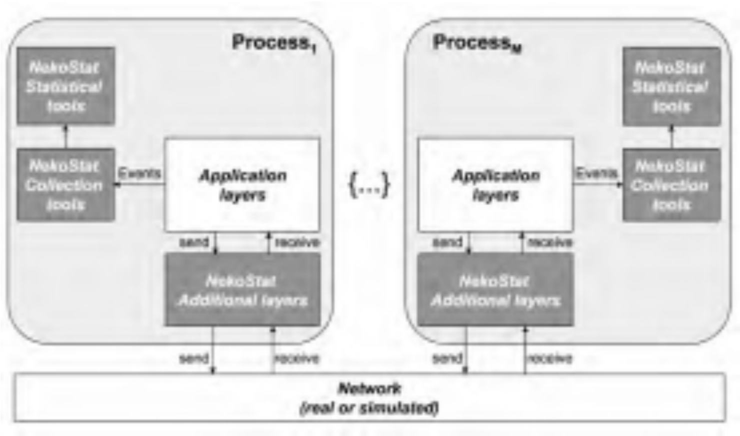

Process$_1$
NekoStat Statistical tools
NekoStat Collection tools
Events
Application layers
send
receive
NekoStat Additional layers
{...}
Process$_M$
Network (real or simulated)

Figure 2: High level view of typical session of analysis of a distributed system made with NekoStat

One of the basic ideas of Neko was to define a framework in which the developer can use the same implementation of a distributed algorithm, both for simulations and real experiments. We wanted to retain this appealing feature in the NekoStat design: the usage of the provided tools is similar both in simulations and in real experiments.

Using NekoStat to obtain an assessment of relevant metrics of a distributed system is simple. First of all, it is necessary to implement the distributed system/algorithm, using the tools available in the Neko framework (Java language). Then, in order to apply the tools available in the NekoStat extension, the following simple modifications have to be performed to the application code:

1. define the interesting *events* and introduce calls to the log(Event) method, of a special predefined logger class (StatLogger) in the points of the source code where the event happens;

2. implement a StatHandler, a class containing the methods to manage the collected *distributed events* and to transform them into *quantities*.

The NekoStat package is actually part of the standard Neko, from the release 0.9 of the tool ([3]).

3. Conclusions And Future Work

NekoStat, an extension to the already existing Neko framework for the analysis of distributed systems/protocols, was here described. Neko, although powerful and easy to use, allows only collection of traces of execution, and does not include any support to manage the gathered events to perform quantitative evaluations, in parallel with the protocol execution. The NekoStat tool is described with details in [4]. An example of use of the NekoStat tool is in the paper [5] where we describe an experimental evaluation of the quality of service of a large class of failure detectors in a WAN environment.

While retaining the appealing features of Neko, NekoStat enriches Neko with mathematical supports to handle the numerical quantities, as well as with analysis supports, to collect relevant distributed events and to analyze them on-line. We are now working on devising additional extensions, both to Neko and to NekoStat, in order to further improve the analysis of distributed systems. In particular, two directions are under investigation:

- to extend the framework to include, as Neko layers, portions of source code of distributed algorithms written in languages different from Java (e.g., C and C++). Removing the restriction to use only algorithm written in Java will allow the analysis of a much richer population of already existing distributed protocols written in languages other than Java, without the necessity of any translation. Apart from easing the analysis process, this feature is very attractive especially in those cases where the translation in Java is not straightforward or possible at all (e.g., because Java does not contain support for some low-level functions). In any case, avoiding the translation improves efficiency and is less error-prone;

- to increase the accuracy in the obtained results; in particular, we are trying to take into account the possible sources of error in the distributed system metrics evaluation. One important point in this direction is the implementation of methods, integrated in the framework, to control the actual level of accuracy of the temporal measurements made using the tool.

References

[1] A. Avizienis, J. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1), 2004.

[2] P. Urbán, X. Défago, and A. Schiper. Neko: a single environment to simulate and prototype distributed algorithms. In *Proc. of the 15th Int’l Conf. on Information Networking (ICOIN-15)*, Beppu City, Japan, February 2001.

[3] P. Urbán. Neko 0.9 website.

http://lsrwww.epfl.ch/neko/.

[4] Lorenzo Falai, Andrea Bondavalli, and Felicita Di Giandomenico. Quantitative evaluation of distributed algorithms using the neko framework: The nekostat extension. In *LADC*, pages 35–51, 2005.

[5] L. Falai and A. Bondavalli. Experimental evalutation of the QoS of failure detectors on Wide Area Network. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2005)*, Yokohama, June 2005.

From Fast to Lightweight Atomic Memory in Large-Scale Dynamic Distributed Systems

Vincent Gramoli – IRISA, INRIA-Rennes, CNRS.

1. Introduction

Replicating a read/write object over a set of distant network locations leads to consistency problem. Assume that an object is modified at some place. From this point on, changes must appear from any location: If a particular client reads the value, then it must return the lastly written value of the object. This simple expression describes the idea at the core of atomic consistency also known as linearizability [Lam86, HW90, Lyn96].

The simplest representation of quorums among a set of *n* elements is the majority sets. Quorums systems are fundamental in distributed shared memory (DSM), because they represent the minimal set of locations where the object must be replicated. That is, reading or writing the object consists simply in accessing a constant number of quorums among the existing ones. The quorum system represents all nodes owning a copy of the object. In the remaining we refer to *n* as its size. The quorum intersection property guarantees that at least one contacted element can testify of the last value written when contacted through a read operation. Therefore quorums are at the core of atomicity and minimizing quorum access time boils down to speed-up read/write operations.

Even though increasing quorum size can tolerate more failures in static systems, dynamic systems can not afford such a solution. Indeed, in static systems where the amount of failures is upper bounded, the intersection between quorums should contain enough elements to guarantee that at least one of it is correct. In dynamic systems where the amount of failures is unbounded, consistency requires additional mechanisms. Among those mechanisms are *reconfiguration* introducing new active quorums, *adaptive* quorum access, and *probabilistic quorum* probing.

This paper discusses the tradeoff relying between operation time and message complexity induced by consistency requirements.

The paper is divided as follows. Section 2 investigates approaches that cope with infinite amount of failures. Section 3 presents solutions to the problem induced by large-scale systems with limited bandwidth capacity. Finally Section 4 concludes the paper.

2. Facing Dynamism

Several existing papers focus on the emulation of shared memory in message passing systems, also known as Distributed Shared Memory (DSM). For instance, Attiya et al. [ABD95] propose a robust single-writer multi-reader (SWMR) shared memory algorithm. This algorithm specifies a two-phase read operation and a single phase write operation, each phase consisting in an exchange of messages between the client and a majority of nodes. More recently, single-phase read operation, namely *fast read*, appeared in [DGLC04, VG06].

1. Reconfigurable Atomic Memory

The robustness in dynamic systems is more challenging. Although using majority of nodes is robust in face of a bounded number of failures, in dynamic systems the number of failures might grow infinitely. That is, reconfiguration has been introduced in RAMBO [LS02] to cope with accumulated failures. The reconfiguration process replaces the sets of contacted nodes by active ones. The quorum replication is sufficient to cope with failures occurring between subsequent reconfigurations.

Fast reconfiguration is at the core of fault tolerance in dynamic systems. More precisely, the faster the system switches to the new active configuration the more fault-tolerant the system is. RDS [CGGMS05] proposes an accelerated reconfiguration process coupled with fast read operations. The RDS algorithm proposes fast reconfiguration using "Fast Paxos" [Lam98, BDFG03] consensus algorithm as a fully-integrated part of the shared memory. RDS implements a leader-based protocol that makes the reconfiguration three message delays long when the leader stabilizes. To conclude, RDS achieves a fault-tolerant MWMR shared memory especially suited for dynamic networks.

2. Restricting the Gossip

Aforementioned algorithms suffer from a potential drawback when applied in large-scale systems where the bandwidth capacity is bounded. Indeed, operation execution at the core of those algorithms relies on the fact that clients know the currently used configuration (i.e., quorum system). Since any node is a potential client, an approach is to maintain global knowledge at each node despite reconfiguration. For this purpose, reconfiguration, one should use all-to-all *gossip* (message exchange). Therefore, the number of messages produced after a reconfiguration is n .

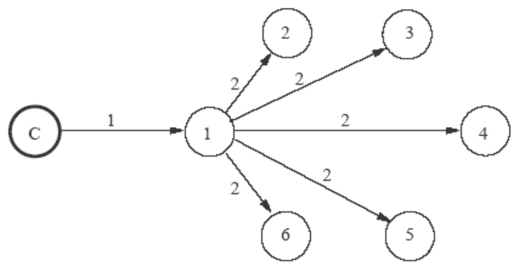


Figure 1. The global knowledge approach

In [GMS05], the authors propose a substantial improvement in order to reduce the communication complexity from quadratic to linear. The key idea is roughly to differentiate replicas from common nodes. In this paper, replicas are called *owners* and they are considered locally by other nodes as active configuration members. Restricting gossip among owners makes reconfiguration possible provided that at least one member of any configuration is active. However, maintaining such knowledge at any node remains unachievable when the number of owners is large, the reconfiguration is frequent, and a configuration member must remain active. These reconfigurable approach requires *O(n)* during a reconfiguration while operations are executed in a constant number of message delays as presented in Figure 1 (the client is represented by a *C*, and the quorum has size 6).

3. Facing Scalability

Because of the recent growing interest for dynamic large-scale systems, a radically different approach has been adopted. This approach relies strongly on a locality principle: participants maintain only a piece of the system information depending on their location. Therefore reconfiguration is made locally and involves less messages.

1. Restricting Knowledge

Assuming that each node maintains the information about a restricted set of nodes, reconfiguration is executed locally, thus, with low communication complexity. The direct drawback of such a solution is the operation latency. Although reconfiguration message complexity is minimized, the time complexity of an operation is increased. Indeed, minimizing knowledge prevents operation from completing after a single round-trip. Suppose that any replica maintains knowledge of only one other replica in a logical overlay. Then, in order to contact an entire quorum a phase will take as many message delays as the number of the quorum elements.

2. Adaptive Atomic Memory

Global/local reconfiguration paradigm is interestingly related to the *proactive/reactive routing* paradigm. The local reconfiguration process is such that only reactive routing can be adopted by phases: a node (not the first contacted one) belonging to the quorum is not identified before the phase reaches its direct neighbor. Unlike local reconfiguration, global reconfiguration process requires that a phase executes a pro-active routing, where the whole set of nodes is known when the phase starts. Algorithms based on the first technique are called *non-adaptive* while algorithms based on the second one are called *adaptive* according to [NW03]. This paper extends this notion to atomic memory.

Recently, some authors have focused on dynamic types of Quorums [NW03, NN05, AM05]. In [AM05], another node is inserted by adding a vertex in a De Bruijn graph, while vicinity is defined by graph neighboring. For instance, the Dynamic Path [NW03] defines quorums in a logical overlay using Voronoi cells and the Dynamic And-Or Quorum System [NN05] defines quorums as a set of leaves.

DSM finds its way into such emergent ideas. SAM [AGGV05, AGGV06], is a read/write memory using adpative operations. Quorums are defined as rows and columns of a dynamic torus grid. This solution provides local reconfiguration, that is, the message required to reconfigure after a single departure is constant. In the meanwhile, the adaptive operations require *O(n)* message delays to complete, where all nodes of a *(n)*-sized quorum are contacted in turn. In the meanwhile reconfiguration requires a constant number of messages, since the number of neighbors is constant. Figure 2 represents the way a quorum is contacted in this example of adaptive approach.

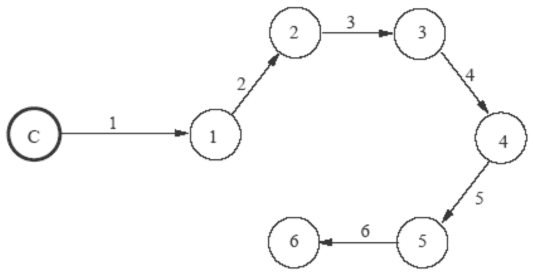


Figure 2. The adaptive approach

3. Probabilistic Atomic Memory

In order to guarantee progress and consistency in asynchronous dynamic systems the previous solutions make some assumptions. For instance, in [CGGMS05] reconfiguration is guaranteed provided that the system stabilizes. Likewise, failure detection is required in [AGGV06] to guarantee consistency.

Recently, probabilistic approaches have been investigated to implement quorum systems. Malkhi et al. defined in [MRWW01] a probabilistic quorum system (i.e. -intersecting quorum system) as a quorum system whose quorum intersection is guaranteed with probability (*1-*). This approach brings a solution for asynchronous dynamic systems where ensuring intersection deterministically remains unachievable without constraining assumptions.

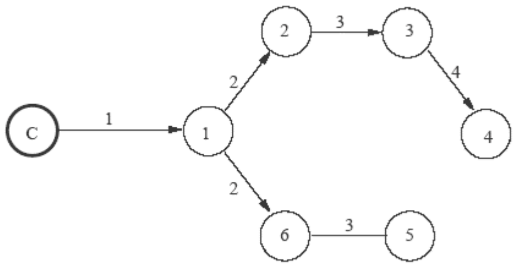


Figure 3. The random walks approach

In [AM05] the authors propose a logical overlay structure using a De Bruijn graph in order to maintain *k*-connectivity. They make this structure dynamic by defining join and leave primitives, each requiring local reconfiguration with *O(log n)* messages before the structure stabilizes. They define quorums that intersect with high probability. To achieve this result, their approach consists in executing *O(n)* random walks (cf. Figure 3), each of length *O(log n)*. That is, a quorum access lasts *O(log n)* message delays.

4. Structureless Atomic Memory

Differently, it is noteworthy that the structure imposes several constraints due to the maintenance of the overlay. When dynamic events—such as join or leave events—occur, the overlay changes. Therefore, the overlay must be readapted to reflect changes involving message exchanges.

Consequently, providing building blocks that are independent from the underlying overlay minimizes communication complexity. In [GKMR06] the authors present a way to preserve object persistence through the use of a set of nodes called a *core*. This core can be used to preserve information persistence, such as quorums, and thus, can serves as DSM building block.

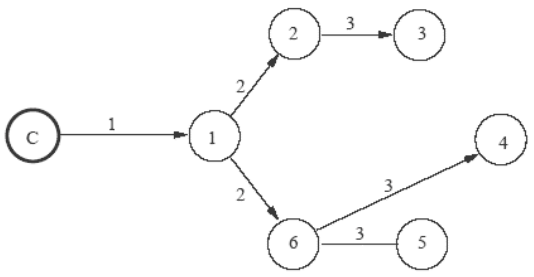


Figure 4. The disseminating approach

The key idea is temporal in the sense that an object persists if operations are sufficiently frequent regarding to the system dynamism, or *churn*---the rate at which nodes enter and leave the system. Roughly speaking, if the object is written at enough locations with a reasonable frequency then the object persists. This building block can be used to provide efficient quorum access with weak maintenance constraint. Figure 4 presents an access through dissemination that is *O(log n)* message delays long.

4. Conclusion

This paper classifies a panel of achievements in the context of quorum-based solution for large-scale dynamic DSM implementation. Two major problems arise from large-scale and dynamic DSM: the time complexity required by operation and the communication complexity required to guarantee consistency while participants join or leave and while object state must reflect write operations. This paper enlightens the inherent tradeoff between these two fundamental issues.

References

[Lam86] On interprocess communication, Part II: Algorithms, L. Lamport, *Distributed Computing*, 1, p.86—101, 1986.

[HW90] Linearizability: a correctness condition for concurrent objects, M. P. Herlihy and J. M. Wing, *ACM Trans. on Programming Languages and Systems*, p.463—492, 12(3), 1990.

[Lyn96] Distributed Algorithms, Lynch, N., Morgan Kaufmann Publishers, 1996.

[ABD95] Sharing memory robustly in message-passing systems, H. Attiya, A. Bar-Noy, and D. Dolev, *J. ACM* 1995, 42(1), p. 124—142.

[DGLC04] How fast can a distributed atomic read be?, P. Dutta and R. Guerraoui and R. R. Levy and A. Chakraborty, *Proc. of the 23th annual symposium on Principles of distributed computing,* p. 236—245, 2004.

[VG06] How fast can a very robust read be?, M. Vukolic and R. Guerraoui, *Proc. of the 25th annual symposium on Principles of distributed computing,* 2006.

[LS02] RAMBO: A reconfigurable atomic memory service for dynamic networks, N. Lynch and A. Shvartsman, *Proc. of 16th International Symposium on Distributed Computing,* p.173—190, 2002.

[CGGMS05] Reconfigurable Distributed Storage for Dynamic Networks, G. Chockler and S. Gilbert and V. Gramoli and P. Musial and A. Shvartsman, *Proc. of 9th International Conference on Principles of Distributed Systems*, 2005.

[Lam98] The Part-time Parliament, Lamport, L., *ACM Transactions on Computer Systems* 16(2), p.133—169, 1998.

[BDFG03] Reconstructing Paxos, R. Boichat and P. Dutta and S. Frolund and R. Guerraoui, *SIGACT News* 34(2), p.42—57, 2003.

[NW03] Scalable and dynamic quorum systems, M. Naor and U. Wieder,, *In Proc. of the 22th annual symposium on Principles of distributed computing,* 2003, p.114—122.

[NN05] The Dynamic And-Or Quorum System, U. Nadav and M. Naor, *Distributed algorithms* p.472—486, 3724, 2005.

[AM05] Probabilistic quorum systems for dynamic systems, I. Abraham and D. Malkhi, *Distributed Systems* 18(2), p.113—124, 2005.

[GKMR06] Core Persistence in Peer-to-Peer Systems: Relating Size to Lifetime, V. Gramoli and A.-M. Kermarrec and A. Mostefaoui and M. Raynal, Technical Report n 1799, IRISA/INRIA Universite de Rennes 1 - Campus de Beaulieu, Rennes, France, 2006.

[AGGV05] P2P Architecture for Self* Atomic Memory, E. Anceaume and M. Gradinariu and V. Gramoli and A. Virgillito, *Proc. of 8th IEEE International Symposium on Parallel Architectures, Algorithms and Networks* p.214—219, 2005

[AGGV06] Self-Adjusting Atomic Memory for Dynamic Systems based on Quorums On-The-Fly. Correctness Study., E. Anceaume and M. Gradinariu and V. Gramoli and A. Virgillito, Technical Report n 1795,, IRISA/INRIA Universite de Rennes 1 - Campus de Beaulieu, Rennes, France, 2006.

[MRWW01] Probabilistic quorum systems, D. Malkhi and M. Reiter and A. Wool and R. Wright, *Information and Computation* 170(2), p.184—206, 2001.

Dependable Middleware for Unpredictable Environments

Odorico M. Mendizabal, António Casimiro, Paulo Veríssimo
{omendizabal, casim, pjv}@di.fc.ul.pt
FC/UL[2]

Introduction

Nowadays, different types of systems with critical dependability requirements can be envisioned, for example systems and applications in the domain of intelligent home devices, transportation systems or critical services infrastructure.

In all these areas the environment surrounding the applications is open, unpredictable or uncertain. Because of this, it is hard, sometimes even impossible, to provide certain guarantees or to develop dependable applications with timeliness or security requirements based on classical distributed system models, for example synchronous and asynchronous ones. The synchronous distributed computing model provides processes with bounds on processing time and message transfer delay. Differently, the asynchronous model is characterized by the absence of any time bounds.

To develop systems using synchronous models, it is necessary to know a priori the time bounds. Many distributed problems can be solved assuming this model, but if some bound is violated, nothing can be guaranteed about the system safety properties. On the other hand, solutions based on asynchronous models do not rely on any time bounds. Unfortunately, because this is much weaker, some problems are impossible to solve in asynchronous environments, such as the well known consensus problem.

A promising way to address these problems consists in assuming a hybrid distributed system model. Thus, it is possible to devise models weaker than synchronous models but stronger than asynchronous models. A recent work presents the wormhole[13], a hybrid distributed system model that allows to overcoming some of the difficulties faced when asynchronous models (uncertainty) meet timing specifications (predictability).

---

[2]Faculdade de Ciências da Universidade de Lisboa. Bloco C6, Campo Grande, 1749-016 Lisboa, Portugal. Navigators Home Page: http://www.navigators.di.fc.ul.pt. This work is partially supported by the EC, through project IST-FP6-STREP-26979 (HIDENETS), through the Large-Scale Informatics System Laboratory (LASIGE).

The wormhole model is composed by different parts that can be defined assuming different sets of properties, which could map into different synchrony levels. The most synchronous parts can provide strong properties to the model. Considering such a context, a potentially asynchronous part can rely on specialized services provided by a more synchronous part, such as duration measures, timing failure detection and timely execution[1] .

In fact, hybrid models are suitable to reason about applications running over open and unpredictable environments. High synchrony levels on distributed systems can be supported by existent technologies (*e.g.* GPS and dedicated networks).

2 Dependable Middleware: Initial Ideas

The possibility of constructing more dependable applications assuming these hybrid models motivates the development of middleware tailored for unpredictable environments. The conception of a middleware suitable to these environments needs to take into account many important aspects of distributed and fault tolerant systems, such as timing failure detection, accurate measure services, QoS adaptation, replicas management, etc. Previous works researching some of these features present very important issues and encourage us to further investigate resilient solutions for unpredictable environments.

A timing failure detection approach using a specific instance of the wormhole model, called TCB was proposed in [2]. The TCB model is used to provide specialized services such as duration measurement, timely execution and timing failure detection. However, we may need to adequate the interface between the timing failure detector and applications, making use of this service as a building block to construct other services, maybe with different failure detection properties[2] .

In [1] a framework for dependable QoS adaptation using TCB was presented. The TCB has the knowledge of available resources, measured in terms of timeliness. Therefore, for some classes of applications, for example the *time-elastic*[3] ones, it is possible to ensure the coverage stability of the applications, based in probability distribution function (*pdf*) that represents the actual distribution of the observed timing variable that determines the available QoS. However, there are a number of more or less subtle issues still requiring attention before the approach can effectively be used in concrete applications. For instance, we will investigate suitable probabilistic distributions to generate *pdfs* in order to enhance the adaptation of the applications to achieve these improvements we intent to apply techniques based on stochastic models and refine the results through experimental evaluation, for instance using the techniques described in [6, 7].

Providing fault-tolerance in such unpredictability environments is a challenging task. Thus, we will investigate an adaptive model for fault-tolerant distributed computing using hybrid models. For this we intend to focus on the well-known consensus problem, taking into account adaptability and timeliness needs. In [2] it was discussed the anatomy of timing failures and some problems observed due to these failures. Unexpected delay,

[1]In this work we will focus on timely requirements, but other facets, such as security, could also be considered.

[2]Other failures detectors for crash or omission could be implemented over a timed failure detection building block.

[3]*Time-elastic* application are those whose bounds can be increased or decreased dynamically.

contamination and decreased coverage are possible undesired behaviors in the faulty systems. These observations together with dependable QoS adaptation remarks showed in [1], give us a motivation to devise protocols to solve the consensus relying on QoS negotiation and monitoring.

Once that consensus can be solved in these environments, reliable atomic broadcast as well as transaction-based protocols can be developed. In [3] it is proven that consensus and atomic broadcast are equivalent problems in asynchronous distributed systems prone to process crashes.

3. Related Work

In order to guarantee timing and safety properties in unpredictable environments, some authors have defined distributed system models stronger than synchronous and weaker than asynchronous. Cristian and Fetzer have devised the timed-asynchronous model, where the system alternates between synchronous and asynchronous behavior, and where parts of the system have just enough synchronism to make decisions such as detection of timing failures [4]. Almeida e Veríssimo have devised the quasi-synchronous model, where parts of the system have enough synchronism to perform real time actions with a certain probability[14]. Others papers dealt with system that are not completely asynchronous, *e.g.* [5], they are called partially synchronous models.

As stated before, we will investigate solutions based in hybrid models, in particular the wormhole model[12, 13]. Wormhole features several subsystems following different sets of assumptions, *e.g.* about synchrony or faults. In the context of our work we are specially interested in applying the wormhole model to distributed systems, rather than locally to a particular node. On the other hand, we are more interested in the time facet of wormholes.

In [2], it was presented a generic timing fault detection approach reasoning in terms of QoS specification to express timeliness and reliability requirements. Other works propose alternative solutions to timely failure detection [9, 10]. These approach are not provide a general solution, they only secure a restrict semantics, or simply provide ad-hoc solutions.

Many works dealing with QoS assume that it is possible to reserve resources[11, 15]. Works dealing with mapping application level requirements into lower level (support subsystem, network, OS) QoS requirements, provide the grounds for the definition of useful interfaces under hybrid models between applications and the middleware.

Further, to provide a suitable protocol to solve consensus in these unpredictable environments works related to QoS adaptation, discussed above, could be very important to achieve the first issues. Therefore, in [8] was presented a hybrid and adaptive model to solve consensus. However, the authors assuming that the protocol run over an architecture equipped with QoS capabilities. We want to devise a protocol to solve the consensus providing the QoS support through a middleware based in the wormhole model.

Reference

[1] A. Casimiro and P. Veríssimo. Using the timely computing base for dependable QoS adaptation. In *Proceedings of the 20th IEEE Symposium on Reliable Distributed Systems*, pages 208–217, New Orleans, USA, October 2001. IEEE Computer Society Press.

[2] A. Casimiro and P. Veríssimo. Generic timing fault tolerance using a timely computing base. In *Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pages 27–36, Washington DC, USA, June 2002. IEEE Computer Society Press.

[3] T. Deepak Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996.

[4] F. Cristian and C. Fetzer. The timed asynchronous distributed system model. In *Proceedings of the 28th IEEE Symposium on Fault Tolerant Computing Systems (FTCS-28)*, pages 140–149, June 1998.

[5] C. Dwork, N. A. Lynch, and L. J. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988.

[6] L. Falai and A. Bondavalli. Experimental evaluation of the QoS of failure detectors on wide area network. In *DSN*, pages 624–633, 2005.

[7] L. Falai, A. Bondavalli, and F. Di Giandomenico. Quantitative evaluation of distributed algorithms using the neko framework: The nekostat extension. In *LADC*, volume 3747 of *Lecture Notes in Computer Science*, pages 35–51. Springer, 2005.

[8] S. Gorender, R. A. Macêdo, and M. Raynal. A hybrid and adaptive model for fault-tolerant distributed computing. In *DSN*, pages 412–421, 2005.

[9] S. Krishnamurthy, W. H. Sanders, and M. Cukier. A dynamic replica selection algorithm for tolerating timing faults. In *DSN*, pages 107–116. IEEE Computer Society, 2001.

[10] S. Krishnamurthy, W. H. Sanders, and M. Cukier. An adaptive framework for tunable consistency and timeliness using replication. In *DSN*, pages 17–26. IEEE Computer Society, 2002.

[11] F. Siqueira and V. Cahill. Quartz: A QoS architecture for open systems. In *ICDCS*, pages 197–204, 2000.

[12] P. Veríssimo. Uncertainty and Predictability: Can They Be Reconciled? In *Future Directions in Distributed Computing*, volume 2584 of *lncs*, pages 108–113. spring, 2003.

[13] P. Veríssimo. Travelling through wormholes: a new look at distributed systems models. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 37(1, (Whole Number 138)):—, 2006.

[14] P. Veríssimo and C. Almeida. Quasi-synchronism: a step away from the traditional fault-tolerant real-time system models. *Bulletin of the Technical Committee on Operating Systems and Application Environments (TCOS)*, 7(4):35–39, 1995.

[15] D. Xu, D. Wichadakul, and K. Nahrstedt. Multimedia service configuration and reservation in heterogeneous environments. In *ICDCS*, pages 512–519, 2000.

A Language Support for Fault Tolerance in Service Oriented Architectures

Nicolas Salatge

LAAS-CNRS, 7 avenue du Colonel Roche, 31077 Toulouse Cedex 04 – France

Introduction

Service Oriented Architectures (SOA) [1] enable the development of loosely-coupled and dynamic applications. Such applications are based on a core notion, the notion of service, and on a contract linking a client and a service provider. This type of architecture is currently used for large-scale applications like e-commerce, but in the next future for applications having stronger dependability requirements.

Beyond contract issues, the problem relates to a fundamental client-provider conflicting standpoint. The clear aim of a provider is to develop a service to attract as many clients as possible all over the world. As a matter of fact, he is not concerned with detailed dependability needs of individual clients. A client developing a WS-based application with dependability constraints has a different perception. The aim is certainly to find the right service on the Net providing the expected functionality, but this is not enough. Some additional specific dependability constraints must be fulfilled and are very much dependent of specific dependability service oriented application. This exactly the sort of problem we tackle in our work. Application developers look at Web Services as COTS (Commercial Off-The_Shelf) components and thus they ignore their implementation and their behaviour in the presence of faults.

In a sense, clients need to develop specific fault tolerance mechanisms well suited to their application. To this aim, we propose a framework to help clients making Specific Fault Tolerance Connectors (SFTC) that implement filtering and other robustness and error detection techniques (e.g. runtime assertions) together with recovery mechanisms that are triggered when the WS does not satisfy anymore the dependability specifications (see figure 1). The same Web Service can be used in several service oriented applications with different dependability constraints and thus taking advantage of several connectors.

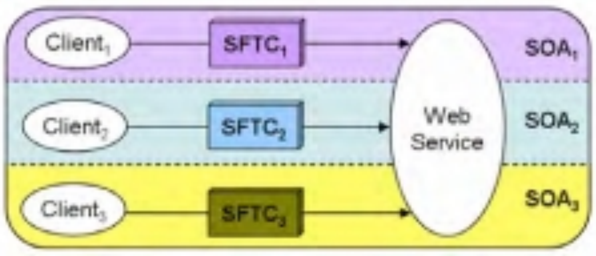


Figure 1: The Specific Fault Tolerance Connectors concept

The problem is similar to the use of COTS components in safety critical systems, and previous work showed that mechanisms like fault containment wrappers was a possible solution [2]. In the AOS context, pre-defined wrappers cannot be defined to satisfy all possible needs. The approach must be more adaptive and enable dependability mechanisms 1) to be defined on a case-by-case basic for a given WS usage and 2) to be highly dynamic and possibly changed according to the needs. To this aim it is mandatory to provide SOA developers with:

- a language support (DeWeL) to describe the dependability features of a connector and,
- a support infrastructure to dynamically manage and run connectors in real applications.

We present the main characteristic of DeWeL in section 1. A program example is given in Section 2. Section 3 introduces the executive support of connectors developed in DeWeL. Section 4 discusses the main benefits of this approach and concluded this paper.

1. DeWeL Specification

DeWeL provides appropriate abstractions and notations to reach the right level of expressiveness together with the appropriate language restriction to obtain reliable code. With these language constraints (see table 1.1), one can expect improving the quality of the connector implementation, i.e. the quality of the runtime code because some critical properties can be verified at compile-time (e.g. program block termination, predictable usage of resources, etc.). The merit of this approach is to ensure a high confidence level in the connector development. For instance, a user can reuse pre-defined recovery strategies like re-directions, replication protocols, generation and catch of exceptions, and insert executable assertions to input and output messages …etc. The objective is to help the user not familiar with Web Services (i.e. WSDL syntax, SOAP messages, etc,) in writing robust code corresponding to its fault tolerance needs.

In order to avoid learning a new language and, more importantly, to simplify its use and understanding for the user, DeWeL borrows its syntax from C (for conditional actions, arithmetic and logic expressions, etc.). However, DeWeL differs very much from general purpose programming languages like C. It can be seen as a declarative language, in particular regarding recovery strategies that are only parameterized, and as a restricted imperative language for the expression of runtime assertions. The later correspond in pre and post conditions that encapsulate the execution of the service (like Before, After, Around advices in Aspect Oriented Programming).

Restrictions language	**Error avoidance**	**Checked property**
- No dynamic allocation - No pointers	Bus error, Segmentation fault, not enough memory	Resources and memory control
- No files		
- No indexed access to arrays	Table overflow	
- No standard loops (while, for)	Service hang	Termination
- No functions - No method overriding		
- No recursive construct		
- No external access to other users data space or system resources	Data corruption	Non-interference

Table 1.1: Essential characteristics of DeWeL

2. Example of DeWeL program

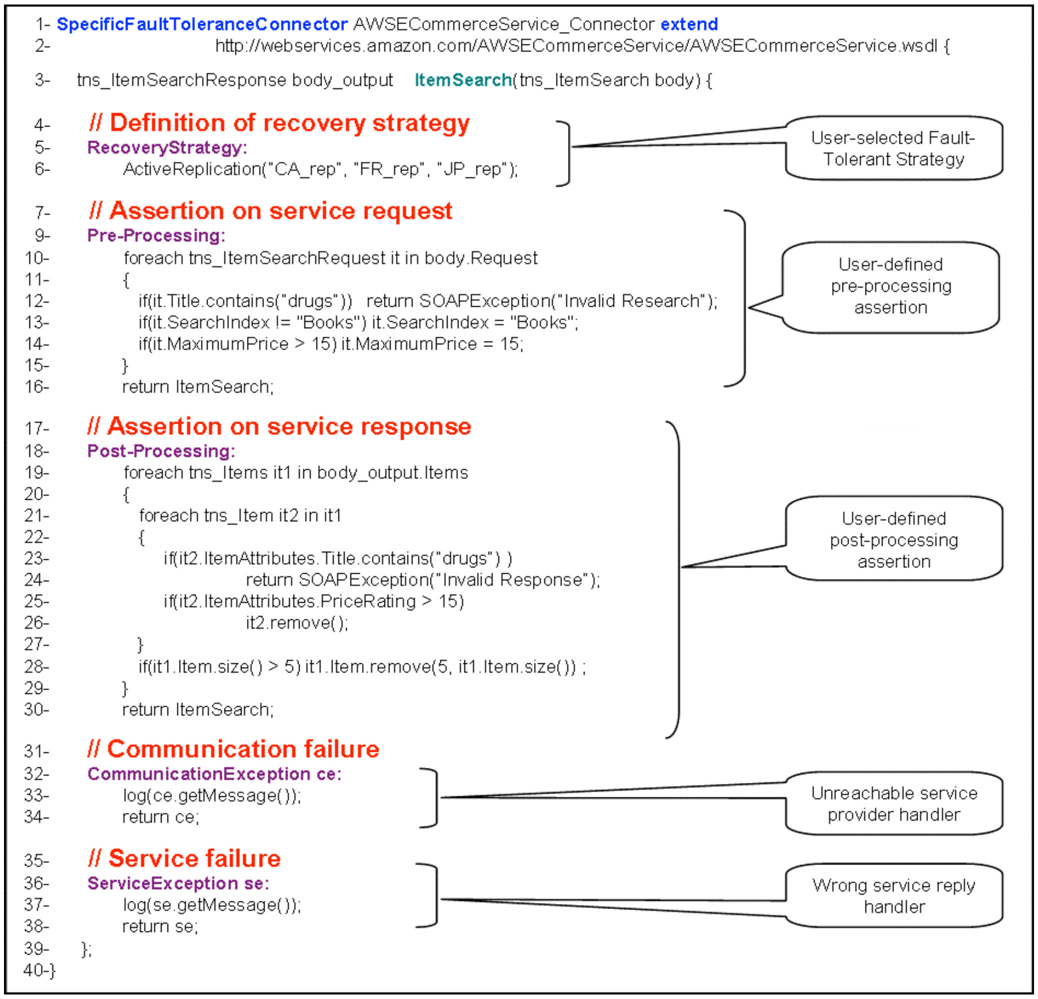
In this example only one WS operation is given (ItemSearch) for the AWSECommerceService of Amazon. The original WSDL document1 is extended with fault tolerance mechanisms. A DeWeL program is in fact developed from a pre-defined template composed of five main sections (see figure 2.1):

- Declaration of the selected RecoveryStrategy;
- Definition of Pre-Processing assertions;
- Definition of Post-Processing assertions;
- Definition of Communication-Exception handlers;
- Definition of ServiceException handlers.

Figure 2 shows a template in DeWeL and the way to produce a Specific Fault Tolerant Connector implementation. The example given here targets the Amazon Web Service and shows how active replication can be selected (line 6) and how various simple assertions are implemented (line 10-16 for pre-processing and 19-30 for post-processing assertions). Error handlers can be defined when the service is unreachable (Communication-Exception, line 31-34) or when it returns a SOAP error (Service-Exception, line 35-39).

In this very simple example, the pre-processing assertions simply checks for illustration an upper bound on some parameter and the post-processing assertion filters both unacceptable replies and upper bounds on SOAP return messages. Such example has been used in our first experiments. Among the six available contracts for Amazon, three Amazon WS replicas (in Canada, France and Japan) have been used to illustrate an active replication error recovery strategy.

[1] http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl



```
SpecificFaultToleranceConnector AWSECommerceService_Connector extend
                    http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl {
 tns_ItemSearchResponse body_output   ItemSearch(tns_ItemSearch body) {

  // Definition of recovery strategy
  RecoveryStrategy:
       ActiveReplication("CA_rep", "FR_rep", "JP_rep");

  // Assertion on service request
  Pre-Processing:
       foreach tns_ItemSearchRequest it in body.Request
       {
         if(it.Title.contains("drugs"))  return SOAPException("Invalid Research");
         if(it.SearchIndex != "Books") it.SearchIndex = "Books";
         if(it.MaximumPrice > 15) it.MaximumPrice = 15;
       }
       return ItemSearch;

  // Assertion on service response
  Post-Processing:
       foreach tns_Items it1 in body_output.Items
       {
         foreach tns_Item it2 in it1
         {
             if(it2.ItemAttributes.Title.contains("drugs") )
                       return SOAPException("Invalid Response");
             if(it2.ItemAttributes.PriceRating > 15)
                       it2.remove();
         }
         if(it1.Item.size() > 5) it1.Item.remove(5, it1.Item.size()) ;
       }
       return ItemSearch;

  // Communication failure
  CommunicationException ce:
       log(ce.getMessage());
       return ce;

  // Service failure
  ServiceException se:
       log(se.getMessage());
       return se;
 };
}
```

User-selected Fault-Tolerant Strategy
User-defined pre-processing assertion
User-defined post-processing assertion
Unreachable service provider handler
Wrong service reply handler

Figure 2.1: Example of a DeWeL program (for Amazon)

3. Runtime & Management Support

The management and the execution of DeWeL connectors rely on a specific platform that is a third-party infrastructure between clients and providers. The storage, the look-up, the delivery, the loading and the execution of Specific Fault Tolerance Connectors are supported by the IWSD platform, an Infrastructure for Web Services Dependability. As soon as a DeWeL program has been compiled, the corresponding connector (i.e. a dynamic library) is created and stored into the SFTC Repository of the platform. At runtime, the platform provides support to perform the recovery mechanisms requested by the user.

The IWSD platform (see figure 3.1) is composed of several software components responsible for the following actions:

- 1) Interception and routing of SOAP messages from clients or providers.
- 2) Processing of requested « Service Web » requests using a user-defined connector developed in DeWeL. The corresponding dynamic library is loaded so that pre and post-processing can be scheduled at runtime.

- 3) Compilation of DeWeL programs and storage of corresponding connectors into the SFTC Repository, management of user accounts and configuration information (e.g. communication endpoints replicas for a given service).
- 4) Health Monitor that is responsible for (i) the self-checking of the IWSD platform and (ii) the evaluation of the current health of WS in operation by means of errors reported by the active connectors.

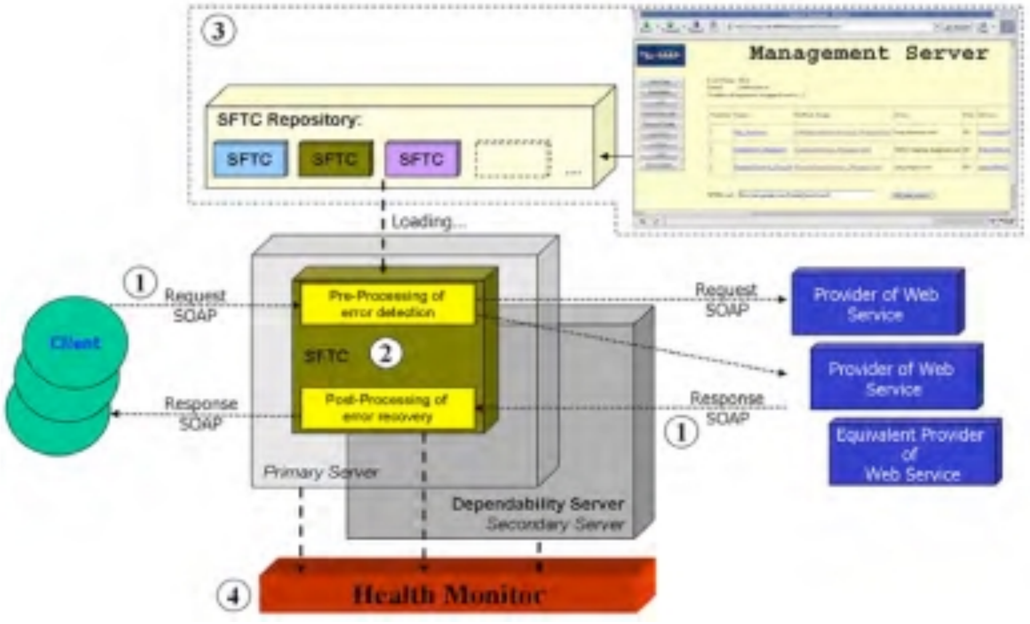


Figure 3.1: IWSD: Infrastructure of Web Service Dependability

Based on elements given into the original WSDL document of the Web Service, a compiler is able to generate specific non-functional mechanisms as a connector and attached to the service. A new service is obtained with better but more importantly user-defined fault tolerance features. This enhanced version can be attached to a new WSDL document and a new access point. This new access point is registered into IWSD. Each connector is thus visible as a WSDL document on the Web. In practice, this document is used by clients to access a given connector targeting a web service, in other words, to access a WS through a Specific Fault Tolerance Connector.

As show in the figure the platform itself can be made fault tolerant using conventional techniques, for instance using a duplex architecture to tolerate crash faults. A detailed description of IWSD is out of the scope of this paper (see [3]).

4. Discussion and conclusion

Dependability issues are the major topic limiting the deployment of SOA in critical domains [4]. The main founders of the Web Services technology (IBM et Microsoft) have spent huge efforts to develop specific WS protocols to guaranty security and safety properties [5]. These protocols are enforced by means of a specific middleware enabling the insertion of dependability mechanisms [6, 7], but this solution is intrusive. Some platforms have also emerged to apply generic recovery strategies using replication [8, 9] but this approach targets WS implementation in isolation.

To address this issue and make adaptive fault tolerance mechanisms to client needs, we propose an infrastructure enabling clients to develop, manage and execute Specific Fault Tolerance Connectors to Web Services. A key feature of this approach is to rely on a Domain Specific Language to develop connectors. DeWeL aims at providing the necessary and sufficient language features to (1) declare and customize built-in fault tolerance strategies and (2) express runtime assertions for each individual operation of a service. The language is supported by a tool suite that applies checks at multiple levels in order to produce robust connectors. Its expressiveness is limited by construction to reach this aim but its conciseness is of high interest to generate reliable code.

The evaluation of a DSL is not always easy as it relates to compiler facilities. However, the essential characteristics of a DSL are expressiveness, conciseness, performance, properties that must be enforced as discussed in [10]. The evaluation of expressiveness of a DSL in practice implies the use of a large set of applications. We have used DeWeL to produce SFTCs for about 150 Web Services, and implemented a TMR connector with Amazon. On these tested services, the DeWeL expression is 18 times smaller that its counterpart in C++, for a program with empty assertion sections. The average overhead is about 3,5% of the response time.

In addition, no specific protocol is needed to use DeWeL connectors in practice. Unlike to SLAs carried out by WS-Agreement protocol [11] and the WSOL language [12], DeWeL does not specify the quality of service of the provider, IWSD just verifies it owing to connectors. From practical viewpoint, the core of this project has been realized with the Xerces-C library. It roughly represents 65000 codes lines corresponding to the implementation of the DeWeL compiler and the connector support infrastructure.

References

[1] H. He, "What is Service-Oriented Architecture?" *http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html*, Sept, 2003.

[2] F. Salles, M. R. Moreno, J. C. Fabre, and J. Arlat, "Metakernels and fault containment wrappers," *29th IEEE International Symposium on Fault-Tolerant Computing (FTCS-29), Madison (USA)*, pp. 22-29, 15-18 June 1998.

[3] N. Salatge and J.-C. Fabre, "A Fault Tolerance Support Infrastructure for Web Services," *LAAS Research Report n°05703, 20 pages*, 2005.

[4] F. Tartanoglu, V. Issarny, A. Romanovsky, and N. Levy, "Dependability in the Web Services Architecture," *In Architecting Dependable Systems. LNCS 2677*, June 2003.

[5] T. S. Donald F. Ferguson, Brad Lovering, John Shewchuk, "Secure, Reliable, Transacted Web Services: Architecture and Composition," *IBM and Microsoft Corporation, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/ wsOverView.asp*, Septembre 2003.

[6] O. Hasan and B. Char, "A deployment-ready solution for adding quality-of-service features to web services " *The 2004 International Research Conference on Innovations in Information Technology*, 2004.

[7] M. G. Merideth, A. Iyengar, T. Mikalsen, S. Tai, I. Rouvellou, and P. Narasimhan, "Thema: Byzantine-fault-tolerant middleware for Web-service applications," *Reliable Distributed Systems, 2005. SRDS 2005. 24th IEEE Symposium* pp. Page(s):131 - 140 2005.

[8] D. Liang, C.-L. Fang, and C. Chen, "FT-SOAP: A Fault-tolerant web service," *Tenth Asia-Pacific Software Engineering Conference, Chiang Mai, Thailand*, 2003.

[9] G. T. Santos, L. C. Lung, and C. Montez, "FTWeb: A Fault Tolerant Infrastructure for Web Services," *In the Proceedings of the 2005 Ninth IEEE International EDOC Enterprise Computing Conference (EDOC'2005)* 2005.

[10] C. Consel and R. Marlet, "Architecturing Software Using A Methodology for Language Development," *In International Symposium on Programming Languages, Implementations, Logics and Programs (PLILP '98), LNCS 1490, Pisa, Italy*, pp. 170-194, 1998.

[11] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, "Web Services Agreement Specification (WS-Agreement)," *Draft 18 Version 1.1*, May 2004.

[12] V. Tosic, B. Pagurek, K. Patel, B. Esfandiari, and W. Ma, "Management applications of the Web Service Offerings Language (WSOL) " *Information Systems 30* pp. 564–586, 2005.

Challenges for an Interoperable Data Distribution Middleware

Sirio SCIPIONI

Dipartimento di Informatica e Sistemistica

Università di Roma “La Sapienza"

Via Salaria 113, 00198 Roma, Italia.

scipioni@dis.uniroma1.it

Introduction

Middleware for data distribution is a natural match, and often a fundamental architectural building block, for a large class of real-time, mission, and safety critical application domains, such as industrial process control, air traffic control, defense systems, etc.

Historically, most of the pub/sub middleware standards such as the Common Object Request Broker Architecture (CORBA) Event Service (CosEvent), the CORBA Notification Service (CosNotification), and Java Message Service (JMS), etc., as well as most proprietary solutions, have lacked the support needed by real-time, mission, and safety critical systems. The main limitations are typically due to the limited or non-existent support for Quality of Service (QoS), and the lack of architectural properties which promote dependability and survivability, e.g., lack of single point of failure.

Recently, in order to fill this gap, the Object Management Group (OMG) has standardized the Data Distribution Service (DDS). This standard gathers the experience of proprietary real-time pub/sub middleware solutions which had been independently engineered and evolved in niches, within the industrial process control, and in the defense systems applications domain. The resulting standard is based on a completely decentralized architecture, and provides an extremely rich set of configurable QoS.

Currently, several vendors provide their own DDS implementation. Each implementation is characterized by additional services and proprietary extensions to the standard. At present, the DDS specification does not address the issue of interoperability between implementations from different vendors. Hence the integration of multiple DDS applications based on heterogeneous implementations requires the development of custom bridges between them.

For this reason, OMG is currently working on defining a companion specification for DDS, describing a standard interoperability protocol which will allow standard DDS implementations from different vendors to exchange messages.

In this paper we identify the main challenges in the realization of a scalable, QoS-driven, interoperable data distribution middleware. In particular, we identify three of the basic DDS features, related respectively to scalable diffusion, timeliness and data availability.

1. Data Distribution Service

It is worth mentioning that the standard defines two level of interfaces. At a lower level, it defines a Data Centric Publish Subscribe (DCPS) whose goal is to provide an efficient, scalable, predictable, and resource aware data distribution mechanism. Then, on top of the DCPS, it defines the Data Local Reconstruction Layer (DLRL), an optional interface which automates the reconstruction of data, locally, from updates received, and allows the application to access data as if it was local.

1.1. DDS Conceptual Model

The DDS conceptual model is based on the abstraction of a strongly typed Global Data Space (GDS) where publisher and subscriber respectively write (produce) and read (consume) data. In the reminder of this Section we will provide a precise characterization of the entities that constitute this global data space.

- Topic. A topic defines a type that can be legally written on the GDS. In the present standard, topics are restricted to be nonrecursive types defined by means of OMG Interface Definition Language (IDL). The DDS provides the ability to distinguish topics of the same type by relying on the use of a simple key. Finally, topics can be associated with specific QoS.
- Publisher. A publisher, can declare the intent of generating data with an associated QoS, and to write the data in the GDS. The publisher declared QoS has to be compatible with that defined by the topic.
- Subscriber. Subscribers read topics in the global data space for which a matching subscription exist (the rules that define what represents a matching subscription are described below).
- Subscription. A subscription is the logical operation which glues together a subscriber to its matching publishers. In the DDS a matching subscription has to satisfy two different kind of conditions. One set of conditions relate to concrete features of the topic, such as its type, its name, its key, its actual content. The other set of conditions relate to the QoS. Regarding the QoS, the matching follows an requested/offered model in which the requested QoS has to be the same, or weaker, then the offered.

Discovery. Another key feature at the foundation of DDS is that all information needed to establish a subscription is discovered automatically, and, in a completely distributed manner. The DDS discovery service, finds-out and communicates the properties of the GDS's participants, by relying on special topics and on the data dissemination capability provided by the DDS.

1.2. Quality of Service

One of the key distinguishing features of the DDS when compared to other pub/sub middleware is its extremely rich QoS support. By relying on a rich set of QoS policies, the DDS gives the ability to control and limit (1) the use of resources, such as, network bandwidth, and memory,(e.g. with TIME_BASED_FILTER and RESOURCE_LIMITS QoS policy) and (2) many non functional properties of the topics, such as, persistence, reliability, timeliness, etc (e.g. through DEADLINE, DURABILITY, DESTINATION_ORDER and other QoS policies).

2. Challenges

In this section we analyze the challenges hidden behind the realization of an interoperable DDS and propose some basic solutions. The challenges are identified according to three classes of requirements addressed by the DDS specification:

- Scalability: requirements for allowing information diffusion to a number of participants that can grow indefinitely and change overtime.
- Timeliness: requirements for QoS properties related to deadline-constrained message delivery.
- Data availability: requirements for QoS properties related to reliable delivery of events, that require to persistently store messages for retransmission and data objects for surviving subscribers failures.

We abstract the components for addressing the requirements into a service architecture where we identify three services, one for each of the above-identified classes. Each service encapsulates a specific solution that introduces specific modifications into the interoperability protocol.

2.1. Scalability

The ability to support a large number of subscribers and high message rates is one of the main objectives of the DDS specification. Publishers and subscribers are organize in an overlay according to a relationship of mutual knowledge between each other. The basic DDS semantics can be implemented through a simple overlay structure, where a publisher for a topic is directly linked with all the subscriber for the same topic. This solution can be implemented in a simple and efficient way, then it should be the first, natural option in basic, small-scale applications. However, when the number of subscriber grows as well as the rate of update of the data objects, this basic overlay organization presents obvious scalability limits.

2.2. Timeliness

The enforcement of QoS policies regarding data timeliness, such as DEADLINE and LATENCY_BUDGET, is based on the capability of the DDS implementation to determine the time elapsed by messages from the source to the destination. This obviously requires the presence of a logical global clock which is in practice a synchronization mechanism between clocks entities composing the DDS.

Reliable clock synchronization is a popular problem in distributed systems and several solutions have been proposed. The challenges posed by the DDS scenario are the possible lack of a time reference accessible by all nodes and the possible large number of nodes in the system. Moreover a further challenge is how to achieve clock synchronization through a standard protocol involving different implementations of DDS.

2.1. Scalability

The idea behind the DDS data model is to achieve a logical global data space where data is written by publishers and read by subscribers. In general, this association between data and subscribers needs to be managed more flexibly for what concerns data availability QoS. In particular the realization of the DURABILITY property require data objects to survive the respective subscribers. Similarly, the HISTORY and LIFESPAN policies require data samples to be kept available after their publication, i.e. to survive the respective publishers.

Hence, a DDS implementation should include a component capable of storing copies of data objects and implement algorithms to keep them consistent. In other words support for data availability QoS is achieved by implementing a Interoperable Persistent Storage Service (IPSS) within the DDS which should be available to both subscribers and publishers for storing updated data objects and histories of data samples. The PSS component could be implemented in separate dedicated set of replicated servers or as a peer-to-peer infrastructure among publishers and subscribers.

Proactive Resilience

Paulo Sousa
University of Lisboa, Portugal
pjsousa@di.fc.ul.pt

Abstract

Building resilient intrusion-tolerant distributed systems is a somewhat complex task. Recently, we have increased this complexity, by presenting a new dimension over which distributed systems resilience may be evaluated — exhaustion-safety. Exhaustion-safety means safety against resource exhaustion, and its concrete semantics in a given system depends on the type of resource being considered. We focus on replicas and on guaranteeing that the typical assumption on the maximum number of replicas failures is never violated. An interesting finding of our work is that it is impossible to build a replica-exhaustion-safe distributed intrusion-tolerant system under the asynchronous model.

This result motivated our research on finding the right model and architecture to guarantee exhaustion-safety. The main outcome of this research was *proactive resilience* — a new paradigm and design methodology to build replica-exhaustion-safe intrusion-tolerant distributed systems. Proactive resilience is based on architectural hybridization: the system is asynchronous in its most part and it resorts to a synchronous subsystem to periodically recover the replicas and remove the effects of faults/attacks.

We envisage that proactive resilience can be applied in many different scenarios, namely to secret sharing, and to state machine replication. In the latter context, we present in this paper a new result, yet to be published, that a minimum of $3f + 2k + 1$ replicas are required for tolerating f Byzantine faults and maintaining availability, k being the maximum number of replicas that can be recovered simultaneously through proactive resilience.

1. Exhaustion-Safety and Proactive Resilience

A distributed system built under the asynchronous model makes no timing assumptions about the operating environment: local processing and message deliveries may suffer arbitrary delays, and local clocks may present unbounded drift rates [7, 4]. Thus, in a (purely) asynchronous system one cannot guarantee that something will happen before a certain time.

Consider now that we want to build a dependable intrusion-tolerant distributed system, i.e., a distributed system able to tolerate arbitrary faults, including malicious ones. Can we build such a system under the asynchronous model?

This question was partially answered, twenty years ago, by Fischer, Lynch and Paterson [5], which proved that there is no deterministic protocol that solves the consensus problem in an asynchronous distributed system prone to even a single crash failure. This impossibility result (commonly known as FLP) has been extremely important, given that consensus lies at the heart of many practical problems, including membership, ordering of messages, atomic commitment, leader election, and atomic broadcast. In this way, FLP showed that the very attractive asynchronous model of computation is not sufficiently powerful to build *certain* types of fault-tolerant distributed protocols and applications.

What are then the minimum synchrony requirements to build a dependable intrusion-tolerant distributed system?

If the system needs consensus (or equivalent primitives), then Chandra and Toueg [3] showed that consensus can be solved in asynchronous systems augmented with failure detectors (FDs). The main idea is that FDs operate under a more synchronous environment and can therefore offer a service (the failure detection service) with sufficient properties to allow consensus to be solved.

But what can one say about intrusion-tolerant asynchronous systems that do not need consensus? Obviously, they are not affected by the FLP result, but are they dependable?

Independently of the necessity of consensus-like primitives, we have recently shown that relying on the assumption of a maximum number of f faulty nodes under the asynchronous model can be dangerous. Given that an asynchronous system may have a potentially long execution time, there is no way of guaranteeing exhaustion-safety, i.e., that no more than f faults will occur, especially in malicious environments [11]. Therefore, we can rephrase the above statement and say that the asynchronous model of computation is not sufficiently powerful to build *any* type of (exhaustion-safe) fault-tolerant distributed protocols and applications.

To achieve exhaustion-safety, the goal is to guarantee that the assumed number of faults is never violated. In this context, proactive recovery seems to be a very interesting approach [10]. The aim of proactive recovery is conceptually simple – components are periodically rejuvenated to remove the effects of malicious attacks/faults. If the rejuvenation is performed frequently often, then an adversary is unable to corrupt enough resources to break the system. Proactive recovery has been suggested in several contexts. For instance, it can be used to refresh cryptographic keys in order to prevent the disclosure of too many secrets [6, 16, 1, 15, 9]. It may also be utilized to restore the system code from a secure source to eliminate potential transformations carried out by an adversary [10, 2].

Therefore, proactive recovery has the potential to support the construction of exhaustion-safe intrusion-tolerant distributed systems. However, in order to achieve this, proactive recovery needs to be architected under a model sufficiently strong that allows regular rejuvenation of the system. In fact, proactive recovery protocols (like FDs) typically require stronger environment assumptions (e.g., synchrony, security) than the rest of the system (i.e., the part that is proactively recovered). This is hard or at all impossible to achieve in homogeneous systems' models [14].

Proactive resilience is a new and more resilient approach to proactive recovery based on architectural hybridization and wormholes [12]. We argue that the proactive recovery subsystem should be constructed in order to assure a synchronous and secure behavior, whereas the rest of the system may even be asynchronous. The Proactive Resilience Model (*PRM*) proposes to model the proactive recovery subsystem as an abstract component – the Proactive Recovery Wormhole (PRW). The PRW may have many instantiations depending on the application proactive recovery needs.

2. An Application Scenario: State Machine Replication

In a previous work [13], we explain how proactive resilience can be used to build an exhaustion-safe state machine replication system. We also give the intuition that the redundancy quorum to tolerate Byzantine faults should have in account the number of simultaneous recoveries triggered through proactive resilience. More recently, we have found precisely how much extra redundancy is needed. We present next a brief explanation of our result.

2.1 Why $n \quad 3f + 2k + 1$?

Consider that you have a replicated state machine replication system with n replicas, able to tolerate a maximum of f Byzantine faults, and where rejuvenations occur in groups of at most k replicas. At any time, the minimum number of replicas assuredly available is $n - f - k$. So, in any operation, either intra-replicas (e.g., a consensus execution), or originated from an external participant (e.g., a client request), a group with $n - f - k$ replicas will be used to execute the operation. Given that some of these operations may affect the state of the replicated system, one also needs to guarantee that any two groups of $n - f - k$ replicas intersect in at least $f + 1$ replicas (i.e., one correct replica). Therefore, we need to guarantee that $2(n - f - k) - n \quad f + 1$, which can only be satisfied if $n \quad 3f + 2k + 1$.

The reasoning above can be seen in practice by analyzing the Byzantine dissemination quorum system construction [8], which applies to replicated services storing self-verifying data, i.e., data that only clients can create and to which clients can detect any attempted modification by a faulty server (e.g., public key distribution system). In a dissemination quorum system, the following properties are satisfied:

Intersection any two quorums have at least one correct replica in common;
Availability there is always a quorum available with no faulty replicas.

If one designates $|Q|$ as the quorum size, then the above properties originate the following conditions:

Intersection $2|Q| - n \quad f + 1$;
Availability $|Q| \quad n - f - k$.

From these conditions, it results that we need $n \quad 3f + 2k + 1$ in order to have a dissemination quorum system in a environment where at most f replicas may behave arbitrarily, and at most k replicas may recover simultaneously (and thus become unavailable during certain periods of time). In the special case when $n = 3f + 2k + 1$, it follows that $|Q| = 2f + k + 1$.

3 Future Work

We are in the process of implementing an experimental prototype of a proactive resilient state machine replication system. The goal is to compare, in practice, the resilience of such a system with the resilience of similar ones built using different approaches.

Another objective we are pursuing is to research ways of using proactive resilience to mitigate Denial-of-Service attacks.

References

[1] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *CCS '02: Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 88–97, 2002.

[2] M. Castro and B. Liskov. Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461, Nov. 2002.

[3] T. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, Mar. 1996.

[4] F. Cristian and C. Fetzer. The timed asynchronous system model. In *Proceedings of the 28th IEEE International Symposium on Fault-Tolerant Computing*, pages 140–149, 1998.

[5] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, Apr. 1985.

[6] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology,* pages 339–352. Springer-Verlag, 1995.

[7] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[8] D. Malkhi and M. Reiter. Byzantine quorum systems. In *Proceedings of the 29th ACM Symposium in Theory of Computing*, pages 569–578, May 1997.

[9] M. A. Marsh and F. B. Schneider. CODEX: A robust and secure secret distribution system. *IEEE Transactions on Dependable and Secure Computing,* 1(1):34–47, January–March 2004.

[10] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks (extended abstract). In *Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing*, pages 51–59, 1991.

[11] P. Sousa, N. F. Neves, and P. Veríssimo. How resilient are distributed *f* fault/intrusion-tolerant systems? In *Proceedings of the Int. Conference on Dependable Systems and Networks*, pages 98–107, June 2005.

[12] P. Sousa, N. F. Neves, and P. Veríssimo. Proactive resilience through architectural hybridization. DI/FCUL TR 05–8, Department of Informatics, University of Lisbon, May 2005. http://www.di.fc.ul.pt/tech-reports/05-8.pdf. To appear in Proceedings of the 21st ACM Symposium on Applied Computing (SAC).

[13] P. Sousa, N. F. Neves, and P. Veríssimo. Resilient state machine replication. In *Proceedings of the 11th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 305–309, Dec. 2005.

[14] P. Veríssimo. Travelling through wormholes: a new look at distributed systems models. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 37(1, (Whole Number 138)):—, 2006.

[15] L. Zhou, F. Schneider, and R. van Renesse. COCA: A secure distributed on-line certification authority. *ACM Transactions on Computer Systems*, 20(4):329–368, Nov. 2002.

[16] L. Zhou, F. B. Schneider, and R. V. Renesse. Apss: Proactive secret sharing in asynchronous systems. *ACM Trans. Inf. Syst. Secur.*, 8(3):259–286, 2005.

A Mediator System for Improving Dependability of Web Services

Yuhui Chen
Yuhui.Chen@ncl.ac.uk
School of Computing Science,
University of Newcastle upon Tyne,
United Kingdom

Abstract

This paper presents a novel architectural solution for improving dependability of Web Services (WSs). This solution is based on a concept of WSs Mediator, which acts as a WSs intermediary implementing fault tolerance mechanisms, multi-routing strategy and making use of existing service redundancy. The distributed architecture of this system makes it possible to collect, publish and make decisions using runtime dependability metadata specifically representing the end-user's perspective of the network and component WSs behaviour, therefore paving the way to achieve dependability-explicit operation of WSs [1].

Introduction

The services-oriented architecture (SOA) and the infrastructure on which SOA systems are employed are inherently unreliable [2]. The existing approaches for improving WSs dependability mainly focus on adopting traditional fault tolerance mechanisms, such as exception handling and recovery blocks [3], in developing individual WSs applications. Some proxy-based solutions make use of redundant components and subsystems passively to improve dependability of integration of a group of component WSs within virtual organizations. However many researchers [4-8] indicate that the network-related failures have serious impact on the overall dependability of SOA applications. Our WSs Mediator offers a service-oriented solution for employing fault tolerance techniques in SOA. It is specifically developed to cope with errors of several major types including failures of the individual component WSs and abnormal events in the underlying network infrastructure.

1. The WSs Mediator approach

The WSs Mediator is a general architectural solution. There are two main goals in this approach. Firstly, it implements fault tolerance mechanisms to offer off-the-shelf fault tolerance solutions in SOA. It improves the dependability of pre-existing WSs. The developers of new WSs can also utilize the solutions provided by the WSs Mediator to achieve better dependability of their system at low costs.   Secondly the WSs Mediator innovatively implements dependability-explicit technique to improve the dependability of WSs from the user's perspective.

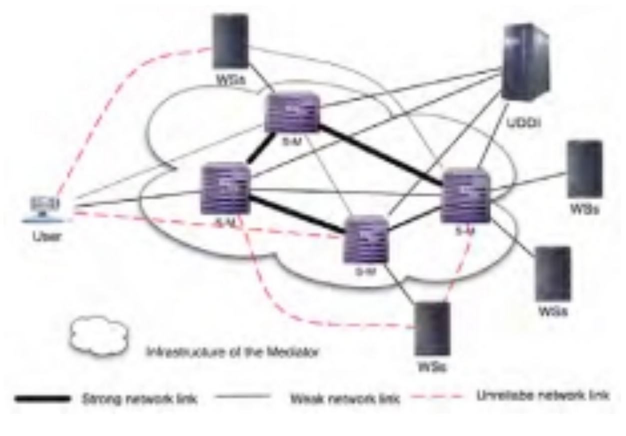


Figure 1: The architecture of the WSs Mediator

1.1 The architecture of the WSs Mediator

The WSs Mediator system consists of a set of Sub-Mediators constituting together an overlay architecture as shown in Figure 1. Sub-Mediators are distributed globally on the major Internet backbones. They have identical functionalities. A Sub-Mediator can work independently to provide services to the users, or it cooperates with each other to enforce dependability-improving strategies. Each Sub-Mediator has dedicated redundant functionality for improving the overall dependability of the WSs Mediator system.

A Sub-Mediator is an intermediary WS between the users and the ultimate WSs they use. The Sub-Mediator supports dynamic reconfiguration. It dynamically processes the users' requests. The behavior of the Sub-Mediator is controlled by dynamic policies. This architecture leads to flexible and optimized fault tolerant execution of the service requests.

1.2 Fault-tolerance solutions.

The WSs Mediator implements several fault tolerance techniques to deal with different type of failures. It provides comprehensive fault tolerance solutions to improve the dependability of WSs. The fault tolerance mechanisms in the WSs Mediator work are implemented as off-the-shelf components. They can be applied independently or in combinations.

- ***Exception handling***

  The WSs Mediaotr implements exception handling mechanism to deal with different type of internal and external failures. This mechanism is able to detect and analyze the exceptions and execute exception handling procedures accordingly.

- ***Recovery Blocks.***

  Recovery blocks is an essential fault tolerance technology adopted in the WSs Mediator system. This mechanism stores SOAP messages received from users. If failures happened during the processing, the stored message can be used to re-launch the processing without interrupting the users.

- ***Multi-routing strategy***

  The WSs Mediator system implements an optimized message routing strategy to overcome network impact between users and WSs. Unpredictable failures such as random network traffic congestion and delays on an Internet backbone [6][9] may occur at anytime. Transport level protocols do not provide efficient solutions to deal with this type of failures for SOA applications. The multi-routing strategy is a desirable solution to improve dependability of the network between users and WSs. This strategy provides a solution to overcome the limitations of the transport level protocols [6].

- ***Employing Service redundancy***

  The loose coupling nature is one of the advantages of the WSs. In some circumstances, if a WS is not available, users can always switch to an alternative WS which provides similar or identical services. It is particularly feasible in specific domains where the WSs have similar or identical API and functionalities. The WSs Mediator system is an off-the-shelf solution to improve dependability of the pre-existing WSs from the user's perspective. It is clear that such WSs will eventually fail. The WSs Mediator implements a mechanism which makes use of service redundancy to overcome this kind of problems. Several (diverse) WSs can be used to achieve N-version strategy [10]. If a WS failed to provide reliable services to the users, alternative WSs can be used.

1.3 Dependability-explicit operation.

The WSs Mediator provides fault tolerance solutions to improve dependability of WSs. However because of the nature of SOA, in some circumstances, traditional fault tolerance technologies are inadequate to improve the dependability of WSs efficiently. The WSs Mediator implements dependability-explicit technique to provide preventive solutions for improving the dependability of WSs especially from the user's perspective.

The Sub-Mediators in the WSs Mediator system collect the information reflecting current dependability characteristics of the available WSs. This is implemented as a runtime dependability daemon for monitoring the WSs and generating dependability-related metadata. These metadata are recorded in a database in each Sub-Mediator and used for online services reasoning and dependability-explicit computing [1]. The metadata consist of a range of values such as failure rates and regular failure types. The main advantage of this metadata collecting approach is that it provides accurate dependability characteristics of WSs from the WSs users' perspective by collecting information reflecting behaviour of the network between the user and WSs [6], therefore it supports fault tolerance specially tailored for the specific users.

2. Mediator extendibility and flexibility

As an off-the-shelf intermediary system the WSs Mediator system can be easily adjusted to follow future development of the WSs standards. Moreover, the flexible design of the WSs Mediator system and its components makes it easy to accommodate the fast-improving WSs technologies such as UDDI and other ontology-based WSs discovery technologies.

3. Conclusion

At present there are several commercial systems focusing on dependability of Web Services. However most of them offer developer-oriented solutions. For instance, the Keynote system [9] uses a globally distributed infrastructure to help web-based application developers to understand the behaviour of their systems. However the true user-oriented solutions for improving dependability of WSs are imperatively needed because of the prominent role of WSs in today's e-Science and e-Business applications. The WSs Mediator system is a dedicated service-oriented and user-oriented solution to meet this demand.

The design of the WSs Mediator system is nearly finished. The full implementation will be completed by June 2006. The system will be evaluated in the following steps. Firstly, we will use an appropriate fault-injection technique to evaluate its behaviour under various network impairments. Secondly, we will apply the system for a small-scale setting involving a number of WSs employed in the e-science applications.

References

[1] J. Fitzgerald, S. Parastatidis, A. Romanovsky and P. Watson, Dependability-explicit Computing in Service-oriented Architecture. DSN 2004 Supplemental Volume. Florence, Italy, June 2004, pp.34-35.

[2] Managing Exceptions in Web Services Environments, An AmberPoint Whitepaper, AmberPoint, Inc. September 2003.

[3] B. Randell, System Structure for Software Fault Tolerance. IEEE TSE, 1, 1975, pp. 220-232.

[4] F. Tartanoglu, V. Issarny, A. Romanovsky, and N. Levy, Dependability in the Web Services Architecture. In: Architecting Dependable Systems, 2004, R. de Lemos, C. Gacek, A. Romanovsky, (eds.) LNCS 2677, pp. 90-109.

[5] M. Kalyanakrishnan, R. Iyer, and J. Patel, Reliability of Internet hosts: a case study from the end user's perspective, Computer Networks, 31, 10, 1999, pp. 47–57.

[6] J. Han and J. Frarnam, Impact of Path Diversity on Multi-homed and Overlay Networks, In Proceedings of the DSN-DCCS 2004, Florence, Italy, June 2004, pp.29-38.

[7] M. Merzbacher and D. Patterson, Measuring end-user availability on the web: Practical experience. In Proceedings of the International Performance and Dependability Symposium (IPDS), June 2002.

[8] J. Duraes and H. Maderia, Web-server availability from the end-user viewpoint: a comparative study, IEEE/IFIP International Conference on Dependable Systems and Networks - Dependable Computing and Communications, DSN-DCCS 2004, Florence, Italy, June 2004

[9] The Keynote Method, Keynote System, Inc, http://www.keynote.com/keynote_method/keynote_method_main_tpl.html, 2006

[10] A. Avizienis. The N-version Approach to Fault-Tolerant Software", IEEE TSE, 11, 1985, pp. 1491-1501.