NODES 09

NOrdic workshop and doctoral symposium on DEpendability and Security

Linköping, Sweden, April 27, 2009

Editors
Mikael Asplund, Simin Nadjm-Tehrani, and Luigia Petre

Linköping Electronic Conference Proceedings, 41
Linköping University Electronic Press
Linköping, Sweden, 2009

http://www.ep.liu.se/ecp/041/
ISSN 1650-3740 (online)
ISSN 1650-3686 (print)

Program Committee

Kaisa Sere, Åbo Akademi University, Finland
Simin Nadjm-Tehrani, Linköping University, Sweden
Christian Damsgaard Jensen, Technical University of Denmark
Ketil Stølen, SINTEF Research centre and University of Oslo, Norway
Jüri Vain, Tallinn University of Technology, Estonia

Organizers

Luigia Petre, Åbo Akademi University, Finland
Kaisa Sere, Åbo Akademi University, Finland
Simin Nadjm-Tehrani, Linköping University, Sweden

Table of Contents

Towards Dependable Placement of NoC Resources

Leonidas Tsiopoulos
Department of Information Technologies,
Åbo Akademi University,
Turku, Finland
leonidas.tsiopoulos@abo.fi

Abstract. In this paper we present an approach on how executable formal specifications of Network-on-Chip routing schemes can help on deciding efficient placement of processing resources on 3D-integrated systems. We use a routing scheme specified with the B Action Systems formalism and we execute it with the model checking and animating tool ProB in order to obtain traces of operation executions based on different data flow scenarios.

1 Introduction

Advances in technology allow us to have thousands of computational resources on a single electronic chip. Recently the Network–on–Chip (NoC) communication paradigm [6, 8] has been proposed as the alternative to bus–based Systems–on–Chip (SoC) communication paradigms. Bus–based communications offer insufficient bandwidth for concurrent on–chip data transactions whereas regular NoC interconnects can scale incrementally and data is routed in a distributed and flexible manner.

The number of resources which can be placed on a single chip increases rapidly with the continuous technology scaling resulting in growing wire delay and increased power consumption while meeting the constraints on interconnect latency is an issue. To overcome these challenges, 3D–integrated systems [7, 10] have been proposed during the last years with the main advantage of reduced length of the global interconnect which in turn reduces power consumption considerably. These complex systems are being used in many important applications, such as in automotive and advanced control systems, hence, it is important that they are reliable. Appropriate methods are needed in order to specify reliable 3D–integrated systems, as well as to model the communication and verify their design.

Formal methods of concurrent programming with adequate tool support are important for the development of complex 3D–integrated systems in order to avoid costly errors in the later design phases and contribute to the *dependability* of such systems. The B Action Systems [4, 13] is a state–based formalism

which was created in order to be able to reason about parallel and distributed systems, like Action Systems [3], within the B Method [1]. We use a hierarchical formal specification of an asynchronous NoC routing scheme [12] in order to show how executable traces of this formal model can help on deciding an efficient placement of communicating resources on a chip for reduced power consumption which in turn aids the reliability of the system.

The organization of this paper is as follows. In Section 2, we present a short introduction to the B Action Systems. In Section 3 we discuss the model of the NoC routing scheme considered for this paper. Section 4 discusses the trace generation procedure within ProB [9] which is a model checking and animating tool for B specifications. Section 5 discusses the related work. Finally, Section 6 concludes this paper.

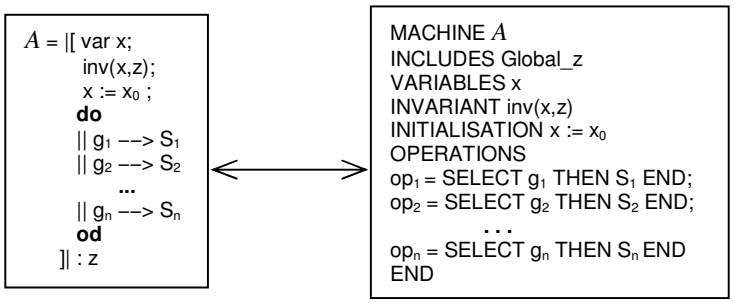
2 B Action Systems

B Action Systems [4, 13] is a state–based formalism based on Action Systems [ref] and the B Method [1] and was created in order to be able to reason about parallel and distributed systems, like Action Systems, within the B Method. Note that a model in B Action Systems is a valid model within the B Method; therefore, tool support for the B Method [5, 9] can be used to analyze models in B Action Systems.

The structure of an action system *A* is shown in Figure 1(a) in which z and x are respectively the global and local state variables. The action system interacts with the environment through the global variables. State variables have types and the set of possible assignments to them constitutes the state space. The statement $x := x_0$ assigns initial values to the local variables. Each action has the form g_i --> S_i, where $i = 1..n$, in which g_i is the guard and S_i is a statement on the state variables. An action is enabled only if its guard evaluates to true. As regards to the behaviour of an Action System [3], the initialization statement is executed first, and thereafter as long as there are enabled actions, one of the enabled actions is selected non–deterministically for execution. When there are no enabled actions, the system terminates.

An action system can be translated to a B abstract machine – an abstract machine is the basic unit of specification in B – as shown in Figure 1(b). This is an action system within B, and is called a B Action System. The system is identified by a unique name. The local variables of the system are given in the VARIABLES-clause. The INVARIANT-clause defines the types of the local variables and gives their guaranteed behaviour. Initial values are assigned to the local variables in the INITIALISATION-clause. The operations (or actions) in the OPERATIONS-clause are of the form op = SELECT g THEN S END, where g is

said to be the guard and S the body. The guard g is a predicate on the variables, and when g holds the action op is said to be enabled. Only enabled actions are considered for execution and if there are several actions enabled simultaneously they are selected for execution in a non-deterministic manner, analogous to the behaviour of an action system. B Action Systems can be composed to model parallel systems.



(a) (b)

Fig. 1. Structure of a B Action System.

Structuring mechanisms such as SEES, INCLUDES and PROMOTES can be used to express B Action Systems as a composition of subsystems [1]. The SEES-mechanism allows read access to the seeing system. The INCLUDES-mechanism allows write access to the variables of the included system. Actions of the included system can also be made available by promoting them into the including system within a PROMOTES-clause. The structuring mechanisms provide an efficient way to model system hierarchy.

Communication can occur between two B Action Systems in two ways: by use of global variables, or by global procedures; the example of this paper uses global procedures for communication purposes. Operations do not take any parameter and they execute autonomously; procedures may or may not have parameters, and they are to be invoked in a certain context. Consider the operation op_p = SELECT g_p THEN S_p || Proc END; this operation is enabled only if the procedure Proc is also enabled. The action and the procedure in its body are executed as an atomic entity.

3 The Executed Formal Specification

Tsiopoulos and Waldén [12] have specified a formal routing scheme relying on the request and acknowledge phases of the asynchronous communication. Within this asynchronous NoC routing scheme, data packets are received at the

input channels of the routers and they are distributed to their output channels for subsequent propagation to the neighboring routers. The routers acknowledge their input channels so that new data packets can be received. Controlling components of routers propagate the data packets to routers that have not received them yet, and prohibit the cycling of data packets to occur. This is because cycling of data packets back to routers which have received them already, reduces on-chip interconnect efficiency and increases power consumption. A hierarchical and compositional development method was used [12] for this purpose. Figure 2 outlines part of the hierarchical structure of the specification of the NoC routing scheme and presents one of the communications between the subsystems using global procedures.

Starting with the simplest subsystem of the routing framework, an asynchronous channel component called PushChannel (data propagation upon request) was specified as a B action system. The channel's inputs and outputs were modeled as global variables. Of these global variables, the necessary input and output asynchronous control handshakes were modeled with boolean variables named cstart and cend respectively, and the input and output data were captured with variables named dstart and dend of the generic type DATA. These variables together with simple procedures to update their state were defined in a separate machine named ChannelData which was included in PushChannel. The latter had two operations: one to propagate data and the request from its input to its output and the other to acknowledge its input after the output data was taken, so that new data and request could be received. The interface of PushChannel consists of two global procedures, ProcChangeCstartAndDstart and ProcChangeCendFalse; the first to update its input with a new request and data and the second to acknowledge its output after the transfer of the output data.

A B action system named Router including eight instances of PushChannel was then created. We note that twelve instances of PushChannel are needed to model propagation of data along the third dimension too. Four actions were specified for controlling the channels and propagating the data along them. For example, action TransferData_N copies the request value of the communication and the data from the output of the input northern channel inN to the inputs of the output channels, outE, outS, and outW so that the data can be propagated further towards the neighboring routers. Simultaneously it sends acknowledgment to the output of channel inN to indicate that Router is ready to receive new data via this channel. Note that six actions are needed to model distribution of data along the third dimension.

Finally, some instances of this router (000.Router, 010.Router, 020.Router, 030.Router, …; the digits correspond to the *xyz* coordinates on the network) were composed into a controlling system named NoC_Region1 (given a new name for

the purposes of this paper) which controlled the data distribution between the routers in a region of the NoC. In order to do so, some of the global interface procedures of PushChannel were promoted within Router to form its interface.

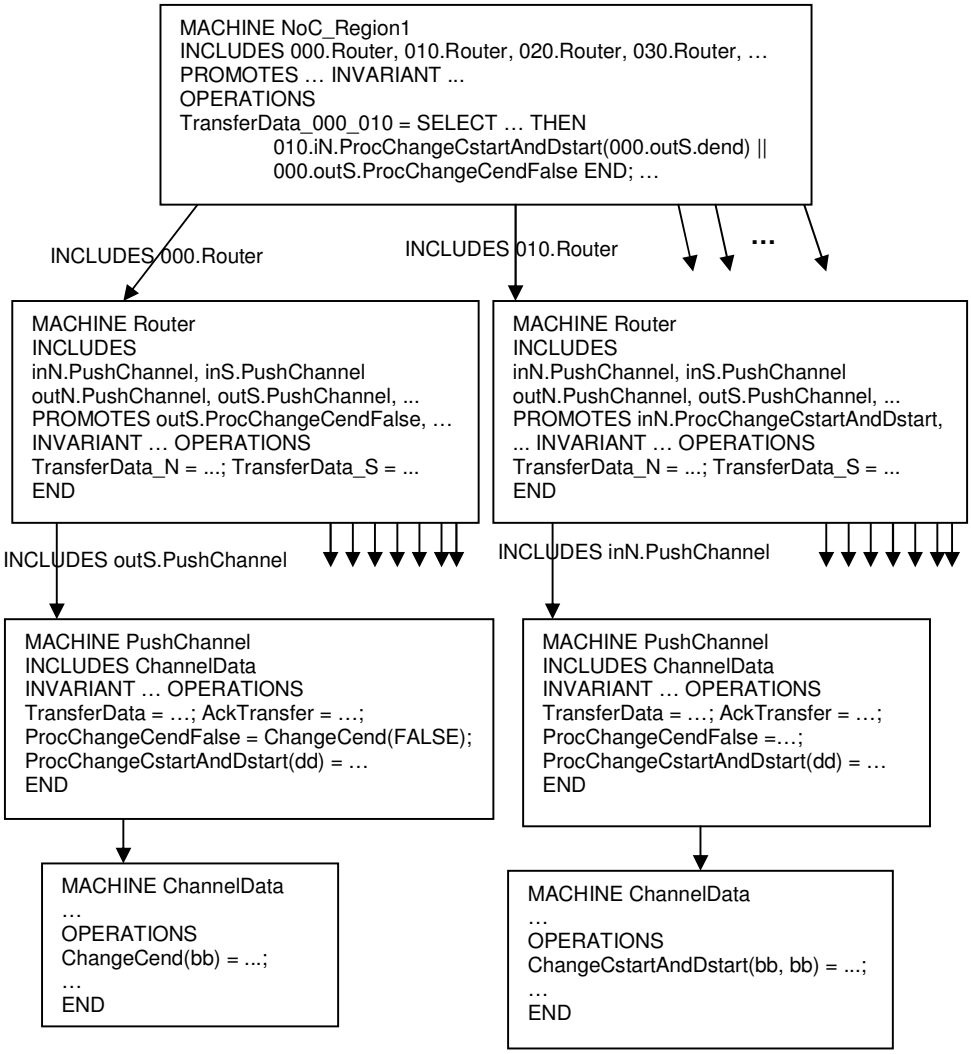


Fig. 2. Machine hierarchy in the NoC specification.

Refer to action TransferData_000_010 of machine NoC_Region1 in Figure 2. This action transfers data from the southern output channel of the router with

coordinates 000 to the northern input channel of router with coordinates 010. The guard checks whether channel 000.outS is ready to transmit and channel 010.inN is ready to receive. If so, the data value of channel 010.inN gets the data value of channel 000.outS. And the flags of both the channels are modified to indicate that the transmission has taken place. The global procedures ProcChangeCstartAndDstart and ProcChangeCendFalse defined in machine PushChannel do this modification. See in the figure, how these global procedures have been promoted to the interface of the router.

4 Trace Generation Procedure

ProB is a model checking and animation tool for B machines [9]. ProB includes a fully automatic animator written in SICStus prolog. ProB takes an *instantiated* model in B, in which any generic set has been instantiated with some concrete values to avoid *state explosion* and generates a finite coverage graph. Within this finite coverage graph, several traces of sequenced operation executions can be obtained. In other words, several different scenarios of operation execution paths for data propagation between resources on a chip can be created by the user.

Figure 3 shows part of one of the planes of a 3D–integrated system with a mesh topology corresponding to the formal routing specification presented in the previous section. Let us assume that a processing element at position 000 wants to send data to another element at position 750. One possible path for this communication is shown in Figure 3.

Each trace of operation executions in ProB originates from some initial state. Table 1 shows the trace of operation executions within the hierarchical B action systems specification corresponding to the communication path shown in Figure 3, including a scenario for possible delays. These delays occur because at the same points the needed operations for the data propagation are being executed for another trace of data communication intersecting the trace shown in Table 1. When these operations are enabled again, the execution of the operations in the path continuous until the data arrives at the destination.

Modelling such data propagation scenarios of formal NoC routing specifications can aid the process of finding a suitable placement of frequently communicating resources on a chip already at an early design level.

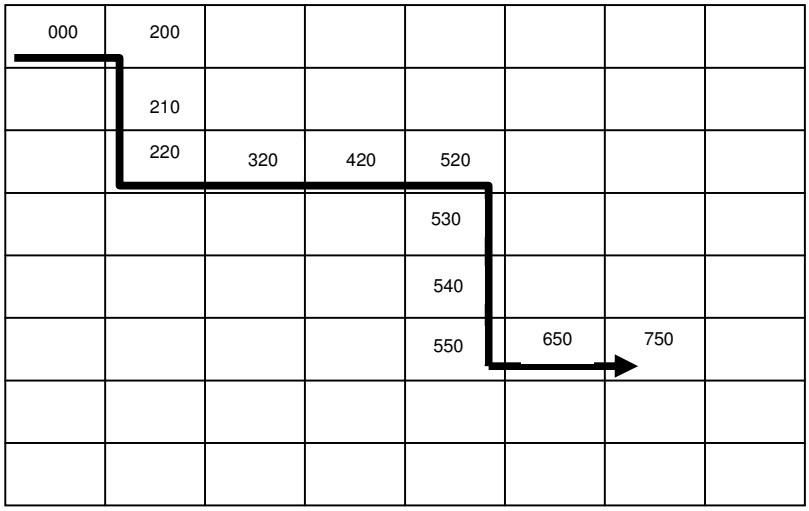


Fig. 3. A possible data propagation path from source 000 to destination 750.

000.localin.TransferData; 000.TransferData_local;
000.outE.TransferData; TransferData_000_200;
200.inW.TransferData; …**delay**; 200.TransferData_W;
200.outS.TransferData; …**delay**; TransferData_200_210;
210.inN.TransferData; …**delay**; 210.TransferData_N;
210.outS.TransferData; TransferData_210_220;
220.inN.TransferData; 220.TransferData_N; …**delay**;
220.outE.TransferData; TransferData_220_320;
320.inW.TransferData; 320.TransferData_W;
320.outE.TransferData; TransferData_320_420;
420.inW.TransferData; 420.TransferData_W;
420.outE.TransferData; …**delay**; TransferData_420_520;
520.inW.TransferData; 520.TransferData_W;
520.outS.TransferData; TransferData_520_530;
530.inN.TransferData; 530.TransferData_N;
530.outS.TransferData; TransferData_530_540;
540.inN.TransferData; 540.TransferData_N; …**delay**;
540.outS.TransferData; TransferData_540_550;
550.inN.TransferData; 550.TransferData_N; …**delay**;
550.outE.TransferData; TransferData_550_650;
650.inW.TransferData; 650.TransferData_W;
650.outE.TransferData; TransferData_650_750;
750.inW.TransferData; 750.localout.TransferData

Table 1. A trace of B operation invocations for data propagation.

5 Related Work

Several works have been presented during the last years on the mapping and placement of resources on 2D and 3D NoC designs. For example, Addo-Quaye [2] presented an approach for thermal and communication–aware mapping and placement for 3D NoC architectures. *Genetic algorithms* were used in order to model solutions to mapping problems. The approach was shown to reduce communication and peak temperature when compared to purely random placements.

Murali et al. [11] presented a method to determine a power efficient topology of a 3D System–on–Chip for a given application and for finding paths for the traffic flows that meet *Through Silicon Vias* constraints. The method accounts for power and delay of both switches and links. The assignment of resources to different 3D layers and the floorplan of the resources in each layer are taken as inputs to the synthesis process, while the output of the process is the optimal positions of NoC switches in each layer and between the layers.

To the best of our knowledge there has not been presented another approach to execute formal specifications of NoC routing schemes in order to aid the process of finding efficient placements of resources on a chip. We believe that our approach can enhance the design of complex 3D–integrated systems and work hand in hand with mapping and placement approaches such as the ones presented in this section.

6 Conclusion

Formal methods with adequate tool support are important for the design of complex NoC systems in order to correct errors in the early design phases and reduce the involved costs of NoC system design and development.

We proposed an approach for finding an efficient placement of resources on a NoC after *data flow* executions at the formal specification level within ProB. To the best of our knowledge it is the first approach on executing several different data flow scenarios within a formal NoC framework in order to aid the process of finding an efficient placement of resources on a chip. This development step is important in order to design reliable and dependable multidimensional integrated systems.

References

[1] Abrial J.–R. (1996). The B–Book, Cambridge University Press.
[2] Addo–Quaye, C. (2005). Thermal-aware mapping and placement for 3-D NoC designs. In Proc. IEEE Int. Syst.-on-Chip Conf., pp. 25–28.
[3] Back, R.J.R., Kurki–Suonio, R. (1983). Decentralization of Process Nets with Centralized Control, Proc. of the 2nd Symposium on Principles on Distributed Computing, pp. 131–142.
[4] Butler, M., Waldén, M. (1996). Distributed System Development in B, Proc. of the 1st conference on the B Method, Nantes, France, pp. 155–168.
[5] ClearSy, Atelier B, http://www.atelierb.societe.com/
[6] Dally, W. J. and Towles, B. (2001). Route packets, not wires: On–chip interconnection networks. In Proc. of the DAC'01, pp. 681–689.
[7] Gutmann, R. J., Lu, J. Q., Kwon, Y., McDonald, J. F., Cale, T. S. (2001). Three-dimensional (3D) ICs: a technology platform for integrated systems and opportunities for new polymeric adhesives. In Proc. Conf. Polymers Adhesives Microelectron. Photon. Pp. 173–180.
[8] Hemani, A., Jantch, A., Kumar, S., Postula, A., Öberg, J., Millberg, M., and Lindqvist, D. (2000). Network on a Chip: An architecture for billion transistor era. In Proc. of the IEEE NorChip Conference.
[9] Leuschel, M., Butler M. (2005). ProB: A Model Checker for B, Proc. FME'03, LNCS Volume 2805, Springer, pp. 855–874.
[10] Li, F., Nicopoulos, C., Richardson, T., Xie, Y., Narayanan, V., Kandemir, M. (2006). Design and Management of 3D Chip Multiprocessors Using Network-in-Memory. ACM SIGARCH Computer Architecture News. Volume 34 , Issue 2. pp. 130 – 141.
[11] Murali, S., Seiculescu, C., Benini, L., De Micheli, G. (2009). Synthesis of Networks on Chips for 3D Systems on Chips. Proceedings of the 2009 Conference on Asia and South Pacific Design Automation. pp. 242-247.
[12] Tsiopoulos, L., Waldén, M. (2006). Formal Development of NoC Systems in B, Nordic Journal of Computing, Vol. 13 (2006). pp. 127–145.
[13] Waldén, M., Sere, K. (1998). Reasoning about Action Systems using the B Method, Formal methods in System Design, Vol. 13(1), pp. 5–35.

Reliability Assessment in Event-B Development

Anton Tarasyuk, Elena Troubitsyna, and Linas Laibinis

Åbo Akademi University, Finland
{anton.tarasyuk, elena.troubitsyna, linas.laibinis}@abo.fi

Abstract. Formal methods are indispensable for ensuring dependability of complex software-intensive systems. In particular, the B Method and its recent extension Event B have been successfully used in the development of several complex safety-critical systems. However, they are currently not supporting quantitative assessment of dependability attributes that is often required for certifying safety-critical systems.. In this paper we demonstrate by example how to integrate reliability assessment into Event B development. This work shows how to conduct probabilistic assessment of system reliability at the development stage rather than at the implementation level.
Keywords: Event-based modeling, reliability assessment, formal verification, Markov processes

Introduction

To demonstrate dependability of a system it is often required to prove statistically that the probability of a catastrophic failure is acceptably low. Usually such an assessment is done after the development is finished, i.e., at the implementation level. However, in some cases the obtain design does not achieve the targeted level of reliability. Obviously, the cost of redesign might be very high.

In this paper we address the problem of reliability assessment at the development stage. We show how to integrate probabilistic assessment of system reliability with its formal modeling and verification in Event B. We demonstrate by example that in some cases we can obtain an algebraic solution connecting the overall system reliability with reliabilities of its components. In the other cases, we can employ probabilistic model checking to obtain a numeric solution.

Our ideas are exemplified via a small case study modeling and assessment of reliability of a data transmission. We believe that integrating probabilistic reasoning into the formal modeling significantly enhances the benefits of the later one. Indeed in this case our formal model would give us not only logical but also statistical evidences of system dependability.

1 Modelling and Refinement in Event B

Event B [1] is an extension of the B Method [2] to model parallel, distributed and reactive systems. The Rodin platform [3] provides automated tool support for modelling and verification in Event B.

Event B uses the Abstract Machine Notation for constructing and verifying models. An abstract machine encapsulates a state (the variables) of the model and provides operations on its state. A simple abstract machine has the following general form:

SYSTEM AM
VARIABLES v
INVARIANT I
INITIALISATION *INIT*
EVENTS
E_1
...
E_N

The machine is uniquely identified by its name *AM*. The state variables of the machine, *v*, are declared in the **VARIABLES** clause and initialised in *INIT* as defined in the **INITIALISATION** clause. The variables are strongly typed by constraining predicates of the machine invariant *I* given in the **INVARIANT** clause. The invariant is usually defined as a conjunction of the constraining predicates and the predicates defining the properties of the system that should be preserved during system execution.

The dynamic behaviour of the system is defined by the set of atomic events specified in the **EVENTS** clause. An event is defined as follows:

$$E = \textbf{WHEN } g \textbf{ THEN } S \textbf{ END}$$

where the guard g is conjunction of predicates over the state variables *v*, and the action *S* is an assignment to the state variables.

The occurrence of events represents the observable behaviour of the system. The guard defines the conditions under which the action can be executed, i.e., when the event is *enabled*. The action can be either a deterministic assignment to the state variables or a non-deterministic assignment from a given set or an assignment according to a given postcondition. These assignments are denoted as $:=, :\in$ and $:|$ correspondingly. If several events are enabled then any of them can be chosen for execution non-deterministically. If none of the events is enabled then the system deadlocks.

The Event B models are formally defined using the weakest precondition semantics [4]. The weakest precondition semantics provides us with a foundation for establishing correctness of specifications and verifying refinements between them. For instance, we verify correctness of a specification by proving that its initialization and all events establish the invariant.

The basic idea underlying formal stepwise development by refinement is to design the system implementation gradually, by a number of correctness preserving steps, called *refinements*. The refinement process starts from creating an abstract, albeit unimplementable, specification and finishes with generating executable code. The intermediate stages yield the specifications containing a mixture of abstract mathematical constructs and executable programming artifacts.

Assume that the refinement machine AM' is a result of refinement of the abstract machine *AM*:

SYSTEM AM'
VARIABLES v'
INVARIANT I'
INITIALISATION *INIT*$'$
EVENTS
E_1
$\ldots$
E_N

The machine AM' might contain new variables as well as replace the abstract data structures of *AM* with the concrete ones. The invariant of AM' – I' – defines not only the invariant properties of the refined model, but also the connection between the state spaces of AM and AM'. For a refinement step to be valid, every possible execution of the refined machine must correspond (via I') to some execution of the abstract machine. To demonstrate this, we should prove that $INIT'$ is a valid refinement of *INIT*, each event of AM' is a valid refinement of its counterpart in *AM* and that the refined specification does not introduce additional deadlocks, i.e.,

$$
\begin{array}{l}
wp(INIT', \neg wp(INIT, \neg I')) \;=\; true, \\
I \wedge I' \wedge g_i' \;\Rightarrow\; g_i \wedge wp(S', \neg wp(S, \neg InvC)), \quad \text{and} \\
I \wedge I' \wedge g_i \;\Rightarrow\; \bigvee_i^N g_i'
\end{array}
$$

2 Example of refinement in Event B

To illustrate modelling and refinement in Event B let us consider a small case study – modelling a simple data channel. The structure of the system is shown in Figure 1. The system comprises two nodes: a sender (A) and a receiver (B), and at each step A can send a request to B. After sending a message A stops sending and listens to the channel to get confirmation from receiver. The communication channel (C) can fail at each step, the failure is repairable and both failure and repair rates are assumed to be known from some reliability report. If a request (confirm) is sent while the channel remains unavailable, the message got lost and the system is shutting down. If the message is confirmed, A can transmit the next request.

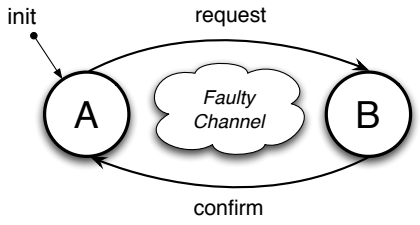


Fig. 1. Case study: overall system structure

In our initial abstract specification we merely model the possibility of successful or failed transmission. The variable res is introduced to represent this. The variable is non-deterministically assigned the values $TRUE$ or $FALSE$. When res obtains the value $FALSE$ then the failure has occurred and the system deadlocks.

MACHINE Channel0
VARIABLES
res
INVARIANTS
inv1 : $res \in BOOL$
EVENTS
Initialisation
begin
act1 : $res := TRUE$
end
Event *evt1* $\widehat{=}$
when
grd1 : $res = TRUE$
then
act1 : $res :\in BOOL$
end
END

Our initial specification is deliberately simple to facilitate probabilistic modelling that we will discuss in the next section. Next we will show how to refine this specification to capture the entire set of requirements given above.

We introduce the variable msg to model the status of message being transmitted, the variable ch to model the status of the transmitting channel and the variable $flag$ to model the desired sequence of events. This is a rather simple refinement step. The specification of it is given below.

MACHINE Channel2
REFINES Channel1
SEES Context0
VARIABLES
`res`
`msg`
`flag`
`ch`
INVARIANTS
`inv1` : $flag \in 0 \mathinner{..} 1$
`inv2` : $ch \in BOOL$
EVENTS
Initialisation
extended
begin
`act1` : `res` $:=$ `TRUE`
`act2` : `msg` $:=$ `A`
`act3` : $flag := 0$
`act4` : $ch := TRUE$
end
Event *evt1* $\widehat{=}$
refines *evt1*
when
`grd1` : $res = TRUE$
`grd2` : $msg = ok$
`grd3` : $flag = 1$
then
`act1` : $msg := A$
`act2` : $flag := 0$
end
Event *evt2* $\widehat{=}$
refines *evt2*
when
`grd1` : $res = TRUE$
`grd2` : $msg = lost$
`grd3` : $flag = 1$
then
`act1` : $res := FALSE$
end
Event *evt3_1* $\widehat{=}$
refines *evt3*
when
`grd1` : $res = TRUE$
`grd2` : $msg = A$
`grd3` : $flag = 1$
`grd4` : $ch = TRUE$
then
`act1` : $msg :\in \{A, B\}$
`act2` : $flag := 0$
end
Event *evt3_2* $\widehat{=}$
refines *evt3*
when
`grd1` : $res = TRUE$
`grd2` : $msg = A$
`grd3` : $flag = 1$
`grd4` : $ch = FALSE$
then
`act1` : $msg :\in \{A, lost\}$
`act2` : $flag := 0$
end
Event *evt4_1* $\widehat{=}$
refines *evt4*
when
`grd1` : $res = TRUE$
`grd2` : $msg = B$
`grd3` : $flag = 1$
`grd4` : $ch = TRUE$
then
`act1` : $msg :\in \{B, ok\}$
`act2` : $flag := 0$
end
Event *evt4_2* $\widehat{=}$
refines *evt4*
when
`grd1` : $res = TRUE$
`grd2` : $msg = B$
`grd3` : $flag = 1$
`grd4` : $ch = FALSE$
then
`act1` : $msg :\in \{B, lost\}$
`act2` : $flag := 0$
end
Event *evt5* $\widehat{=}$
when
`grd1` : $flag = 0$
then
`act1` : $ch :\in BOOL$
end
END

Next we will show how to integrate probabilistic reasoning into Event B modelling.

3 Probabilistic modelling in Event B

Automatic tool support is essential for ensuring scalability of formal modelling. Hence availability of the tool support is essential when choosing the probabilistic basis underlying integration of probabilities into Event B. Maturity and good usability of probabilistic model checking using PRISM tool [5] encouraged us to choose discrete-time Markov chains (DTMCs) and Markov decision processes (MDPs) as the underlying semantics for our probabilistic enhancement of Event B modelling. In mathematics, a Markov chain is a stochastic process having the Markov property, it means that the description of the present state of the process fully captures all the information that could influence the future evolution of the process. Future states will be reached through a probabilistic process instead of a deterministic one, i.e. according to a certain probability distribution. Any Markov chain is completely defined by its matrix of transition probabilities and initial distribution. Markov decision processes are an extension of Markov chains, the difference is the addition of actions and rewards structures, in some cases MDP could be reduced to a Markov chain. More formal definitions of these of Markov processes can be found in many books on probability theory, e.g. [6–8].

Probabilistic model checking is an automatic formal verification technique for analysing quantitative properties of systems which exhibit stochastic behaviour. PRISM is a probabilistic model checker, a tool for formal modelling and analysis of systems which exhibit random or probabilistic behaviour. It supports three types of probabilistic models: discrete-time Markov chains, continuous-time Markov chains and Markov decision processes, plus extensions of these models with costs and rewards [5]. A system specification in PRISM is constructed as a parallel composition of modules, which can interact with each other. In general, a module in PRISM looks as follows:

module *Module_name*

$var : Type$ **init** $\ldots$;

$[]\quad grd1 \rightarrow p_1 : action_1 + \cdots + p_n : action_n;$
$[]\quad grd2 \rightarrow q_1 : action'_1 + \cdots + q_m : action'_m;$
$\ldots$

endmodule

As it is easy to see, Event B models can be easily augmented with the required probabilistic information and analysed using PRISM model checker. In the next section we will show how assess reliability of a system modeled in Event B using PRISM model checker as well as using mathematical theory of Markov processes.

4 Reliability Assessment in Event B Modelling

In engineering, reliability [9, 10] is generally measured by the probability that an entity $\mathcal{E}$ can perform a required function under given conditions for the time interval $[0, t]$:

$$R(t) = P[\mathcal{E} \textit{ not failed over time } [0, t]].$$

Let T be the random variable measuring the uptime of the entity:

$$R(t) = P[T > t] = 1 - P[T \leq t] = 1 - F(t).$$

There is a strong correlation between Event-B specifications and discrete time Markov processes. Lets consider some Event-B model where only local nondeterminism is possible, i.e. at every moment of time one and only one event might be enabled. In this case at every moment of time actions of the (single) enabled event define a number of possible future states of the system. As was mentioned before, only two types of actions are presented in Event-B: deterministic and non-deterministic assignments. Deterministic assignment defines the future system state unambiguously, in other words, it defines state transition with probability 1. Non-deterministic assignment defines a set of possible states, but the choice between them has daemonic nature. Replacing this non-deterministic choice with some probability distribution we obtain a set of possible probabilistic transitions similar to the onex in a discrete Markov chain. The replacement of such type is always valid because for any programs $prog$ and $prog'$: $prog \sqcap prog' \sqsubseteq prog_p \oplus prog'$, $\forall\, p \in [0, 1]$ [12]. If the global non-determinism is also possible in a Event-B model then it can be represented by Markov decision process in a similar way.

Now let us return to our case study – modelling simple data channel. Let us assume that the transfer rate of A and the service rate of B are known.

The concrete specification consists of two group of events, the first group models the channel's availability and the second one models the message transferring process. These group of events can be represented by two Markov chains, but while the transferring process depends on the channel's availability, the corresponding Markov chain are not independent and reliability analysis of such system can be complex

The probabilistic model checking PRISM can be used to assess reliability of our channel. The results of the analysis using PRISM are given below.

However, model checking does not give an analytical representation of reliability function. This might be disadvantageous, e.g., it does not give a guidance on how to choose components that allow to achieve the desired system reliability.

Lets consider another approach based on mathematical theory of Markov chains. One of the possible approaches to work with two or more non-independent Markov chains is to try to define the correlation between them. For instance, the concrete specification of our case study can be represented by Markov chain (see Figure 2.) with transition probabilities depending on channel's availability. The considered example is very simple and this is not a problem to derive an

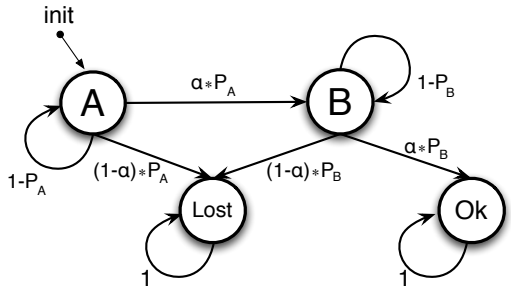


Fig. 2. State transition diagram for the time non-homogeneous chain

analytical view for availability function α, namely $\alpha(k) = \frac{P_R+P_F\cdot(1-P_R-P_F)^k}{P_R+P_F}$. But still, as can be seen, α depends on time k and obtained Markov chain is time non-homogeneous, and analysis of such type of chains is not a good idea.

Another way to work with a number of non-independent Markov chains is to try to build their superposition [6]. This is a relatively easy task when all chains are regular, but in our case study the chain describing the message transferring process is absorbing. It is clear, that to describe two processes as a superposition we need to define a state space of an output process and find its transition matrix P. For our example the superposition can be build by decomposing of two states A and B into four states A_1, A_2, B_1 and B_2, where A_1 represents a system state when the message is in A and the channel is available, A_2 – the message is in A and the channel is unavailable and so on. The corresponding state transition diagram is shown in Figure 3.

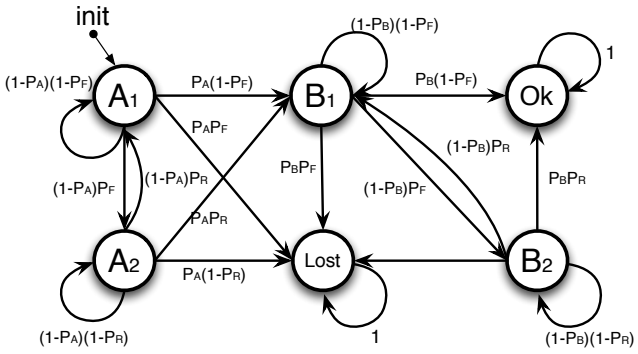


Fig. 3. State transition diagram for the time homogeneous chain

Now, when we know the transition matrix P, we can calculate the fundamental matrix $N = (I - Q)^{-1}$, where I is an identity matrix and Q represents the transitions between transient states. We can calculate also the matrix $B = \{b_{ij}\} = N \cdot R$, where R represents the transitions from transient to ergodic states and b_{ij} is the probability that the process starting in transient state s_i ends up in absorbing state s_j [7]. Let β be the probability of absorption in *"Ok"* state

and from matrix B we can find that

$$\beta = \frac{(P_A + P_R - P_A \cdot P_F - P_A \cdot P_R) \cdot (P_B + P_R - P_B \cdot P_F - P_B \cdot P_R)}{(P_A + P_F + P_R - P_A \cdot P_F - P_A \cdot P_R) \cdot (P_B + P_F + P_R - P_B \cdot P_F - P_B \cdot P_R)}$$

In our example the reliability function can be represented in terms of the total uptime of the system or in terms of a number of successful request-confirm transfers within given time interval. The last one is better in respect to reliability engineering since it gives more concrete information about the system functioning and omits the standby periods when failures are impossible. On the other hand, both representations are connected in a trivial way: as long as $\{A_1, A_2\}$ and $\{B_1, B_2\}$ comprise two non-communicable classes of transient states (process cannot goes back from B to A), the mean number of times the process remains in non-absorbing states equals to the sum of the time periods the process spends in these two classes, namely $\frac{1}{P_A+P_B}$. Let X be the random variable measuring the number of successful transfers, obviously it has a geometric distribution, and the mean of X and the reliability function of the system are $E(X) = \frac{\beta}{1-\beta}$ and $R(t) = \beta^{t+1}$ respectively.

It is interesting to compare results obtained with probabilistic model checking to those obtained analytically. Let us consider the input vector of probabilistic characteristics $(P_A, P_B, P_F, P_R) = (0.5, 0.8, 0.01, 0.5)$, then $\beta = 0.9758$ and $E(X) = 40.3485$. The results of the analysis using PRISM are illustrated in Figure 4 and they are closely related to the analytical ones: the graph shows system unreliability in terms of the total uptime and the "Property details" window demonstrates the total expected number of sent requests, i.e. $E(X) + 1$.

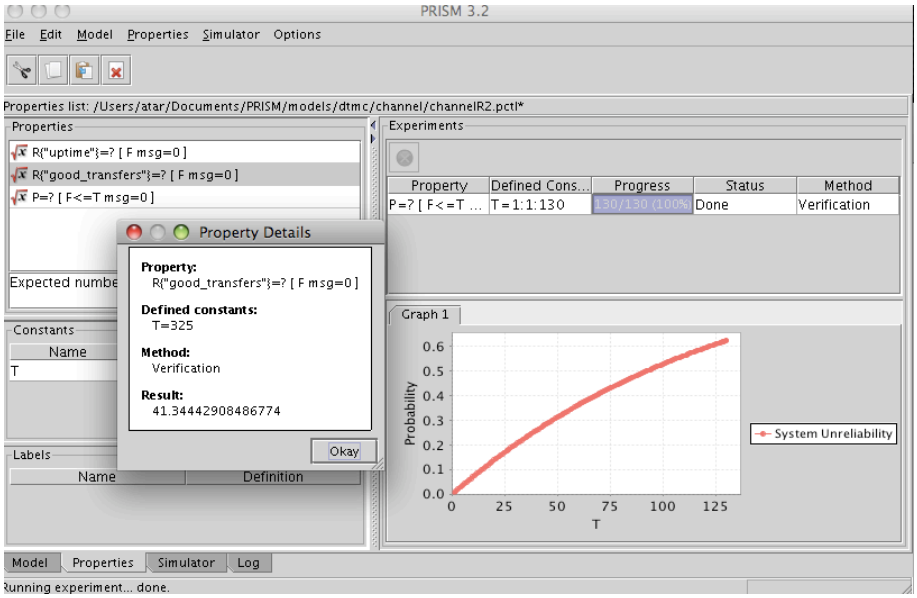


Fig. 4. Formal verification with PRISM

Conclusion

In this paper we have shown an example of integrating formal development by refinement with probabilistic assessment of system reliability. We demonstrated that for rather small specifications we can obtain an algebraic solution expressing overall system reliability as a function of reliabilities of its components. However, we have also demonstrated that for complex systems we can obtain a numerical estimate of reliability using PRISM model checker. Our approach supports reliability assessment already at the development phase and can give guidance on optimizing the design from dependability point of view. Moreover, it can help us to early diagnose the problems of the chosen design, so that the desired level of dependability would be nevertheless achieved. The similar topic in the context of refinement calculus has been explored previously [11, 12]. However, we see a great benefit in integrating probabilistic reasoning into the framework that has a mature tool support [3]. As our future work it would be interesting to further explore the connection between Event B modeling and dependability assessment. In particular the topic of probabilistic data refinement seems to be promising.

References

1. J.-R. Abrial. Extending B without changing it (for developing distributed systems). In H. Habiras editor, *First Conference on the B method,* pages 169-190. IRIN Institut de recherche en informatique de Nantes, 1996.
2. J.-R. Abrial. *The B-Book: Assigning Programs to Meanings.* Cambridge University Press, 2005.
3. RODIN Event-B Platform http://www.event-b.org/
4. E. W. Dijkstra. *A Discipline of Programming.* Prentice-Hall, 1976.
5. PRISM probabilistic model checker http://www.prismmodelchecker.org/
6. W. Feller. *An Introduction to Probability Theory and Its Applications.* John Wiley & Sons, 1967.
7. J. G. Kemeny, J. L. Snell. *Finite Markov Chains.* D. Van Nostrand Company, 1960.
8. D. J. White. *Markov Decision Processes.* John Wiley & Sons, 1993.
9. A. Villemeur. *Reliability, Availability, Maintainability and Safety Assessment.* John Wiley & Sons, 1991.
10. P. O'Connor. *Practical Reliability Engineering.* John Wiley & Sons, 1995.
11. A. K. McIver, C. C. Morgan, and E. Troubitsyna The probabilistic steam boiler: a case study in probabilistic data refinement. In J. Grundy, M. Schwenke, and T. Vickers, editors, *Proc. International Refinement Workshop, ANU, Canberra,* Discrete Mathematics and Computer Science, pages 250-265. Springer-Verlag, 1998.
12. A. K. McIver, C. C. Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems.* Springer, 2005.

Contract-based design in controller development and its evaluation

Pontus Boström[1], Marta Pląska[1], Mikko Huova[2], Matti Linjama[2], Mikko Heikkilä[2], Kaisa Sere[1], Marina Waldén[1]

[1]Department of Information Technology
Åbo Akademi University
Joukahaisenkatu 3-5
FIN-20520 Turku
Fax: +358 2 215 4732
{pontus.bostrom, marta.plaska, kaisa.sere, marina.walden}@abo.fi

[2] Department of Intelligent Hydraulics and Automation (IHA)
P.O. Box 589
FIN-33101 Tampere
Fax: +358 3 3115 2240
{mikko.huova, matti.linjama, mikko.heikkila}@tut.fi

1. Introduction

Software is present in many application areas and is becoming an integral part of mechanical appliances. Therefore cost-effective development of reliable control software has become more critical than ever. In this paper we describe a lightweight formal approach for control systems development and an application of the method to the development of a part of a controller for a digital hydraulic system. We simultaneously evaluate the impact of the used methodology on the development and on the final system. Digital hydraulics involve advanced controllers to achieve high performance and to realize sophisticated characteristics, such as energy saving. This affects the software of the hydraulics controllers, which is therefore highly complex when compared to more traditional controllers. Therefore adequate software development techniques have to be used in order to manage the complexity and to ensure the reliability and functional correctness of the developed system.

Model-based design has become a popular method for development of embedded control software. In this design method, a simulation model of the controlled system is also constructed. This enables fast and cheap analysis of the performance and correctness of the controller. Simulink is one of the most popular tools for model-based design at the moment. This is due to its user-friendly graphical modelling language, simulation tools and its large library of ready-made components. However, Simulink lacks tools and concepts for stepwise design and modular techniques to analyse system correctness. These features would be desirable to better reason about the correctness of the constructed system in a more manageable way and thereby increase the reliability of the system.

To enable stepwise design and modular analysis of correctness, we have introduced contract-based design in Simulink [1] [2]. Contracts here refer to pre- and post-conditions for programs that describe the assumptions the program makes about its environment and the results it can compute. Contract-based design is a well-known technique for object-oriented software development [3]. Our aim has been to transfer those principles to Simulink. To validate that Simulink models satisfy their contracts, we have developed formal analysis techniques based on action systems [1] [4] [2]. Formal reasoning about Simulink models has also been investigated in several notations before. There is a translation to Lustre [5] that can handle a large subset of Simulink. Another formalisation is a translation to Circus [6]. This translation can also handle a relatively large subset of Simulink. The focus is there on analysis of models and refinement of models into possibly parallel code. Neither of those works considers contracts as an integrated part of the formalisation. However, contracts could be introduced there also.

This paper describes a practical application of contracts on the development of a part of a digital hydraulics controller using Simulink. Contracts are used as an aid for structuring the system and for analysis of the system in the form of testing and design reviews. The case study demonstrates positive impact of using contracts on the quality of the resulting control software. Only an overview of the development method and application of it to a case study is presented here. A complete paper that gives a more thorough presentation of the topic has been submitted to a journal [7].

To ensure the quality of the constructed system, quality measurements should be performed. Quality measurements should be, and often already are, a part of the development cycle [8]. Since control systems are becoming more software intensive nowadays, we study and evaluate the impact of lightweight formal methodology in perspective of quality of final product and its development process. We also research the system's complexity characteristic, as it influences the schedule, costs and risks of the development, as well as

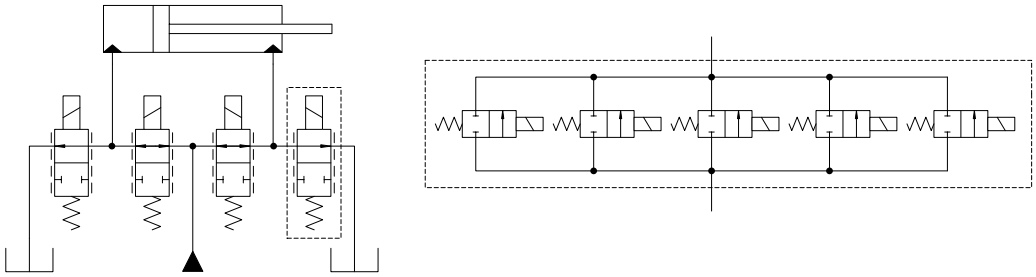
system's understandability [9]. Additionally, complexity has an impact on reliability and cost management of the software process [10].

The paper first shortly describes the application area of the case study, i.e. digital hydraulics. Then a presentation of Simulink and contracts is given. The development steps carried out to create the software for the system in the case study are then presented. Subsequently, evaluation of the development process and the software quality, with a special focus on the complexity of the system is given. Finally, we conclude and give directions for our future work.

2. Application Domain and Case Study Description

The evaluation of applicability of our development method was done using a case study from the area of digital hydraulics. Digital hydraulics is a rather new area of fluid power, which aims at reducing the complexity of the mechanical parts by improving control algorithms. One of the objectives of digital hydraulics is the cost-effectiveness and ease of manufacturing of these parts. However, a sophisticated controller is needed to obtain good performance of the system. Therefore, the need for high quality control software in digital hydraulic systems arose.

Hydraulic systems are widely used in industrial and mobile applications which require high forces or high power to weight ratio. One typical application of digital hydraulics is hydraulic cylinder control, which traditionally have been achieved using complex analogue proportional and servo valves. Digital hydraulics makes use of simple on/off-valves connected in parallel to achieve similar or better performance [11]. Figure 1 illustrates a digital hydraulic system. The figure (b) shows a Digital Flow Control Unit (DFCU) with five on/off-valves connected in parallel. The figure (a) shows digital hydraulic valve system with four DFCUs for controlling a hydraulic cylinder. Digital hydraulic valve systems can produce different control modes and thereby save energy, which is an important benefit as the efficiency of traditional hydraulic systems is usually considered to be low [12]. The digital hydraulic system also has high fault tolerance when compared to traditional systems, which use only one valve to control each cylinder [13]. To achieve fault tolerance, suitable control algorithm has to be used together with some method to detect faulty valves.



(a) (b)

Figure 1. Cylinder drive with a 4- DFCU digital hydraulic valve system (a), as well as a hydraulic diagram of a DFCU consisting of 5 on/off-valves (b).

A relatively complex model-based control algorithm is used in order to be able to utilize all features of the digital hydraulic system. The objective of the controller is to find the optimal configuration of opened and closed valves. The controller uses a steady-state model of the valves and cylinder to be able to estimate the behaviour that would eventually result from the different valve control combinations. There are usually a large number of possible combinations that should be analyzed to find the optimal control solution. For example, a valve system with twenty valves has over one million different control combinations. Several techniques are used to obtain an optimal control configuration using only relatively light-weight calculations that do not require high performance computing hardware.

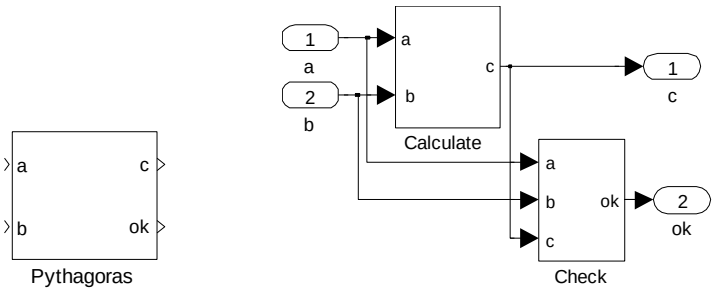
The correct operation of the controller is critical because hydraulic systems are usually heavily loaded and hazardous if malfunctioning. The control algorithms have previously been developed in Simulink without using a structured development method. However, there were problems with software quality and maintainability.

3. Overview of Simulink and contract-based design

Simulink is a part of the MATLAB environment developed by Mathworks Inc. It is a graphical language, where behaviour is described with dataflow diagrams. A diagram contains *functional blocks* that describe how data is manipulated. These blocks are also often *parameterised* to make them more flexible. Blocks can also contain memory, which means that the output of these blocks depends also on this internal state. The blocks are connected by *signals* that describe how data flows between them. The connection points for signals in the blocks are called *ports*. A diagram can be hierarchical, where the functional blocks are organised into *subsystem blocks*. A subsystem block can contain any number of ports for input and output of data to and from the subsystem. An example of a Simulink diagram is given in Figure 2 (b). This diagram contains two subsystem blocks: *Calculate* and *Check*. The blocks with rounded corners correspond to the ports of the subsystem they are in. They enable communication over the subsystem boundary.

The subsystem block in Figure 2 (a) contains a subsystem to calculate the length of the hypotenuse in a triangle according to Pythagoras theorem, $c = \sqrt{a^2 + b^2}$. In Figure 2 (b), the subsystem *Calculate* calculates the result *c*, while the subsystem *Check* checks if the calculated solution is accurate enough. If the solution is good enough then *ok* is set to *true* otherwise *ok* is set to *false*. The content of subsystems *Calculate* and *Check* is shown in Figure 3 (a) and (b), respectively.

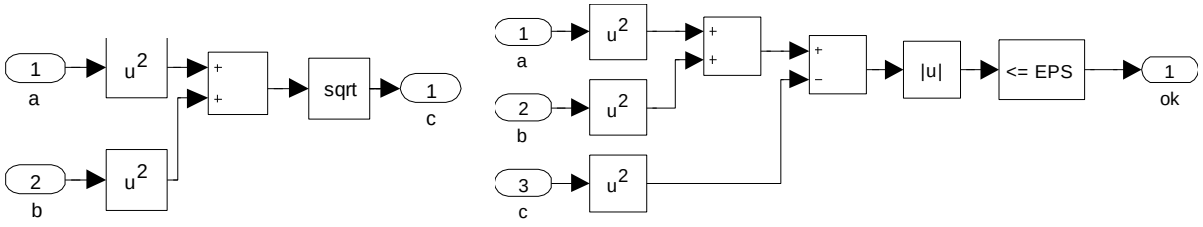
The idea of contracts from object-oriented systems can be applied to Simulink diagrams. The unit that is described by contracts is here the subsystem block. The idea with contracts is that they give a more high-level description of the subsystem block behaviour than the subsystem diagram. This enables reasoning about diagrams in a more modular fashion. The correctness of a diagram can be analysed in terms of the contracts of the subsystem blocks that it contains. The detailed content of the subsystems does not have to be known. The contracts also document the division of responsibility between subsystems.



(a) (b)

Figure 2. Subsystem that uses the Pythagoras theorem to compute the length of the hypotenuse (a), as well as the content of that subsystem (b)

Each subsystem can be described by a pre-condition that states what it assumes of the values on its in-ports and a post-condition that states which values can be produced on its out-ports. Assume we have a subsystem M_s with a list of in-ports p_i, list of out-ports p_o, as well as block parameters c. The pre-condition is a predicate of the form $Q^{pre}(p_i,c)$. The post-condition is of the form $Q^{post}(p_o,p_i,c)$. Often a subsystem or a diagram will not function correctly with arbitrary block parameters. To describe the valid values of block parameters c, a predicate $Q^{param}(c)$ is used.



(a) (b)

Figure 3. The content of subsystem *Calculate* (a) and subsystem *Check* (b) in Figure 2

Table 1. Contracts of the subsystems presented in Figure 2 and Figure 3

Subsystem	pre-condition	post-condition
Pythagoras	$a \geq 0 \wedge b \geq 0$	$ok \Rightarrow \mid a^2 + b^2 - c^2 \mid \leq EPS$
Calculate	*true*	*true*
Check	*true*	$ok \Rightarrow \mid a^2 + b^2 - c^2 \mid \leq EPS$

Consider the subsystem for calculating the length of the hypotenuse in a triangle from Figures 2 and 3. The contracts for the different subsystems are given in Table 1. The subsystem *Pythagoras* also has one block parameter *EPS* that gives the desired precision of the computation. The contract for this block parameter is that *EPS* should be greater than zero, $Q^{param} \triangleq EPS \geq 0$. The complete contract of the subsystem *Pythagoras* states that the inputs *a* and *b* should be greater than zero. If this condition is satisfied the subsystem will ensure that the result has been calculated with desired precision if *ok* is set to true. The post-condition subsystem *Calculate* states that it will compute any value, since the precision of the calculation is not known, which is only modelled coarsely here. The subsystem *Check* then checks that the solution is accurate enough and sets the port *ok* accordingly. Note that here it is assumed that all ports have the type double. The contracts encode design decisions about division of responsibility in a clear way. The contract clearly states what inputs can be given to a subsystem and what outputs the subsystem will produce in response to those inputs.

To fully benefit from contracts it should be possible to prove that a subsystem always satisfies its contract. Furthermore, it should be possible to analyse *complete Simulink models*, i.e. subsystems that do not have any in-ports. In this case, we need to show that all of the subsystems in the model are connected in such a way that every pre-condition and post-condition are satisfied. To achieve these goals, we have analysed [2] [4] the Simulink diagrams using refinement calculus and action systems [14] [15] [16]. In the case study discussed in this paper, the validation has been carried out using testing. However, the testing was based on the rules of correctness given by the theory.

4. Development process

In this section we present an outline of the development process for a part of a controller for a digital hydraulics system. The developed subsystem is an important part of the controller and is used to evaluate the application of contract-based design in the MATLAB/Simulink environment [17]. The developed subsystem is complex enough to generate realistic data about the development method. The subsystem presented here is a re-engineered version of an old subsystem with the same basic functionality, but it is also extended with several new features.

The design process started from gathering the requirements of the subsystem. First, the subsystem has to be able to choose the correct control solution according to several criteria that different applications of the valve system has, e.g., find balance between accuracy of velocity and pressure tracking, reduce unnecessary switching of the valves and to minimise the energy consumption. The re-engineering was also aiming at making the subsystem suitable for a broad range of applications by increasing the configurability, as well as making the user interface easy to use. Since the functionality of the controller was extended, the system became more complex.

After gathering the requirements, a first high-level specification of the subsystem was created. It consisted of parameters, inputs and outputs together with contracts that described their properties. Requirements for the parameters and inputs were defined using parameter- and pre-conditions. The functionality of the application was guaranteed using post-conditions for the outputs.

After the high-level specification of the subsystem was completed, it was then refined into a subsystem diagram where the functionality was decomposed into five subsystems. Each of these subsystems was also described by contracts, which were written following the same pattern as in the high-level specification. The contracts were used as a basis for allocating functionality to each subsystem and to describe the assumptions made for each of them. Here contracts were used to analyse that the diagram indeed satisfies the contract of the high-level specification via design reviews.

During the design process the design reviews were performed multiple times by a team of developers. At this point contracts gave a good basis for discussing and analysing the design at an early development stage. Moreover, they helped to gain confidence that the specification was correct and complete, as it was done accurately using contracts. This in turn influenced the quality of the final control algorithm, as shown in Section 5 in more detail.

After the refinement phase, each of the five subsystems was implemented according to the specification documents as Simulink diagrams. Afterwards, unit testing was performed for each subsystem, thus enabling testing activity before finalising the implementation phase. The unit tests checked that the subsystems conformed to their contracts. The pre-conditions described valid test-data and the post-conditions described the valid outputs of the subsystem. The outputs from a subsystem were checked after the execution of each test and the defects could be detected by checking if the post-conditions were violated. Simulation of the complete controller with a model of a mobile boom test system allowed evaluation of its performance. Testing and simulation were considered as sufficient for confirming the correctness of the subsystem. Proofs of correctness were omitted, as being too time and effort consuming considering the extra confidence in the system that would have been gained.

5. Evaluation of development

Quality measurements were done for the development process and for the constructed subsystem. The study of the quality of *final subsystem* was done with respect to defect-related metrics, size of the system and its structure, which all influence reliability. We also consider system's complexity, since it impacts its maintainability. Complexity is also a major factor when talking about testing coverage. The evaluation of the *development process* included the analysis of its distribution over time, as well as the examination of activities regarding the defect handling (defect removal). The aim is to investigate the impact and suitability of the applied methodology on system development. The data collection for our examination was done in parallel with the development, thus enabling thorough quality evaluation.

The number of defects in the system is considered as one of the most important aspects of quality. We have measured the number of defects and their distribution over the development phases with respect to their origin and removal phase, excluding the severity level classification. Here by defect (fault) we understand an anomaly in a product. In the developed subsystem eight defects in total were detected. The small defect density, equal to 0.55 defects per thousand of generated lines of code (kGLOC), was not only a result of methodology used, but additionally experienced developers, design reviews, code generation and relatively small system size (14428 GLOC in the C language, 846 corresponding Simulink blocks). For comparison, in an earlier development, where contract based design was only introduced as a new development approach, the defect density was equal to 1.38. A critical system, e.g. the air traffic control system investigated in [18], that uses formal methods and C language, has 1.25 defects per thousand of non-commented source code lines (NCkSLOC) after delivery.

The information about defect distribution is particularly interesting, as it shows the impact of the methodology on the development process and on the cost of handling defects. In our study the defects originated from the specification and refinement (3) phase and programming phase (5). The defect discovery was as follows, programming phase: 3, unit testing: 4, system testing: 1. It is worth noticing that only one defect was found at the last stage of the development. It is generally thought that the earlier a defect is found the cheaper it is to fix it [19]. Therefore the final result is very good, as there were no defects found after the system was deployed. However, the system has not been used extensively yet, therefore some defects might still be latent. The coverage of the tests was also not measured and therefore the effectiveness of the test cases in finding defects cannot be guaranteed.

The same part of the controller has been developed earlier using an unstructured development methods. The system developed in this paper has been compared to that system. We computed the *total complexity* as a sum of *structural* and *data complexity* for the developed subsystem [20]. We observed relatively high structural complexity (dependent on quantity of subsystems and connections between them). This was caused by structuring the system into numerous subsystems, thereby decomposing functionality, and simultaneously increasing the connections between subsystems.

There was also a minor increase in data complexity (dependent on number of input/output data and connections between the subsystems). This was caused by the fact that more functionality needed to be accomplished by the system. Moreover, there were more user selectable and fine-tuning related parameters, in order to make the system configurable, more usable and flexible. Additionally, there were many computational requirements due to relatively complex control algorithms.

More sophisticated control algorithm, additional parameters, structuring the system, increase of the system's size (54% in number of blocks compared to the previous version) and other system enhancements reflected in rather minor increase in total complexity of the subsystem. The developers' perception confirms the outcome of complexity analysis and attributes it to the use of contract-based design. In fact, although to some extent more complex, the system is more readable and understandable. It should also be stated that the developers were constantly extending their comprehension of the system. Therefore, the quality of the software should be considered as being influenced by increasing experience of the developers. The system is now more maintainable and reusable, as the lower layers can be modified without the necessity of altering the entire system and higher layers can be reused later on.

The contract-based design approach influences not only the quality of the final product but also the development process and its distribution over time. Figure 4 presents the distribution of development phases over the development cycle. We distinguish five development phases, i.e. specification, refinement(s), programming, unit and system testing. System design (specification and refinement phases) in the test application took majority of the development time (approximately 60% of the total development time of 94 man-hours). Since the focus was shifted to the proper design of the system, coding and testing phases were relatively short (35% in total). For comparison in COCOMO II effort model the average time provided for these phases is 58% [21]. Concentrating on the design enables early defect detection and at the same time prevents defect propagation to the later stages of the development.

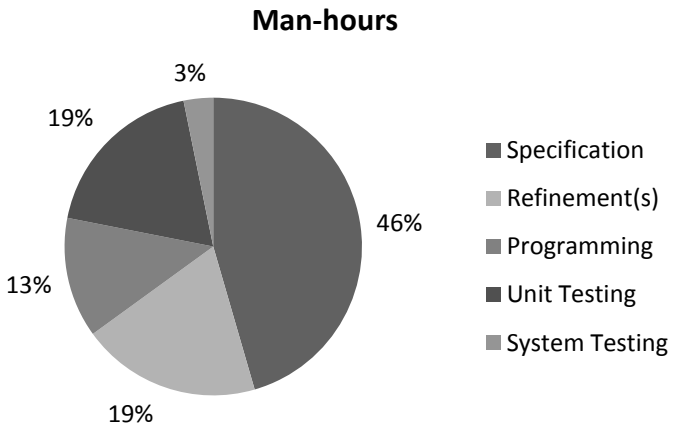


Figure 4. Distribution of time over the development cycle

Design reviews performed by development team were a part of the system design phase (about 10% of the time) and contributed to the high quality of the product. Concentrating on system design and thorough specification of the system benefited in better comprehension of the constructed system. Furthermore, system was well documented, thus facilitating later maintenance.

6. Conclusions and future work directions

Contracts support the decomposition of functionality into subsystems and allow modular analysis of the Simulink models. The analysis can be carried out with different levels of formality. In the case study in this paper the analysis was performed using design reviews and testing. Our experience with using contracts has been encouraging. This positive experience was supported by the evaluation of the impact of used methodology on development process, as well as quality of final product. The contracts facilitate the design process of the system. By documenting the decomposition of the system into subsystems in a legible way, they give explanation for the design decisions. Comprehensible and carefully prepared specification also benefits unit testing and shortens its time. The structure of the system improved when using contracts, as the functionality of the system is well decomposed. Moreover, understandability of the system increased, since the developers focused on the objectives of the created software. This in turn improved its reliability. Given that the subsystems have clear specifications of their functionality, as well as the assumptions they make about the environment where they are used, the system is also more maintainable and reusable. This will benefit in case of the future system adjustments or developments of similar type.

Further research on influence of lightweight formal methods on quality of the created system should be conducted. Measurements can be used for the evaluation and improvement of dependability attributes, as well as give feedback on methods applied in the development process. The complexity of the system is one of the characteristics that have a major impact on its development, usability, maintainability, risk analysis, as well as cost. We have worked on establishing complexity model, which would suit the specific MATLAB/Simulink environment. Our model enables the analysis of the structure and the data flow in the system. We have examined the complexity model for the higher level subsystems. The complexity was computed for each level of subsystems separately. We need to extend this model to cover lower level subsystems. This can be achieved by considering computational layers and ready made Simulink specific subsystems. Afterwards it would be possible

to create the complete complexity metric for the whole system. We would also like to compare our model with other complexity metrics. The aimed relative metric is McCabe complexity [22], adjusted to the Simulink development environment. This would enable the evaluation of the validity of our model.

Correctness of the developed system is important from a dependability point of view. More research on validation of Simulink diagrams with respect to contracts is needed. The contracts can be used as a basis for formal verification of Simulink models. More research is necessary to develop tools and techniques that would enable this to be carried out in an economical way. Validation by testing should also be investigated. When testing, there is a need to determine the appropriateness of test cases. Different types of coverage metrics for both Simulink diagrams and contracts should be investigated.

Bibliography

1. Boström P., Linjama M., Morel L., Siivonen L., Waldén M., *Design and Validation of Digital Controllers for Hydraulics Systems*, The 10th Scandinavian International Conference on Fluid Power. Tampere University of Technology, Tampere, Finland (2007)

2. Boström P., *Formal design and verification of systems using domain-specific languages. Ph D thesis*. Turku Centre for Computer Science, Turku, Finland (2008).

3. Meyer B., *Object-Oriented Software Construction*. 2 ed., Prentice-Hall (1997).

4. Boström Pontus, Waldén Marina, Morel Lionel, *Stepwise development of Simulink models using the refinement calculus framework*, 4th International Colloquium on Theoretical Aspects of Computing (ICTAC2007). Springer, Macao, China (2007)

5. Tripakis S., Sofronis C., Caspi P., Curic A., *Translating discrete-time Simulink to Lustre*, ACM Transactions on Embedded Computing Systems (TECS), 4 (2005), pp.779-818.

6. Cavalcanti A., Clayton P., O'Halloran C., *Control Law Diagrams in Circus*, Proceedings of FM 2005. Springer-Verlag (2005)

7. Pontus Boström, Mikko Huova, Marta Pląska, Matti Linjama, Mikko Heikkilä, Kaisa Sere, Marina Waldén, *development of controllers using Simulink and contract-based design*, International Journal of Critical Computer-Based Systems (IJCCBS), (2009) (Submitted February 2009).

8. Fenton N., Neil M., *Software metrics: roadmap*, Conference on the Future of Software Engineering. , Limerick, Ireland (2000)

9. *Software Technology Roadmap*. Carnegie Mellon Software Engineering Institute, Pittsburgh (2008)

10. Center Software, *Guidelines for Successful Acquisition and Management of Software Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems*. (2003)

11. Linjama M., Koskinen K.T., Vilenius M., *Accurate tracking control of water hydraulic cylinder with non-ideal on/off valves*, International Journal of Fluid Power, 4 (2003), pp.7-16.

12. Linjama M., Huova M., Boström P., Laamanen A., Siivonen L., Morel L., Waldén M., Vilenius M., *Design and Implementation of Energy saving Digital Hydraulic Control System*, The 10th Scandinavian International Conference on Fluid Power. Tampere University of Technology, Tampere, Finland (2007)

13. Siivonen L., Linjama M., Vilenius M., *Analysis of Fault Tolerance of Digital Hydraulic Valve System*, Bath Workshop on Power Transimission and Motion Control (PTMC'05). , Bath, UK (2005)

14. Back R.-J., Kurki-Suonio R., *Decentralization of Process Nets with Centralized Control*, Proceedings of the 2nd ACM SIGACT-SIGOPS Symposium of Principles of Distributed Computing. ACM (1983)

15. Back R.-J., Sere K., *Stepwise Refinement of Action Systems*, Structured Programming, 12 (1991), pp.17-30.

16. Back R.-J., von Wright J., *Trace Refinement of Action Systems*, Proc. of the 5th International Conference on Concurrency Theory, CONCUR'94. Springer-Verlag, Uppsala, Sweden (1994)

17. Mathworks Inc., MATLAB/Simulink, http://www.mathworks.com

18. Hatton L., *What is a formal method, (and what is an informal method)*, COMPASS '97 - Are we making progress towards computer assurance?. IEEE, Gaithersburg, Maryland, USA (1997)

19. Cem K., Bach Kaner,, Pettichord B., *Lessons Learned in Software Testing: A Context-Driven Approach*. Wiley (2001).

20. Pląska M., Waldén M., *Quality Comparison and Evaluation of Digital Hydraulic Control Systems*. Turku Center for Computer Science (TUCS), Turku (2007)

21. Yang Y., He M., Li M., Wang Q., Boehm B., *Phase Distribution of Software Development Effort*, ESEM'08. ACM, Kaiserslautern (2008)

22. McCabe T.J., Butler C.W., *Design Complexity Measurement and Testing*, Communications of the ACM, 32 (1989), pp.1415-1425.

Risk Analysis of Privacy Protection in Social Networking Sites

Heidi E. I. Dahl	Mass Soldal Lund	Ketil Stølen
SINTEF ICT	SINTEF ICT	SINTEF ICT
Heidi.Dahl@sintef.no	Mass.S.Lund@sintef.no	Ketil.Stolen@sintef.no
Norway	Norway	Norway

Extended Abstract

The interest in social networking sites such as Facebook and MySpace have exploded in recent years, and it is a common conception that such networking sites will be central to public participation in the future. Though the main use at the moment is social interaction between individuals, it is clear that communication through social networking could be beneficial in other context. The presence of government agencies and politicians on Facebook is an example of this.

Another emerging use of elements from online social networks is collecting information for research. When doing large scale surveys, enabling social networking features such as user generated content and discussion among the participants allows researchers to collect other kinds of inputs than those accessible through more traditional information collection techniques. These kinds of interaction data are traditionally only available through face to face interaction between researchers and groups of participants.

However, allowing for interaction between participants in a survey entails new challenges in terms of how researchers handle sensitive information, and how the participants are asked to provide details. How to obtain privacy protection in social networking sites is still an open question, but it is evident that security risk analysis must be a key component when the privacy of participants is considered [1][2][3].

We present risks in relation to privacy issues, based on an analysis of the Design Feedback Tool (DFT), an application (in development) for conducting large scale surveys. The DFT combines features from traditional questionnaires with elements from social networking sites. The analysis was performed according to the CORAS method for security risk analysis [4][5][6]. We show how the CORAS method was applied for analysing privacy in the DFT; how this analysis influenced the solution, and how privacy issues of the system are addressed.

Security Risk Analysis of the Design Feedback Tool

The scope of the analysis was the handling of privacy issues in the DFT focusing on privacy and data protection issues connected to the use of DFT, both by researchers and survey participants.

The direct assets focusing the analysis were therefore sensitive personal information and identifying information [7], as two types if information regulated by Norwegian law in terms of privacy. In the following we will use the term sensitive information to mean information of either type. We chose to distinguish between two sources of information, the researcher's data set and user created content.

The risks associated with the researcher's data set are essentially the same as for any sensitive data stored in electronic form, no matter how it was collected. Improper storage and handling of the data before it has been anonymized may lead to sensitive information going astray. Examples of this kind of risk are

- Unencrypted memory stick with data is forgotten in taxi, and accessed by a later passenger.
- Researcher accidentally emails data set to wrong person.
- Researcher not involved in the project associated with the data set gains access because the data set is stored on a common server.

When the survey is conducted online, the researcher needs to keep track of where data is stored and make sure it is secure and not kept when it should be deleted. The DFT is run by the researchers on an external server, so the contract regulating maintenance and backup of the server should take into account possible privacy issues. An example risk related to this is

- Sensitive information from the survey is backed up on the server, and not deleted when the survey has ended.

Using an online community tool such as the DFT to conduct surveys introduces new risk elements compared to more traditional methods where only the researcher sees each participant's responses. There are of course parts of DFT surveys that will remain private (e.g. demographic questions such as gender and age), but even when the information asked for is not necessarily sensitive, participants may include sensitive information by accident. Unless each contribution is moderated, with the time lag this involves, the interaction between the participants may involve disclosure of sensitive data. An example of this kind of risk is

- A woman mentions how she uses an application in relation with her daughter's illness.

Another factor is the use of user generated rich media such as pictures and movies. A rich media file may by itself identify the participant and the surroundings and things that happen in the background may disclose either type of sensitive information about the participant and people close by. An example risk related to this is

- A participant contributes a movie taken at what is clearly a union meeting, showing the people participating in the background.

The above examples were the main risks uncovered in the security risk analysis.

Acknowledgements

The security risk analysis and this extended abstract were completed with funding from the SINTEF-internal project *Rik og Sikker*.

References

[1] Dwyer, C., Hiltz, S. R., & Passerini, K. (2007). Trust and privacy concern within social networking sites: A comparison of Facebook and MySpace. Proceedings of AMCIS 2007. Retrieved 12 March, 2009, from http://csis.pace.edu/~dwyer/research/DwyerAMCIS2007.pdf

[2] Olsen, T., Mahler, T., Seddon, C., Cooper, V., Williams, S., Valdes, M., et al. (2005). Privacy in Relation to Networked Organisations and Identity Management: Legal-IST

[3] Woo, J. (2006). The right not to be identified: privacy and anonymity in the interactive media environment. New Media and Society, 8(6), 649-967.

[4] Folker den Braber, Ida Hogganvik, Mass Soldal Lund, Ketil Stølen, and Fredrik Vraalsen. Model-based security analysis in seven steps – a guided tour to the CORAS method. BT Technology Journal, 25(1):101-117, 2007.

[5] Heidi E. I. Dahl, Ida Hogganvik, and Ketil Stølen. Structured semantics for the CORAS security risk modelling language. Technical Report A970, SINTEF ICT, 2007.

[6] The CORAS tool. Retrieved 12 March, 2009, from http://coras.sourceforge.net/

[7] Personvernombudet for forskning, Ord og Begreper. Retrieved 12 March, 2009, from http://www.nsd.uib.no/personvern/forsk_stud/begreper.html

An Authentication Framework for Nomadic Users

Naveed Ahmed and Christian Damsgaard Jensen
Department of Informatics and Mathematical Modeling (IMM)
Technical University of Denmark, Denmark
nahm@kth.se, christian.jensen@imm.dtu.dk

Abstract

Security and usability are often horn locked and system administrators tend to configure systems so that they favor security over usability. In many cases, however, the increased security results in usability that is so poor that users feel the need to circumvent the security mechanisms. This is probably best explained by considering password based authentication, where a user is actively involved in the process. If the time required to log in to an account is considered too high, users tend to leave their terminals logged in throughout the day and share their account with other users. This is particularly true for nomadic users who move around in ubiquitous computing environments and avail from different IT services from many different locations. In many ubiquitous computing environments, where information processing is not considered the main priority, management often accepts this practise in order to increase productivity, e.g., in a hectic hospital environment, medical staff has to login and logout of various machines several times in an hour, but the repeated interactions consume a considerable amount of time, causing organizational inefficiency, job frustration and a tendency towards defeating the obstacle by leaving terminals logged in or choosing short and easy to type passwords. Therefore, a password based authentication mechanism, which is quite simple and secure in personal computing, has become too cumbersome for nomadic users, which means that other means of authentication must be developed for nomadic users.

In this paper, we focus on usability of authentication for nomadic users in a ubiquitous computing environment. We identify requirements for authentication of nomadic users and propose an authentication framework for this class of users. A prototype of the proposed authentication framework has been developed, which supports persistent and multi-factor authentication without the active intervention of a user.

We evaluate the usability of the developed mechanism by considering the time required to authenticate when logging in to a workstation and compare this to classic password based authentication. The evaluation shows that the proposed mechanism saves a significant amount of time for the nomadic users, which reduces the incentive to circumvent the authentication mechanism. Thus, the mechanism will both provide users with better job satisfaction and increased organizational efficiency, while at the same time increase the effective level of security of the system.

Keywords: *Security, Usability, Ubiquitous Computing, Nomadic Users, Authentication.*

1. Introduction

In the last four decades, human-computer interaction has gone through an evolution. This evolution reflects three different paradigms of computing, which are identified by Allan Kay, Tomei[24], and Weiser[28]. We refer to these paradigms as centralized or mainframe computing, decentralized or personal computing and ubiquitous computing. In the early days of computing, centralized computing was the predominant paradigm, where a single mainframe computer was shared by multiple users on a time or resource basis. Even today, centralized computing plays an important role in the world's largest corporations, including many Fortune-1000 companies[25][27]. From the late 80s, personal computing started to dominate, marked by the number of PC users crossing fifty million[40]. The personal computing paradigm requires a computer to each user and it includes both fixed and mobile computing. More recently, ubiquitous computing paradigm has become more pronounced. In ubiquitous computing, users avails many different machines that are embedded into the environment. Computers used in ubiquitous computing environment are often characterized by being small, inexpensive, robust, shared, networked, and distributed all over the places where they might be required[29].

This evolution of computing paradigms is also reflected in the way computers are used. Depending on how

a person fulfills his computing needs, we identify three phases of user evolution. In the first phase, a user has to access computers that are placed in a single physical location whenever he has a problem that requires computational resources. Thus the freedom of mobility for a user is severely constrained by physical or virtual access to the computing resource. In the second evolution phase, which is mobile computing, users are enabled to move freely, because they can carry their computation resources with them. More recently, a third phase of evolution has emerged, as we start embedding computing devices into artifacts of the surrounding environment. We have termed this as nomadic use of computing, because it is characterized by the fact that there is no inherent need to carry a computing device nor is there any requirement to access a single computer in a central location, because computers have become part of the environment, thus a nomadic user can freely move and compute where ever and when ever required.

Unfortunately, development of suitable security mechanisms lag behind the user's evolution towards nomadic use of computing. The most obvious example of this, is password based authentication, which is used in most operating systems like UNIX, Linux and Windows etc. This mechanism does not impose any serious usability limitation for stationary or mobile users, but for nomadic users, who have to frequently login, logout and share terminals, usability factor plays an important role to determine its effective level of security. Most common security observations are use of small and easy to remember password, sharing password with colleagues and with in a group, omitting to logout etc. From a pure usability point of view, an ample amount of time is being consumed in authentication process, so it is often considered an obstacle to “real work” by users who do not appreciate the underlying need for security.

Common usability problems associated with password based authentication are already highlighted in the literature. A pilot study in health care[34] highlights typical nomadic use cases which cause security vulnerabilities. Jeyaraman and Topkara[3] have recognized usability problems of passwords very well and have proposed some complementary enhancements. A survey for phrase-based passwords shows many vulnerabilities in their practical use[6]. On the other hand, to achieve usability, some graphical password schemes have been proposed[38][4]. Hopper and Blum[5] proposes an alternative to passwords authentication but the result still imposes usability constraints for nomadic use. A security analysis of passwords based authentication by Gorman[19] overlooks this important factor of usability which is significant for nomadic users. The majority of proposed solutions focused on stationary and mobile users and thus lacks in addressing nomadic use of computing where usability requirements are more stringent.

Corner and Noble[7][12] have put forward an authentication mechanism which does not require user interaction at all. Bardram et al.[11] have proposed a secure user authentication mechanism which is proximity based. But their main focus is to avoid vulnerabilities that are part of personal computing paradigm such as theft. The idea of proximity based login has also been used in many other products in ubiquitous computing research like Active Badges[13][21], AT&T's ActiveBat[15] which also uses session migration, Microsoft EasyLiving[14] , IBM's BlueBoard[16], AwareHome[20] and Personal Interaction Points[12] which uses RFID tokens for authentication and XyLoc System[17] etc. However these mechanisms are designed for particular work environments which might be consider as subset of a general nomadic environment. Nevertheless their designer has not analyzed nomadic use cases extensively, to come up with generic requirements which might be applicable to any authentication mechanism used in a nomadic environment.

Similarly the use of multi-factor authentication has extensively been investigated in literature for biometrics[1][2]. Jonnson's 'Jury' framework[8] appears very useful to merge any number of authentication mechanisms together. However, primary focus of research in the area of multi-factor authentication is exploration towards a higher level of confidence in authentication mechanism and thus mostly deficit from usability aspects.

In this paper we have investigated usability aspects of classic password based authentication mechanisms. As a result we have came across some intriguing facts, which must be considered in order to make authentication mechanisms more usable and to raise effective level of security. In fact, our findings points to a simple criteria that any usability feature, which is linked to system security, must be conceded as an integral part of security architecture. Thus, we have designed a authentication framework and have implemented a scaled down version of it in form of a prototype, which consists of a Debian Linux system and RFID reader. The evaluation of system shows a considerable improvement in terms of usability and thereby security.

The rest of the paper is organized as follows; In next section we have presented a brief about password

based authentication in nomadic context to highlight several usability constraints and security vulnerabilities. We have also recounted some relevant security notions. The third section contains details of our designed authentication framework and its prototype implementation. In the fourth section we have presented results of our usability and security analysis, and in last section we have concluded our discussion.

2. Authentication and Nomadic users

To envisage the role of authentication in system security, let's recall some security notions. A security mechanism enforces a security policy. This security policy defines the intended level of security for a security mechanism[39] and we refer to this as 'Designed Security'. However, the actual security achieved in practice is often less than or at most equal to this intended level, we call this the 'Effective Security' level. Moreover, effective security tends to decrease with time, mainly due to the discovery of more vulnerabilities in security mechanisms, advances in technology, improved techniques of cryptanalysis etc. In an authentication mechanism for nomadic users, there is another aspect of this difference which is due to usability caused by repeated and lengthy interactions. Generally, human tends to become more casual with the passage of time. This is illustrated in the Figure 1, which depicts a gradual decline in the effective security over time. It is important to note, however, that the curve shown in Figure 1 is for illustrative purposes only; the actual shape of the curve depends on the type of the system, the configuration and operation of the system and parameters of the computational environment in which the system is deployed.

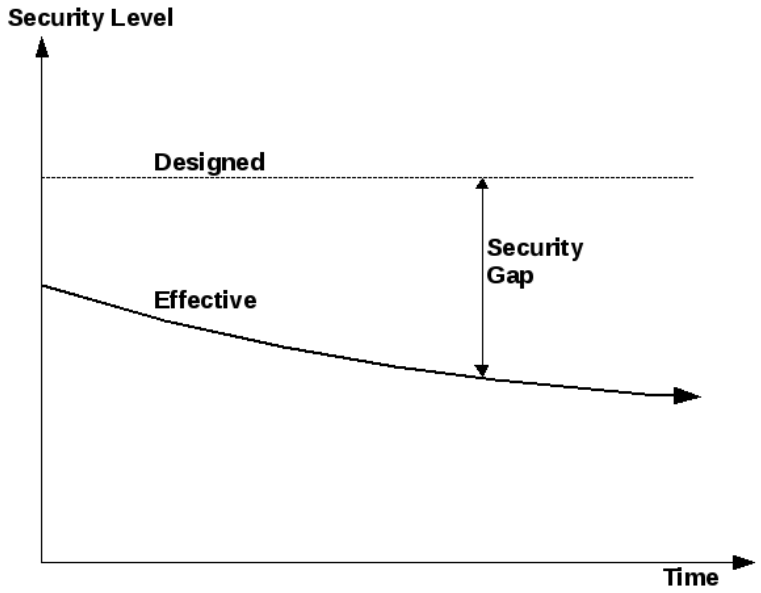


Figure 1: Difference of Effective and Designed Security levels

Password based authentication is a prerequisite for most access control mechanisms implemented in modern operating system like Linux, Unix and Windows etc. Thus passwords are our first and most fundamental security check to ensure computer security. An authentication mechanism is primarily designed to verify the claim made by a subject about their identity[19]. All mechanisms, which authenticate a human, can be divided in two classes. First class is called human-centric authentication and involves recognizing intrinsic or pseudo-intrinsic property of a human. Although Dhamija and Perrig[37] split this class in knowledge based and biometric authentication but we consider knowledge as human pseudo-intrinsic property. Second class is called device-centric authentication, where user is not directly authenticated. In fact, authentication mechanism authenticate a device on behalf of user.

A password based authentication is essentially a human-centric authentication. Generally a human-centric authentication needs active involvement of user and thus can't be done frequently due to usability constraints. For example it is difficult to do persistent authentication after each 500ms for pass phrase or voice recognition. However in device-centric approach, for instance repeated scan of RFID tag after each 500ms is quite common.

Security vulnerabilities using password based authentication have already been identified, especially in the context of health care[35][36]. For instance, modern hospitals have quite ubiquitous infrastructure in form of well connected network with centralized access to electronic patient's record (EPR). Doctor's work place have a couple of terminals to facilitate the check-up of patient or to carry out some research work. Conference halls and meeting rooms have their own terminals. Nurses use terminals at their work place to keep track of patients. Drug stores have a few terminals of their own which are connected to central data base to get information about prescriptions. Emergency rooms and operation theaters also have associated terminals. In a ward, there could be a terminal behind each patient's bed displaying important information regarding the care of the patient to authorized staff. It might be desirable that when a doctor visits a patient in a ward, all relevant data is available in a particular context. Some hospitals also provide terminals in ambulances which are then connected to the hospital information infrastructure. For example this may be used for triage in large emergencies or in-advance preparation in emergency treatment for heart patients.

It's interesting to look at how nomadic users work in the hectic environment of hospitals. For example a doctor has to see dozens of patients, he has to respond to emergency calls and he has to carry out surgery. After a few days, each doctor will have a complete different set of patient and possibly work at many different locations in the ward. As hospitals have to operate 24 hour a day, so typically they work in three or more shifts. One can imagine how computing devices are being shared with in single shift and across shifts. Moreover most of these nomadic users are handling sensitive health data of patients.

The typical behavior of such nomadic users is summarized in following points[36];They tends to use short and easy passwords[41][34], in their desire to login quickly. If the system enforces mechanisms to ensure the quality of passwords, people tend to write their password down, in order to avoid forgetting it. Sometimes they do not even logout. This might be on purpose to save time on their return or due to age related forgetfulness[22]. Passwords may be sniffed due to continual need to be typed in during nomadic use. Users tend to give away their password to their colleague for delegation purpose and sometime share it in a group.

Beside the wastefulness of ample amount of time, there are also very obvious security vulnerabilities. Short and easy passwords are always subject to numerous attacks as indicated in some security analysis[19][9][10], and this is more certain for nomadic use. The common problem with classic knowledge based authentication is that people tend to forget it[37][41], causing denial of service. Writing a complex password on a piece of paper is violation of very basis of human centric authentication and vulnerable to stealing. We agree with Bardram[11] and consider logout as an integral part of the authentication process. Failure to do so, is in fact a failure of the authentication mechanism. Password sniffing is very difficult to avoid in nomadic use. Giving away one's password is like making a copy of an authentication token and giving it to someone-else, and moreover allowing to make further copies of that. This is the worst form of delegation. In group based password scheme, no one is accountable although every one is authorized. Revocation is difficult to achieve in password based authentication. Accenture Terminated Employee Survey[41]shows that about 10% of ex-employee access the databases of their former employer. This is because employer were not able to properly revoke their permissions. Moreover revocation takes effect when user tries to re-authenticate and thus in some cases attacker get a chance of stealing a session. Active user involvement in authentication process consumes lot of time when we add small chunks of time of all nomadic users. Apparently this is a matter of efficiency but may cause partial denial of service attack. An attack on system security is always expected from weakest point[31] and thus these vulnerabilities are points of attack and must be addressed.

After analyzing password based authentication mechanisms in the context of nomadic users, we have reached on some conclusions. First of all, there should be no conscious interaction between a user and the authentication mechanism. As we have seen previously, that a small amount of active interaction ends wasting a substantial amount of time when we add authentication events over a period of time. Moreover, frequent interruptions and delays provide strong motivation for a busy nomadic user to circumvent the authentication mechanism.

Further we believe that the underlying architecture of an authentication mechanism should support persistence and preferably context based authentication. This avoids a major usability constraint by shifting responsibility of logout from users to the authentication mechanism in a way that preserves accountability. We also conclude that the mechanism should support a user-friendly delegation preferably at authentication level. This reflects the fact that in practice, delegation is normally accomplished by giving away one's password or using a group password. And at last, authentication mechanism should be multi-factor. This is

because human centric authentication is formidable to perform periodically for sake of persistence and in most of the cases a conscious user engagement is required. On other hand device centric authentication has problems of it's own in form of stolen tokens and lack of trust on single authentication device etc.

Some of these conclusions, although in a different context, are already in the state of art in form of Corner and Nobel's work on persistent authentication[12], Bardram's proximity based login[11] and Jonsson's co-authentication framework[8]. Importantly, our conclusions are implementation independent and also not aimed to increase the level of designed security. In fact, their expressed purpose is to avoid usability constraints but indirectly they avoid the decrement in effective level of security in nomadic environments.

3. Authentication Framework for Nomadic Users

We have designed an authentication framework based on our analysis presented in the previous section. Our framework is intended for a networked nomadic environment and uses persistent and multi-factor authentication techniques without conscious engagement of users. We have termed it as 'Authentication Framework for Nomadics', and abbreviated as AFN. Following is the design detail and rationale for AFN.

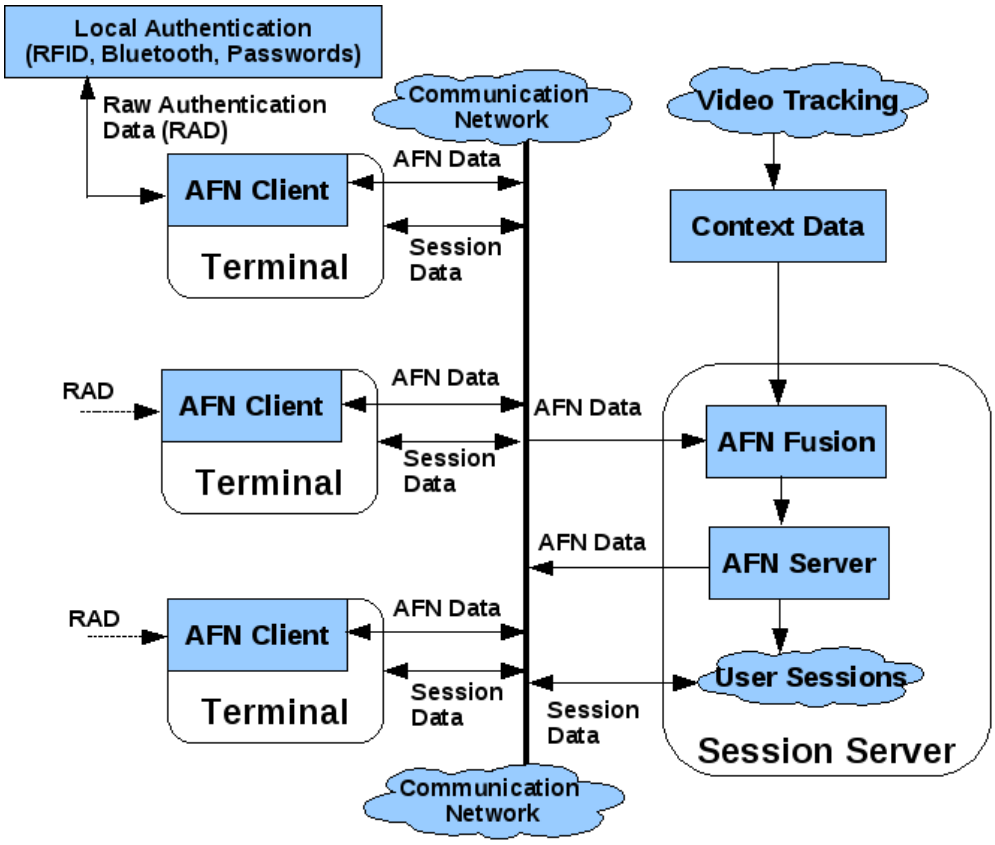


Figure 2: Top-level Architecture of AFN Authentication Framework

The basic architecture of AFN is shown in the Figure 2. It is a distributed network framework and it consists of three main parts. On each client terminal, we have a 'AFN-Client' application who interact with local authentication sources (e.g. RFID reader, Bluetooth etc). This application sends authentication data to 'AFN-Fusion', which is part of the main session server. 'AFN-Fusion' is a fusion engine for all type of authentication data in the system, including context data. It output authentication decision to 'AFN-Server' application which is responsible for activating, suspending, locking and unlocking of a user session. All user session reside on session server and can be remotely accessed from any terminal. We can configure it to implement a specific authentication policy. For example, an authentication policy can specify that authentication should be granted only if user's RFID badge and his bluetooth mobile are both present and validated. Underlying detail on multi-factor technology can be found in Jonnson's co-authentication framework[8]. Similarly the detail about context data (i.e. user's location) and corresponding authentication

techniques can be found in 'PAISE' scheme and related experiment by Kirschmeyer, Hansens and Jensen[33].

'AFN-Server' is responsible for starting, stopping, locking, unlocking of a user session which can be remotely accessed from a terminal. The decision of AFN-Server depends on authentication decisions being received from AFN-Fusion module. For example, if a user is sufficiently authenticated on a terminal, his relevant session is automatically invoked at that terminal. When user departs, the terminal is locked and session is suspended back to session server.

On client side, the most natural and efficient way of authentication for nomadic user is proximity-based login technique, which allows users to be authenticated on a device simply by approaching it physically[11]. However specific techniques for proximity based login are not in the scope of this paper, but it appears obvious that the choice of proximity based login mechanism will have a strong impact on the overall security of the system. However, in this paper we are more interested in usability and security improvements that can be obtained through the usability of the security mechanisms. It was indicated earlier that most human-centric authentication techniques are not suitable for persistence and seamless active authentication. Although, there are plenty of device centric approaches, these again come with a risk of device being stolen or tampered with. The best solution could be to combine the benefit of each technique, which leads to a hybrid solution where multiple authentication techniques are merged to form a unified authentication mechanism, which is also known as multi-factor authentication.

For nomadic users we may use the Jury framework[8] for achieving multi-factor authentication in our proposed framework. It enables an easy integration of arbitrary many authentication techniques, including biometrics, knowledge based and device centric techniques. Moreover it's easy to integrate with wide range access control systems which work on either probabilistic or binary authentication result. For instance, we can combine RFID, Bluetooth and password based authentication along with machine learning algorithms with help of this framework. A typical authentication policy for this framework could be as follows:

1. When user enter in the nomadic environment first time, he should enter his password.
2. System automatically associate RFID chips present in his clothes, watch, shoes and badge, with user identity. Similarly an active Bluetooth from user mobile is an additional binding parameter. We can specify that whenever RFID badge along with one of additional binding parameter is present, system should consider it as an authenticated user.
3. Machine learning may be used to not automatically login on a system, when user just pass nearby. Also a mechanism should not automatically try to authenticate a user on a system, which a user normally does not use.

In our framework, authentication data used in fusion process and authentication data used by 'AFN-Server' to invoke corresponding sessions is independent. The mapping between these two types depends on security certificate of users. A user's security certificate binds a user to multiple authentication devices. This architecture also enables us for a user level delegation.

To demonstrate our authentication framework, we have implemented a scaled down version. This prototype consists of a Intel based desktop computer, running UBUNTU Linux(which is most popular distribution of Debian Linux[18]) with GNOME desktop. Both server and client part of authentication are present on single machine and communicate with each other using network sockets. For authentication purpose we have augmented classic login based mechanism with RFID based authentication mechanism, which represents a branch in multi-factor authentication.

Multiple user sessions are simultaneously present on computer. When a person wearing a RFID tag approaches the machine, his relevant session becomes active and displayed. Also this mechanism provides persistent authentication which means that user is authenticated as long as he is present on machine. If he depart from the scene authentication is automatically revoked in the system and relevant session is suspended which might be reactivated in future.

4. Analysis and Result

First of all, let's consider security vulnerabilities caused by usability and were described in previous

section. In our mechanism, user has to enter password once so there is no motivation for using short and easy passwords. Moreover, disclosure of password is not a complete system failure but in fact, it can be detected by our multi-factor authentication. We have used persistence authentication and thus user logout as soon as he departs from scene. As user does not have to enter password in normal flow of work so password sniffing by observation is also not possible. We support a nice user level delegation which is secure way of giving one's password to other person and also there is no need to have a group password. Our mechanism is persistent with very short term authentication memory which makes attack due to stealing of pre-authenticated session not possible. As authentication process is automated, it saves a user from wasting valuable time in authentication process. These facts suggest that our mechanism is able to address a large set of security vulnerabilities and thus contributes to increase the effective level of system security.

We have also analyzed implementation of our security mechanism by dividing it in three steps and then performing detailed security analysis at each step. Some underlying assumptions for our prototype, which are part of access control mechanism provided by Debian Linux, are as follows; Firstly, there is only one super user in server system. Secondly, 'AFN Fusion' and 'AFN Server' runs with privileges of this super user. Thirdly, their is a certain trust level on physical integrity of sensors, for instance RFID reader is not tampered. Fourthly, each terminal is running a trusted copy client application 'AFN Client' and Fifthly, interprocess communication using network socket is secure. These assumptions are quite reasonable for a Linux based network.

In the first step 'AFN fusion' receives all authentication information from individual sensors. For our prototype this means a list of identifications for all tag present in the field of RFID reader. Since we have assumed that hardware is not tampered with, thus this data present a valid input. One may argue that a tag might be cloned, but this can be avoided by using tamper-resistant tags e.g. Zero knowledge RFID as described by Engberg, Harning and Jensen[32]. Implicitly we are also assuming some basic conditions which are part of every device centric authentication, like RFID. This implies that tag presented for authentication is currently possessed by owner. Moreover a inherent deficit of confidence on any device centric mechanism is well represented during fusion where each authentication sensor has certain level of trust.

In the second step 'AFN fusion' is able to combine all authentication data it received. The only security vulnerability which is possible in this case is denial of service. For example tag is not scanned in previous interrogation. It is also possible, for example,if their are large number of tags present in field. To avoid this, 'AFN Server' samples authentication data at relatively high rate and operate on principle of moving average. Third step is correctly mapping authentication data to user session. This involves scanning the user's security certificates to find out the user for which output of 'AFN fusion' is valid. All these certificates are signed so they can't be tampered with and also user's don't have write access to them. 'AFN Server' will not activate a session on a particular terminal, if a valid 'AFN client' is not running their.

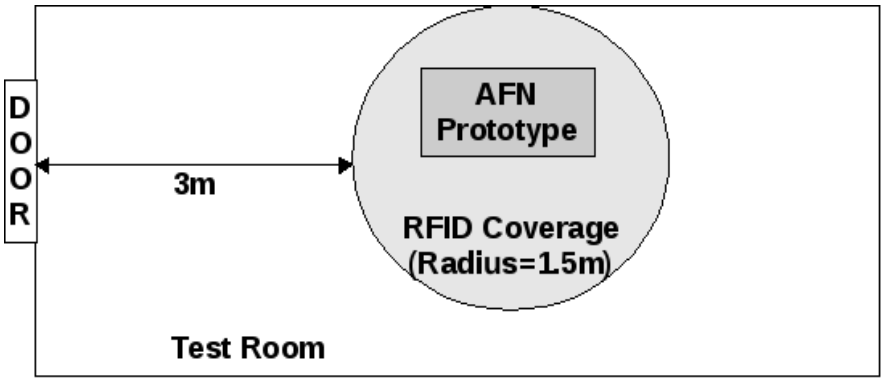


Figure 3: Experimental setup

To estimate usability performance let's define a benchmark to represents a typical nomadic environment. We assume a user login 12 time in an hour and their are 8 working hours per day. The physical infrastructure for our experiment is shown in the Figure 3.

Test results are shown in the Table-1, by averaging one hundred individual authentications for each case. Most notable is total authentication time in a whole day, which turns out to be just 1.8 minutes for our prototype. This is quite significant as compared to almost 20 minute time consumed by password based authentication. This is because an average user takes around 8 second to type a good complex password. We have experimented with passwords of length ten, and they all include alphabets, numeric and special character. This quality of password is roughly equivalent to 64-bit random identification stored in RFID tags we have used in the experiment. In other words, by using the prototype, one can have an extra person for every 24 persons which might be very attractive especially for those environments where price for a working hour is high, for instance in hospitals.

S/N	**Feature**	**AFN prototype**	**Password based system**
1	Session Lock	Automatic	Manual
2	Session Unlock	Automatic and takes less than 1 second	Manual, On average take about 8 seconds for good passwords.
3	Switching User	Automatic and takes less than 1 second	Manual, On average take about 25 seconds, if system does not support simultaneous multiple sessions. In other case it takes 10 seconds.
4	Run-time System Resources	~500MHz of Celeron D processor	Zero
5	New Session Creation Time	10 seconds	20 seconds
6	Daily time consumed in Authentication	1.8 minutes	20 minutes

Table 1: Performance Comparision

Conclusion

Unfortunately, the evolution towards nomadic use of computing is not accompanied by use of appropriate security mechanism. A password based authentication mechanism, when used by nomadic users, leads to many security vulnerabilities, causing a considerable drop in level of effective security. It turn out that most of these security vulnerabilities are due to usability constraints present in classic authentication mechanisms As nomadic users are very conscious about usability issues, so they try to circumvent security checks. Thus usability should be considered as an integral part of overall design of an authentication mechanism.

We have developed an authentication framework and have evaluated its prototype, from both usability and security perspectives. We have improved the quality of interaction between human and computer by minimizing usability constraints. This contributes to saving time which may result in more job satisfaction along with economic advantages for an organization. We have increased the level of effective security by addressing usability constraints related to authentication mechanisms. As we have shown in our security analysis we have removed most of those security vulnerabilities which are normally present in a system where user is actively involved in authentication process.

At last, we also confess that a true analysis of nomadic users require cross-disciplinary approach including anthropology and psychology along with a couple of clinical and field trials, in order to truly discover the best usability options that can be incorporated in any security mechanism.

References

[1] Lin Hong, Anil K. Jain, and Sharath Pankanti, " Can multibiometrics improve performance", Technical Report MSU-CSE-99-39, Department of Computer Science, Michigan State University, 1999.

[2] Imran Naseem and Ajmal Mian, "User Verification by Combining Speech and Face Biometrics in Video", Advances in Visual Computing, ISBN 978-3-540-89645-6, Pg. 482-492, 2008.

[3] Sundararaman Jeyaraman and Umut Topkara , "Have the cake and eat it too – Infusing usability into text-password based authentication systems", Proceedings of the 21st ACSAC, Pg. 473 –482, 2005.

[4] D. Davis, F. Monrose and M. K. Reiter, "On User Choice in Graphical Password Schemes," In Proceedings of the 13th UNIX Security Symposium, August 2004.

[5] Nicholas J. Hopper and Manuel Blum, "A secure human computer authentication schemes", CMU-CS-00-139, School of Computer Science, Carneige Mellon University, May 2000.

[6] Cynthia Kuo, Sasha Romanosky and Lorrie Faith Cranor, "Human Selection of Mnemonic Phrase-based Passwords", ACM International Conference Proceeding Series Vol. 149, Pg. 67–78, 2006.

[7] Mark D. Corner and Brian D. Noble, "Zero-interaction authentication", Proceedings of the 8th annual international conference on Mobile computing and networking Atlanta, Georgia, Pg. 1–11, 2002.

[8] Einar Jonsson, "Co-Authentication - A Probabilistic Approach to Authentication", Master's thesis, IMM-Thesis-2007-83, Informatics and Mathematical Modeling, Technical University of Denmark, DTU, 2007.

[9] Bruce L. Riddle, Murray S. Miron, and Judith A. Semo, "Passwords in use in a university timesharing environment", Computers and Security Vol 8 (7), Pg. 569 – 578, November 1989.

[10] Daniel V. Klein, "Foiling the cracker: A survey of, and improvements to, password security", Proceedings of the second USENIX Workshop on Security, Pg. 5-14, July 1990.

[11] Jakob E. Bardram, Rasmus E. Kjær, and Michael Ø. Pedersen, "Context-Aware User Authentication: Supporting Proximity-Based Login in Pervasive", UbiComp 2003: Ubiquitous Computing, Pg. 107-123, 2003.

[12] Mark D. Corner, Brian D. Noble, "Protecting applications with transient authentication", Proceedings of the 1st international conference on Mobile systems, San Francisco, California, Pg. 57 – 70, 2003.

[13] F. Bennett, T. Richardson, and A. Harter, "Teleporting- Making Applications Mobile", Proceedings of the IEEE Workshop on Mobile Computer Systems and Applications, Pg. 82–84, 1994.

[14] B. Brumitt, B. Meyers, J. Krumm, A. Kern and S. Shafer, "EasyLiving: Technologies for Intelligent Environments", Handheld and Ubiquitous Computing, Pg. 97-119, 2000.

[15] A. Ward, A. Jones, and A. Hopper, "A new location technique for the active office", IEEE Personal Communications, Vol. 4(5), Pg. 42-47, October 1997.

[16] Daniel M. Russell and Rich Gossweiler, "On the Design of Personal & Communal Large Information Scale Appliances", Ubicomp 2001: Ubiquitous Computing, Pg. 354-361, January 01, 2001.

[17] Xyloc family of products, Ensure Technologies (Ypsilanti, Michigan) , <http://www.ensuretech.com>, Last visited March 24th, 2009.

[18] Ladislav Bodnar, "Top Ten Linux Distributions", <http://distrowatch.com/>, Last visited April 1st, 2009.

[19] Lawrence O'Gorman, "Comparing Passwords, Tokens, and Biometrics for User Authentication", Proceedings of the IEEE, Vol 91(12), Pg 2019-2040, 2003.

[20] K. Nagel, C. D. Kidd, O'Connell, T. O'Connell, A. Dey and G. D. Abowd, "The Family Intercom: Developing a Context-Aware Audio Communication System", Proceedings of UBICOMP, Pg. 176-183, 2001.

[21] R. Want, A. Hopper, V. Falco, and J. Gibbons, "The Active Badge Location System," ACM Transaction

on Information Systems, Vol 10(1), Pg. 91-102, January1992.

[22] Science News University of California, San Francisco. "Age-related Memory Loss Tied To Slip In Filtering Information Quickly." ScienceDaily dated 5 September 2008. <http://www.sciencedaily.com/releases/2008/09/080902143234.htm>, Last visited April 1st, 2009.

[23] Department of Defense, Trusted Computer System Evaluation Criteria dated 1985, <http://csrc.nist.gov/publications/history/dod85.pdf>, Last visited March 30th, 2009.

[24] Lawrence A. Tomei , "Encyclopedia of Information Technology Curriculum Integration", Information Science Reference; illustrated edition , ISBN-13: 978-1599048819, February 5, 2008.

[25] Mike Ebbers, Wayne O'Brien and Bill Ogden, "Introduction to the New Mainframe: z/OS Basics" dated July 2006, <http://publibz.boulder.ibm.com/zoslib/pdf/zosbasic.pdf>, last visited March 26th, 2009.

[27] Pam Snaith and Rob Steiskal, "Mainframes are still mainstream", White paper by CA Inc, June 2007. <www.ca.com>, Last visited March 30th, 2009.

[28] Mark Weasor, "Nomadic Issues in Ubiquitous Computing", Xerox PARC (Palo Alto Research Center), <http://www.ubiq.com/hypertext/weiser/NomadicInteractive> , last visited March 26th, 2009.

[29] Marcia Riley, "Ubiquitous Computing: An Interesting New Paradigm", <http://www.cc.gatech.edu/classes/cs6751_97_fall/projects/say-cheese/marcia/mfinal.html>,Last visited March 26th, 2009.

[30] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege and D. Spence, "Network Working Group: RFC 2904", August 2000.

[31] Charles P. Pfleeger and Shari Lawrence Pfleeger, "Security in Computing", Prentice Hall Professional Technical Reference, 2002.

[32] Stephan J. Engberg, Morten B. Harning and Christian Damsgaard Jensen, "Zero-knowledge Device Authentication:Privacy & Security Enhanced RFID preserving Business Value and Consumer Convenience", Proceedings of the 2nd Annual Conference on Privacy, Security and Trust (PST'04), 2004

[33] Martin Kirschmeyer, Mads S. Hansen and Christian D. Jensen, "Persistent Authentication in Smart Environments", 2nd International Workshop on Combining Context with Trust, Security and Privacy. Trondheim, Norway, 2008.

[34] J. Bardram, T. Kjær and C. Nielsen, "Mobility in Healthcare - Reporting on our initial Observations and Pilot Study",Technical report of a clinical study, CfPC 2003-PB-52, Center for Pervasive Computing, 2003.

[35] Jens Bæk Jørgensen and Claus Bossen, "Executable Use Cases for Pervasive Healthcare", IEEE Software Volume 21 , Issue 2, Pg. 34 – 41, ISSN:0740-7459, March 2004.

[36] Jakob Bardram, "The trouble with login: on usability and computer security in ubiquitous computing", Personal and Ubiquitous Computing Vol9(6), Pg. 357–367, ISSN:1617-4909, November 2005

[37] Rachna Dhamija and Adrian Perrig, "Deja Vu: A user study using images for authentication", In the Proceedings of the 9th USENIX Security Symposium, Denver, Colorado, August 2000.

[38] I. Jermyn, A. Mayer, F. Monrose, M. Reiter and A. Rubin. "The Design and Analysis of Graphical Passwords", Proceedings of the 8th UNIX Security Symposium, August 1999.

[39] Matt Bishop, "Computer Security: Art and Science" , book published by Addison-Wesley Professional, ISBN-13: 978-0201440997, 2002.

[40] Computer Industry Almanac, "25-Year PC Anniversary Statistics", Press release August-2006, <http://www.c-i-a.com/pr0806.htm>, Last visited April 1st, 2009.

[41] Password Research, "Authentication Statistic Index" maintained by Bruce K. Marshall, <http://passwordresearch.com/stats/statindex.html>, Last visited April 1st, 2009.

Authenticating Mobile Users Using Untrusted Computers - A Smartphone Approach

Anna Vapen

Department of computer and information science
Linköpings universitet, SE-58183 Linköping, Sweden
e-mail: annva@ida.liu.se

Users of web applications are mobile in the sense that they use different computers, for example at work, at home or at an Internet café. These computers can be considered untrusted since they can contain keyloggers and malware. When a user logs in to use a web application there is a need for an authentication method that is secure even if the computer cannot be trusted, and that can be used on any computer anywhere.

Normally usernames and passwords are used in simple authentication solutions. The problem is that users need to remember a large variety of different usernames and passwords. For a password to be secure it needs to be relatively complex which makes it difficult to remember. To use complex passwords there is a need for secure storage of the passwords, or an alternative method such as using one-time passwords and challenge-response where the user is presented with a challenge that only this user can give the correct response to.

Both for storage of passwords and for running a cryptographic application that calculates a response we need a trusted environment that can be reached by the mobile user. Many security-conscious organizations, specifically in the areas of e-commerce and online banking, use hardware security tokens for authentication. The token can be for example a smartcard, a USB-stick or other hardware specific for the application. In recent years mobile phones have been used in authentication solutions and identity management. Because of their prevalence, mobile phones are an excellent platform for something the user wants to have available at all times.

Using a smartphone, an enhanced mobile phone similar to a PDA, we get access to a rich set of input channels (e.g. camera, voice, keypad, touch screen, accelerometers, GPS etc) and communication channels (long distance as GSM/WCDMA, WiFi etc and short distance as Bluetooth, NFC etc). They also contain trusted hardware in the form of a SIM or USIM card.

An interesting challenge with using a smartphone as a hardware security token is to explore its interactive features to be able to create a highly usable and fast authentication solution that can provide a high level of security in security critical settings as well as for simpler services like social networks, e-mail and blogs. One solution that we are investigating is the possibility of using optical input as part of an authentication process where the user has a smartphone as a hardware token.

Efficiency Issues in a Switched LAN

Luigia Petre lpetre@abo.fi
Muhammad Mustafa Hassan mhassan@abo.fi
Department of Information Technologies, Åbo Akademi University

***Abstract*—Ethernet—a switched local area network—is the market leader for LAN Technologies today. From businesses to government organizations, home to shopping centers, and NGOs to universities, Ethernet is the sole shareholder of the local area networks. Due to such a wide-range, Ethernet has become an intensively addressed research area, with concerns ranging from the high level network design problems to the mid-level theoretical protocols architectures to the low level physical hardware details. As a result, new standards, increased speeds and better technology for Ethernet are very often reported.**

Due to the fast and intensive development, the diameter of the LAN deployment has been increased to a bigger scale. Local area networks have grown from a couple of hundreds of clients to several thousands of clients advancing the concept of a Campus Area Network. The huge enhancements in network size—and constant growth—led to various concerns in the switched network design, some very crucial. First, the fault tolerance and the availability of the network should be ensured for virtually 100% of the time. Second, the aggregation of the switched traffic flow to the network core and the Internet is a major aspect, as the proper size of upstream links keeps the network from being congested. Third, the proper design of the switched network has quality, speed and efficiency aspects to consider.

The above mentioned issues do not only arise when a network is being deployed. They can also arise after the establishment of a network, due to several reasons like arbitrary growth, careless planning, design mistakes, replacement of devices/links upon failures or for extending, etc. All these concerns lead to one common theme: that of the efficiency of the LAN.

This thesis provides a case study of a university LAN: we set out to uncover just how efficient the Åbo Akademi University's Ethernet is. The results we present are obviously characteristic to this particular network. We found that the network design of the Åbo Akademi LAN was influenced over the years by the limited budget as well as the lack of stringent security and efficiency needs. While there is no reason to generalize our findings, we can still draw several lessons from our study. We have observed various efficiency problems and proposed solutions to them. By simulation we have also visually illustrated the differences and improvements our solutions would bring.

Index Terms—Ethernet, hierarchal design, LAN, Switched LAN, Switching, ÅA Network

I. Introduction

Computing and communication have had an essential role in people's lives from the early ages. Starting from abacus and tally marks to the room-sized ENIAC, and then to the PC and dust computing devices, a drastic revolution took place in the field of computing. The field of data communication also transformed radically—from lighted beacons, smoke signals and homing pigeons to telegraph and radio, and then to the computer networks and the Internet. During the early stages, the means for storage, processing and communication of data were distant. Humans used to store data in their minds, and later on papers. For computing they needed some abacus type of machines, while for communications they needed drums or smoke signals and later the telegraph. With the advent of the computers and the stored program concept, the means of storage and processing merged. Still, communication was isolated. The forms of data to be stored and to be communicated were different. With the invention of ALOHA and ARPA in early 1970s [HYPERLINK \l "Bre99" 1], these fields started to integrate. The same machines were now able to store, process and send data—with the involvement of some Data Communication Equipment (DCE). Today, we see an almost complete integration. A normal computer can store the data of any type—voice, video, images, programs, games, etc.—, process it, and send it to another computer. Storing, processing and transmitting use data in compatible formats and take place over computers2], [HYPERLINK \l "Tan96" 3].

Because of this integration, during the last two decades, the use of networks has grown drastically both at the individual and organizational level. Today networks have become an integrated part of human life. Their use ranges from military applications to space science and business applications to home and leisure activities. The tremendous increase in the use of networks everywhere has transformed them into a basic necessity and a very crucial theme for researchers. While we increasingly need networks, we also have increasing problems associated with them. So, we need more research into different concerns relating to the networks—like stability, scalability, delays, bandwidth and congestion etc. Different type of networks exhibit different problems, thus having different themes of research. Ethernet, a switched *Local Area Network* (LAN) is one of the most used networks. In the following, we introduce Ethernet and present some problems associated with it in the case of *Åbo Akademi* (ÅA) university Network. We then propose some possible solutions justified via simulations.

II. Switched Ethernet

In the late 1960s, Norman Abramson along with his colleagues at University of Hawaii introduced the concept of *contention based networks* 1], [HYPERLINK \l "Tan96" 3]. Their network ran on a shared wireless medium with a medium access scheme named ALOHA. The resulting network was called ALOHAnet—a broadcasting network because of the shared wireless medium. In 1972, shortly after the introduction of ALOHA, Bob Metcalfe and David Boggs at PARC, XEROX Corporation further came up with enhancing the original ALOHA concept to a 2.94 Mbps data rate network1], [HYPERLINK \l "Int08" 4]. The major enhancements were the use of a cable instead of the wireless medium, the introduction of carrier sensing, and the speed. This newly introduced medium access mechanism was called *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD). The resulting network was named Ethernet1], [HYPERLINK \l "Wil97" 2], 5].

From 1972 to 1990, Ethernet received a number of modifications and enhancement but the core concept of the technology remained the same. With the advent of the Ethernet switch at Kalpana [HYPERLINK \l "Bre99" 1],6] in 1990, the shape of the underlying technology and devices changed. The network started to use Ethernet switches instead of hubs. The basic technique in an Ethernet switch was similar to a telephone switch in which the two ports of the communicating parties were connected for the time of the communication only. This relatively new invention for the Ethernet changed the way the medium was being used. With the use of a switch the network was no longer a broadcasting network. Three years after the introduction of switches, Kalpana delivered another big invention in 1993 namely the *full-duplex transmission* mode. The full-duplex transmission allowed the clients to send/receive simultaneously. Before the invention of Ethernet switches, the full-duplex transmission was not possible because the network ran on a shared medium. Later on, in 1997, IEEE ratified its standard 802.3x for full-duplex/flow control Ethernet [HYPERLINK \l "Bre99" 1], 7].

These inventions unleashed the realms of Ethernet. Soon after, in 1995, the 100 Mbps version arrived as IEEE802.3u [HYPERLINK \l "IEE08" 7], then in 1999, 802.3ab which provides a speed of 1000 Mbps. Then in 2002, IEEE802.3ae was published which further enhanced the Ethernet to 10 Gbps. And now IEEE 802.3ba group is working for the ratification of 40 Gbps and 100 Gbps Ethernet standards 8]. All this rapid development became possible because of the invention of switching technology for Ethernet.

III. Switched Network Design

Several protocols rely on directed broadcasts for their correct functioning. These protocols include, for example, *Address Resolution Protocol* (ARP) and *Routing Information Protocol* (RIP) [HYPERLINK \l "Wil98" 9], 10]. As the networks grew bigger, the problems associated with one large enterprise-wide single broadcast domain became apparent in terms of congestion, latency, waste of bandwidth and the increased load on the processors of connected nodes [HYPERLINK \l "Boy01" 10] 11]. A traditional solution to resolve the problem was to divide the whole network into several small broadcast domains with the use of a *router*. Hence, routers are CPU-based devices which calculate forwarding and routing decision on a per-packet base, the involvement of a processor and memory for the forwarding/routing of each and every packet and its decapsulation/encapsulation introduces a considerable amount of delay that becomes a bottleneck in high-speed switching LANs [HYPERLINK \l "Boy01" 10]. With the advent of *Virtual LANs*, *higher layer switching* and *multi layer switches*, the routers that were being used to separate the broadcast domain are replaced with multi layer switches 12]. But the huge sizes of networks and the large variety of devices and functionalities available has made the network design much more complex [HYPERLINK \l "Int083" 12]. Building flat networks without a proper design strategy is not any more in practice now. Today, designers use what we recognize as the hierarchal network design13], [HYPERLINK \l "Boy01" 10], 14].

A. Hierarchal Network Design

The hierarchal network design is based on the traffic flows in enterprise's *Campus Area Network* (CAN). A CAN is normally a complex network, hence for the simplicity it is decomposed into three layers. First, we have the traffic which may be destined for services which are campus-wide or external in their scope. Second, there is the traffic which is between similar broadcast domains on the same CAN. The third and the basic one is destined for a local broadcast domain [HYPERLINK \l "Boy01" 10].

To clarify this structure we take an example of Åbo Akademi that is composed of several faculties with each faculty having several departments. Figure 3.1 illustrates the concept of every department having its own network segment. The traffic generated from a client on the segment of the Department of Information Technologies and destined for the same segment is local traffic. If the source is at the Department of Information Technologies and the destination is at the Department of Chemical Engineering (same faculty) then the traffic is remote. And finally, if the traffic is generated for a destination which is neither in same segment, nor in the same faculty—may be outside the campus network, or a resource on the campus network—then this traffic is campus-wide or enterprise-wide in its scope.

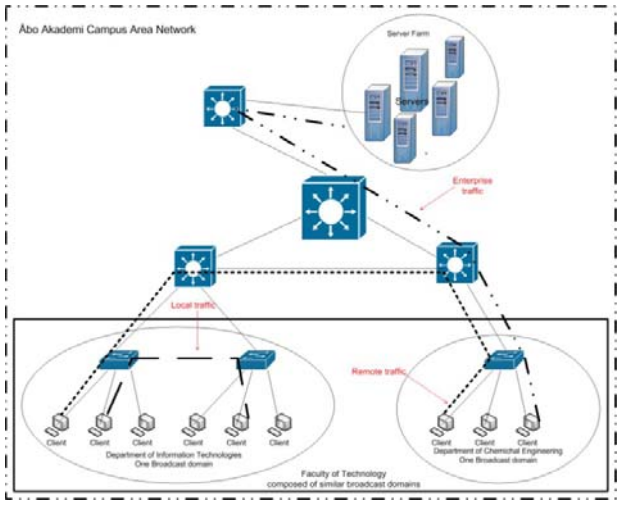


Figure 3.1: An example of different traffic flows

Based on these traffic flows, a hierarchal network design consists of three layers. The access layer 10], [HYPERLINK \l "Int083" 12], 15], [HYPERLINK \l "Way03" 16] provides the connectivity to the end devices. The medium used by these devices can be in any form, ranging from wired rings, shared or switched mediums to the wireless medium. The devices which form this layer are normally functioning at the data link layer. Logically, the access layer devices need not have any routing or similar network layer functionality. This is because their role is to provide medium access to the end user. The only other functionality they need to perform is the separation of broadcast domain into several smaller broadcast domains. This job can be accomplished very well with the definition of VLANs.

The distribution layer 10], [HYPERLINK \l "Int083" 12], 15], [HYPERLINK \l "Way03" 16] aggregates the traffic coming from the access layer devices, performs inter-VLAN routing and enforces enterprise policies. If the traffic generated by the end systems is destined to the same broadcast domain, then it needs not to be sent to the distribution layer. If the traffic flows need to go out of the generating broadcast domain, the access layer device hands over the traffic to the distribution device. Distribution device checks for all policies, access control mechanisms, and routing information and then acts as defined for that particular type of traffic.

The core layer 10], [HYPERLINK \l "Int083" 12], 15], [HYPERLINK \l "Way03" 16] is the heart of a network. It is the place where the aggregation of the entire traffic over a network takes place. Due to the very high volume of traffic present, the devices at the core layer should be extremely efficient. This efficiency requirement is achieved with a combination of two aspects. First, very efficient devices need to be used with the greatest throughput both at network layer and data link layer. Second, this layer is given the responsibility to switch frames on data link layer. Normally, no inter-VLAN routing, packet processing, enforcement of policies or similar functions can be given to this layer—although it can perform all these functions.

IV. Åbo Akademi Switched Ethernet

Åbo Akademi University is a traditional seat of learning situated in Turku, Finland. The university owns a reasonably large network which is spanned over the buildings scattered across the city. The network is built upon Ethernet technology using all the switching devices. Only the wireless LAN devices are on a shared broadcasting medium network due to the natural constraints on wireless medium. The network has a hierarchal structure which is apparently built upon three layers. One layer is providing connectivity to end devices. The middle layer is aggregating the traffic from end switches. And the third layer is working as the core of the network. However, these three layers do not fit in the defined three layer architecture that we discussed in the previous section. This is because the logical functions a distribution layer performs—inter-VLAN routing, policy enforcement etc—are not being performed in the middle layer in Åbo Akademi network. This middle layer is merely aggregating the traffic from end devices and passing it to the network backbone. For these aforementioned reasons, we classify Åbo Akademi (ÅA) network as a three layer network having *core*, *aggregation* and *edge* layer—and not the core, distribution and access layer.

A. ÅA Network: The Core Layer

The core layer in ÅA network is the backbone network that provides means for connectivity between distant buildings located throughout the city and to rest of the world. In Figure 4.1, we illustrate the architecture of Åbo Akademi backbone network and its devices.

It is apparent from the figure that Åbo Akademi network comprises a *firewall*, four *routing-switches* and links between them. Three routing-switches are providing connectivity to the aggregation layer and firewall, while fourth routing switch is used to communicate with rest of the world. The description of the objects in Figure 4.1 is as following:

Network Clouds: there are four network clouds. `TuY/TKKK` cloud represents the university of Turku network. The second cloud is `Funet` that provides ÅA network the gateway to Internet. The outside world traffic of ÅA network passes through this pathway. The third cloud represents the `Syduast` network. These three clouds are connected over 1000BASE-LX links. Finally, we have the *Demilitarized zone* (DMZ).

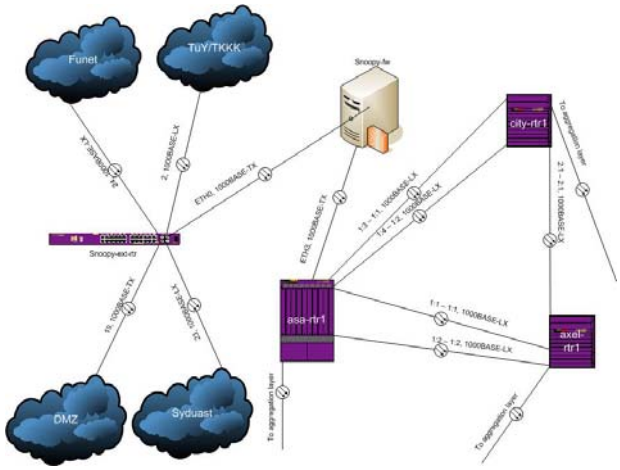


Figure 4.1: ÅA network backbone

Routing-switches: the backbone has four routing switches. The `snoopy-ext-rtr` is the end routing-switch which is providing connectivity to rest of the world. The other three routing switches form a ring in between them. This ring provides connections to all the aggregation layer devices in ÅA network.

Firewall: `snoopy-fw` is a heavy duty server computer that runs checkpoint firewall software to implement organizational policies and to protect Åbo Akademi network from malicious network attacks. All the traffic going out to Internet or coming in from Internet passes through this firewall

B. ÅA Network: The Aggregation and Edge Layers

Every building of Åbo Akademi has a separate aggregation switch with some exceptions when several buildings located in close proximity use a shared aggregation switch. For example, `DataCity` and `Korposträm` buildings do not have their own aggregation switches. These buildings share `BioCity` aggregation switch. All the aggregation switches finally terminate in one of the three ring forming core layer devices named `city-rtr1`, `asa-rtr1` and `axel-rtr1`. On the other side, these aggregation switches provide connections to the edge layer switches. Several edge layer switches together cover an area/building of Åbo Akademi. For simplicity, we do not present the full details of the network here. We refer to the original work for the complete details.

V. Problems, Solutions and Simulations

Due to the large number of devices and continuous expanding size of the network, ÅA network has accumulated complexity in the structure. We have sought to find out the structural and design problems in the network especially related to efficiency. Upon analyzing the ÅA network we have uncovered four main areas of potential problems: the naming convention, the extended hierarchy, the uplink bottlenecks and the (lack of) fault tolerance. Here we describe our findings, suggest some solutions, and present the simulated results of current and suggested scenarios via OPNET IT Guru Academic Edition. The simulations are presented for all but the first problematic area, as it was not applicable to this.

A. Naming convention

Network device names can either be user-friendly or not. When we talk about user-friendliness, we refer to the ease of utilization. The major service offered by a name is the identification of an object, hence the user-friendliness in names essentially means how easy the identification of an object is, whose name is being addressed. This identification may be in several aspects, for example, identification of hardware, tasks, geographical location, location in network plane etc.

Mainly, network device names can either be arbitrary or containing precise information in clear form or some kind of encoded form. For the aggregation switch at `biocity` building of ÅA network is named `bkf-esw2`. This name does not seem to be conveying much information about the type, job and place in hierarchy of Åbo Akademi network.

If we want to be precise in conveying information through names, we have to assign it a name, for example, like *`eswbiofloor1-bkf-esw3-city-rtr1.abo.fi`*. Now this name tells us that this is an edge switch `<esw>` at `Biocity <bio>` first floor `<floor1>` connected to aggregation layer switch `bkf-esw3` which belongs to `city-rtr1` hierarchy. But there is a problem with this kind of unstructured and non-planned naming scheme. The information relays to

everyone. Network personnel would not like that everyone just knowing the name of a device gets to know about all of its information. The simple solution is to encode this information in a way that only concerned personnel can decode. For this purpose we have to develop a naming convention. An example naming convention may be like the following:

```
<type><building><wiring closet><device number>-<aggregation tree>-<core tree>.<domain>
```

The elaboration of variables in this naming convention is given in Table 5.1

The name of above mentioned device now becomes *eswty6204103-bct-ct.abo.fi*. Because we have fixed the length and type of variables, the absence of any delimiter in first portion does not create an ambiguity. We can still tell that first three characters tell this is an edge layer (esw) switch at Biocity (ty6). The next four digits tell us the wiring closet along with the device number shown from next two digits. After that we used some delimiters. Using delimiters or not using them is just a matter of choice. One may use them throughout the name, and other may design a convention without them.

TABLE 5.1
DESCRIPTION OF NAMING VARIABLES

Name	Length and Type	Description
type	Three characters	It is the type of device, a device can be for example esw: edge switch asw: aggregation switch cor: core switch rou: router
building	Three alphanumeric	It is the name of the building in which the device is geographically located. A table can be created for building codes using either name initials like bc for Biocity or with some other information, for example, the address of Biocity(Tykistökatu 6): the code can then be ty6.
wiring closet	Four digits	It is the number of the wiring closet in which this device is physically located.
device number	Two digits	A wiring closet may have multiple devices. So these devices can be numbered to keep the names unique.
core tree	Two alphabets	ct: tree rooted at city-rtr1 as: tree rooted at asa-rtr1 ax: tree rooted at axel-rtr1
aggregation tree	Three alphabets	Every tree rooted at one of the core layer devices can be sub-divided into small aggregation level trees. These trees can be coded as bct: tree rooted at aggregation layer switch at Biocity. ict: tree rooted at aggregation layer switch at ICT-huset gad: tree rooted at aggregation layer switch located at Gadolinia building
domain	Variable	This is of course abo.fi in our case

Another advantage is that, by fixing the type and length of variables in the name, automated scripts using wild cards can be run very efficiently throughout the network. For example, a script meant to do a task on all the aggregation layer switches in Åbo Akademi network can use asw* as a parameter. Now it will address all the aggregation layer switches (leaving all other devices). Currently, the aggregation switch at Arken is arken-bdk-1-1-esw1, at Gadolinia it is gado-esw1 and at Biocity it is bkf-esw3. With current names there is no mechanism of using wild cards, but instead one has to write down all the names separately.

Having such long encoded names may seem like a questionable choice. However, this approach encodes the device names and achieves the double benefit of hiding obvious information from hackers/crackers as well as displaying necessary information for network personnel. The level of encryption here is a matter of choice. The more difficult the codes are, the tighter the security level is, hence the difficulty in decoding the device information. Therefore, a moderate choice in difficulty seems suitable. For an overview of naming conventions deployed in different university campuses, see 17], [HYPERLINK \l "Rut08" 18], 19].

B. Extended Hierarchy

There are several paths in Åbo Akademi network where the structuring goes beyond the conventional three layer architecture: this really increases the complexity of the network structure. Namely, some edge layer switches extend the hierarchy. This introduces a fourth level in the network, connected to edge layer and residing beneath it. On some paths the depths of hierarchy goes to even fifth level. The network segment under `gripen-esw1` switch is one such example. In Figure 5.1, we show the depth of hierarchal structure under `gripen-esw1`.

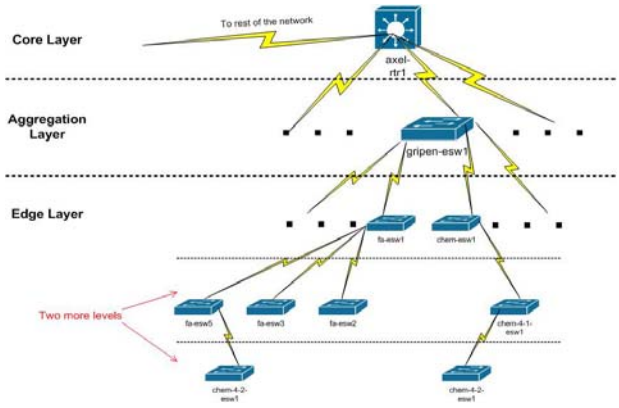


Figure 5.1: Extended hierarchy under `gripen-esw1`

Extending the hierarchy in this way is a bad network design practice. A good network design should not go beyond a third level [HYPERLINK \l "Joh01" 20]. The more processors, queues, memories and processing one gets involved in the path, the more increasing the delays. If an extension is needed, i.e. a new switch has to be added to the network, it should be given connection from an aggregation layer switch, instead of an edge layer switch.

The aggregation layer switches under which this kind of structure has been build in the Åbo Akademi network are `gripen-esw1`, `gado-esw1`, `bib-esw1`, `humanisticum-esw1`, `domus-esw1`, `axel-esw1`, `bkf-esw3` and `ict-bd002-esw1`. This comprises a major part of Åbo Akademi network's aggregation layer.

Suggested Solution, Simulations and Results

In Figure 5.1, we have shown a part of the Åbo Akademi network where he hierarchy goes to a depth of fifth level. We have simulated this model in OPNET IT Guru with two scenarios, one showing the original hierarchy, and the other built on a proposed strict three level hierarchy. The second scenario which is built for a strict three level hierarchy places switches of 4th and 5th level directly under `gripen-esw1`. The purpose of this simulation is to verify that increasing the depths of hierarchy induces unnecessary delays in network traffic. Figure 5.2 shows the models built for this purpose.

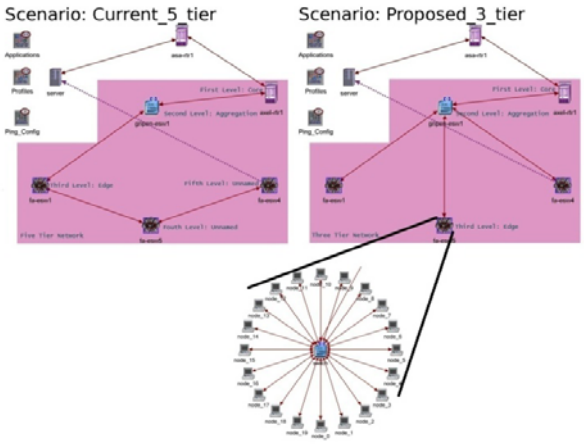


Figure 5.2: Extended hierarchy: simulated scenarios

In the left part of Figure 5.2, the original hierarchy is shown. On the right hand, the proposed hierarchy is modeled, namely a 3tier hierarchy.

On the top, two routing switches named `asa-rtr1` and `axel-rtr1` are installed to build up a core layer. These are BlackDiamond 6808 and Alpine 3808 respectively. The device models chosen are the same as installed in the real Åbo Akademi network. Then we have an aggregation layer switches named `gripen-esw1` which is an HP ProCurve 4000M

Then there are edge layer LAN segments comprising three switches and 20 client computers connected to each switch. There is a server which is connected to `asa-rtr1`. This server computer is receiving requests from the clients on LAN and sending the replies back. All the links in this simulation are 1Gbps, except the links between workstation and the switches. Links are configured without any background link utilization to find out the maximum effect of extending hierarchy. If there we find the delays in this scenario where links are totally free—there is no background utilization—then there would be more delays when links would be in heavy usage.

The objects `Applications`, `Profiles` and `Ping_Config` are used to define the type, flow, timing and amount of traffic. The object `Applications` actually defines the applications which are in use on this simulated network. It can be used to define a number of applications. For this, and the others simulation in next sections, we have chosen a few applications randomly. The object `Profiles` defines the way an application is used to generate traffic by this particular scenario. And the object `Ping_Config` is used to define ping packets generation and the behavior of the ping program, i.e., the interval between packets and repetitions etc. There is also a direct link between `fa-esw4` segment and the `server`. It is not a physical link. It is instead a logical link dedicated to defining the flow of the ping traffic as specified by the object `Ping_Config`.

After running these scenarios for 30 simulation minutes, we gathered the statistics for three different values namely Ethernet delay, ping response time and Database entry response time. These statistics are gathered for an end system `node_1` on `fa-esw4` segment. They show the difference in above mentioned statistics for when `fa-esw4` is directly connected to aggregation layer, and when it is connected to `fa-esw5`, which connects to aggregation layer via `fa-esw1`.

Ethernet Delay

The difference in Ethernet delay in both cases is significant. The delay figures are given in milliseconds that are not insignificant because engineers are researching ways to reduce nanosecond delays21].

In Figure 5.3, we show the difference between a 3tier and a 5tier topology.

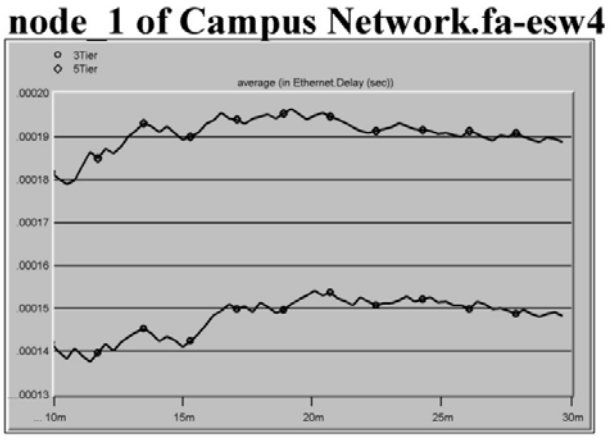


Figure 5.3: Extended hierarchy: Ethernet delay

The delay in Ethernet traffic in the case of a 3tier topology is measured around 0.15 milliseconds on average and it keeps around 0.19 milliseconds in the case of a 5tier topology. One thing to notice here is that the traffic load and the received traffic for both topologies is configured to be the same. We illustrate this similarity in Figure 5.4.

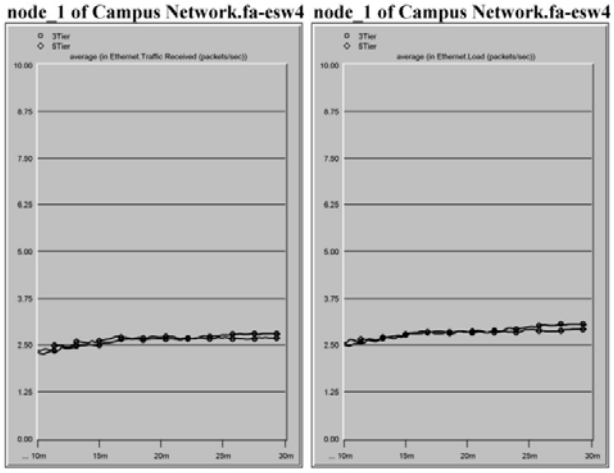


Figure 5.4: Extended hierarchy: Ethernet traffic and load

Ping response

The induced delays due to building an extended topology are also exhibited in the ping response time. The average delay for the 3tier topology is around 0.29 milliseconds, while the average delay in the ping response for 5tier topology is almost 0.39 milliseconds. We show the collected statistics in Figure 5.5.

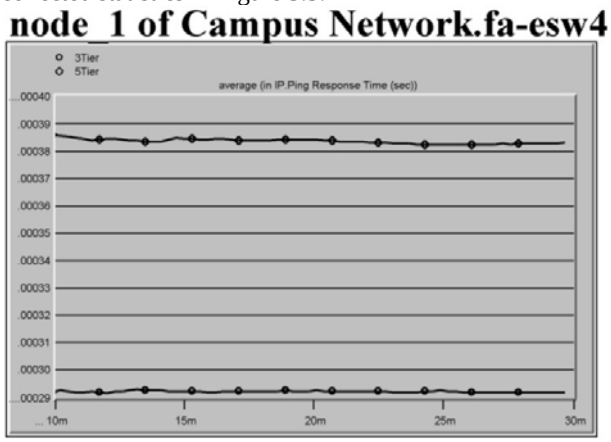


Figure 5.5: Extended hierarchy: ping response time

The ping traffic generated by the two scenarios is same. In Figure 5.6, we show the counts of ping request and ping reply packets exchanged between the server and the client.

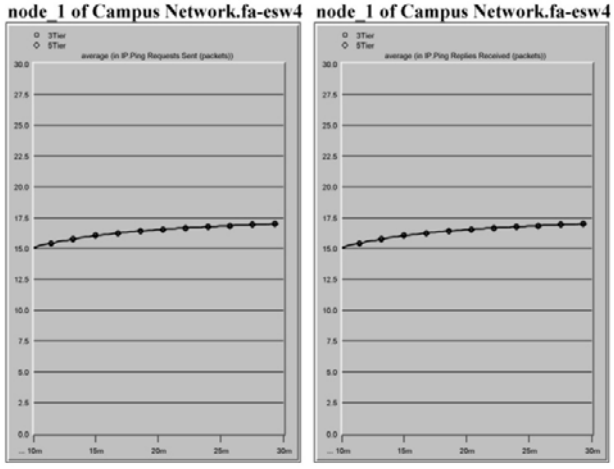


Figure 5.6: Extended hierarchy: ping traffic

Database Entry Response

The extended hierarchal structure also affects the performance of database entry packets. In Figure 5.7, we show the differences in the proposed and the current structure. The response time for a 5tier structure is 160 to 170 milliseconds. On the other hand, if `fa-esw4` is directly connected to aggregation layer, this response time drops to around 140 milliseconds. The difference of 20 milliseconds is a very big figure in terms of network response time.

The traffic sent and received for both scenarios is almost similar which keeps around 0.03 to 0.04 packets per second. We show this traffic patterns in Figure 5.8. The minor difference between the two traffic flows is not significant as it is less than one packet per second. The difference comes due to the generation of traffic based on the pseudo-random generators. Otherwise the configuration of the traffic for both scenarios is the same.

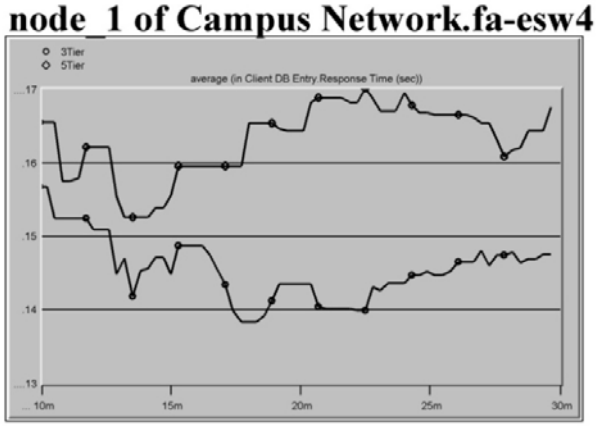


Figure 5.7: Extended hierarchy: DB entry response

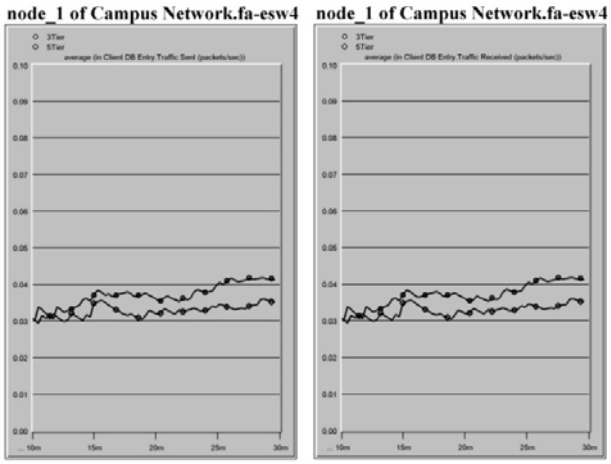


Figure 5.8: Extended hierarchy: DB entry traffic

It is very clear from the figures generated by the simulations that extending the hierarchy to a higher level induces unnecessary delays in network traffic. Therefore, the unnecessary extension of hierarchy should be avoided.

C. Uplink Bottlenecks

The concept of network traffic aggregation refers to the act of collecting several small links to a bigger link. Ideally, if one has to aggregate x network links of y Mbps capacity each, then the aggregating link should be capable of carrying $x \times y$Mbps data. However, not all the links are fully busy all the time and consequently, a general rule of aggregating the network traffic is to use the ratio of 3:1 [HYPERLINK \l "Joh98" 13] (e.g. 30 links of 100 Mbps each are aggregated into one link of 1000 Mbps, instead of one link of 3000 Mbps). Technological and economical limitations do not allow us to provide uplinks that aggregate the full capacity of a network segment. What the designers do is provide the best solution which falls in between cost effectiveness and quality effectiveness.

Åbo Akademi network at some places does not aggregate the links at all. The links being aggregated and the link which is aggregating are almost of same capacity. This situation simply creates a bottleneck. Consider a network segment of Åbo Akademi that we present in Figure 5.9. All the edge switches in `ICT-huset` are connected with aggregation layer switch

`ict-bd002-esw1` over a link of 1Gbps. This aggregation layer switch is further connected to `city-rtr1` over a link of bandwidth 1Gbps. The traffic from 22 switches is being aggregated at `ict-bd002-esw1` over 1Gbps links. Roughly this becomes 22Gbps capacity which is being aggregated and then transferred to `city-rtr1` over a link of 1Gbps. This presents a very high potential for bottlenecks.

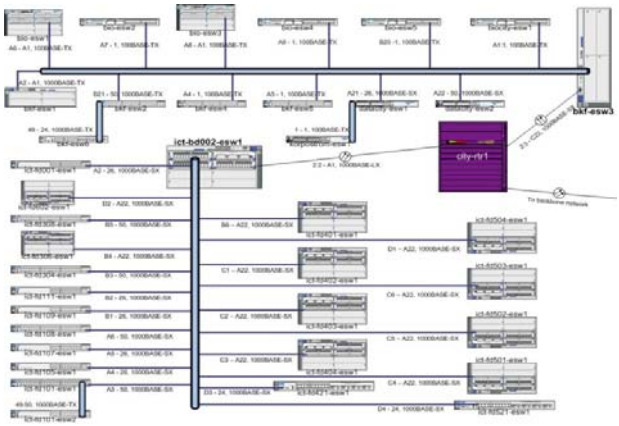


Figure 5.9: ÅA Network: city-rtr1 hierarchy

The solution to the problem is providing enough capacity in the aggregation links. If the clients are connected to edge switches over 100 Mbps, then the edge switch should be connected to aggregation layer over a link of at least 1000 Mbps, and so on. The backbone should be running at least ten times the speed of end stations20]. We now present what we have simulated as the suggested and the current scenarios for a segment of ÅA network.

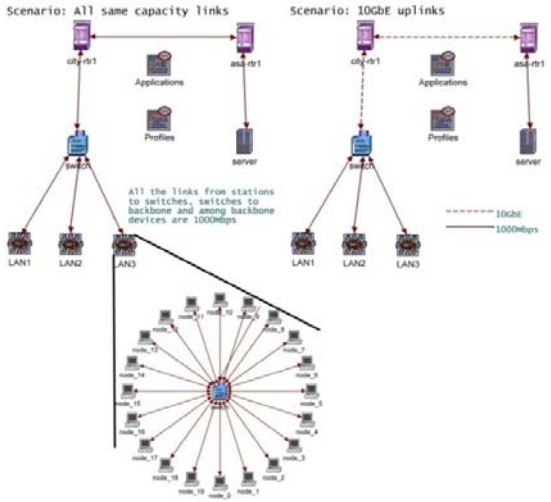


Figure 5.10: Uplink bottlenecks: scenarios

Suggested Solution, Simulations and Results

To show that an uplink of the same capacity as the links to the individual nodes creates a bottleneck and reduces overall network efficiency, we simulated a segment with two scenarios. The first scenario simulates all the links with the same capacity. The second scenario simulates uplinks with a bigger capacity as we have shown in Figure 5.10. On the left part, the scenario is similar to the current structure of Åbo Akademi network. All the links either connecting end stations or providing uplinks are of 1Gbps capacity. On the right part in the Figure 5.10, the proposed architecture is simulated. In this proposed architecture, the end stations are connected to switches over a link of 1Gbps capacity and the switches are connected to upstream switches over links of 10Gbps capacity. The objects `Applications` and `Profiles` are used to configure applications for end stations. In this simulation we configured database query traffic and http traffic. Both end stations generate email, ftp and http requests destined for the server.

After running this simulation for 30 simulation minutes, we have collected statistics for queuing delay, ftp traffic and the email traffic.

Queuing Delay on Links

The term Queuing delay refers to the time a chunk of data waits in a queue for its turn for transmission [HYPERLINK \l "Dav00" 22]. In Figure 5.11, we show the differences in Queuing Delay of two simulated scenarios. It is measured on the uplink between devices `switch` and `city-rtr1`. The difference in queuing delay of both scenarios is apparent in Figure 5.11. The proposed scenario has a queuing delay which is less than one microsecond, while the current scenario exhibits a queuing delay which keeps around 7 microseconds.

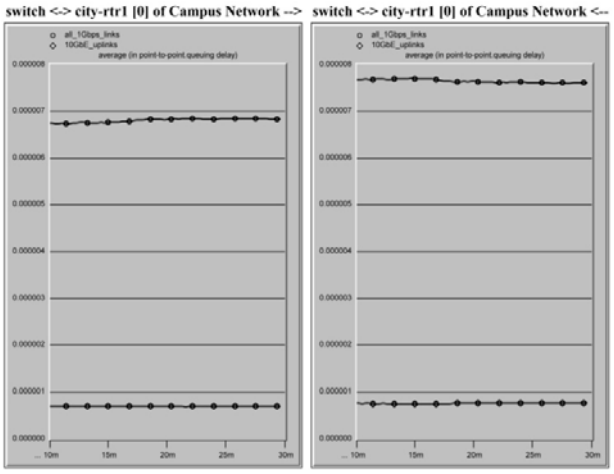


Figure 5.11: Uplink bottlenecks: queuing delay

Email Response Time

Because of the same `Profiles` and `Applications` objects used in both scenarios, the email traffic sent and received in both cases is the same. We have illustrated this in Figure 5.12. However, there are still differences in the response times of both scenarios, due to the reason that in first scenario the uplinks create a bottleneck. In Figure 5.13 and 5.14, we show the results collected from the simulation. The email upload response in the scenario with bigger uplinks is around 1.025 milliseconds, while it is around 1.125 milliseconds in the scenario with same uplinks. Similarly, the email download response time is around 1.025 milliseconds for the scenario with bigger uplinks, while it keeps around 1.1 milliseconds for the scenario with same uplinks.

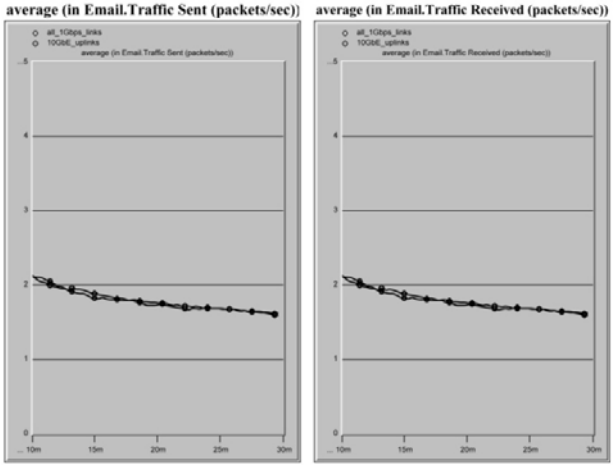


Figure 5.12: Uplink bottlenecks: email traffic

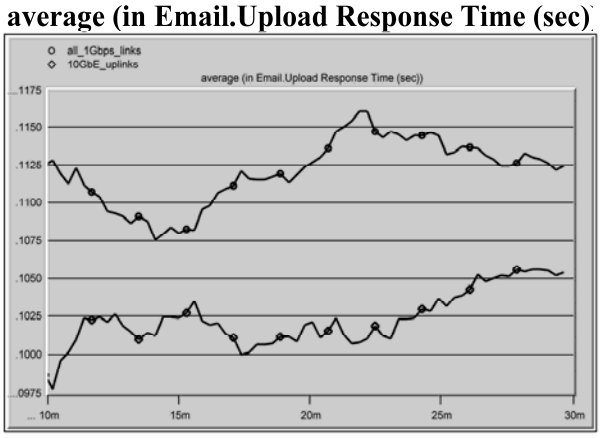


Figure 5.13: Uplink bottlenecks: email upload response time

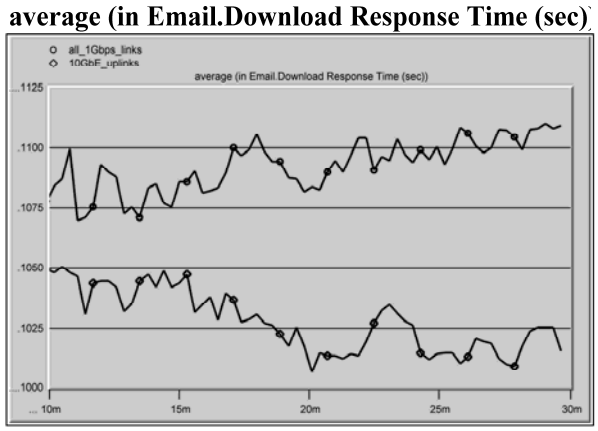


Figure 5.14: Uplink bottlenecks: email download response time

Ftp Response Time

The differences in both the upload and the download response time in ftp traffic are very significant. We show the difference in download response time in Figure 5.15. The response time of the two scenarios almost keeps 5 to 10 milliseconds of difference all the time. This difference is very noticeable and should be eliminated. Similarly, Figure 5.16 shows the difference of response time in ftp upload. The same behavior is exhibited here: the difference keeps between 5 to 10 milliseconds. The traffic for both scenarios is generated by the same `Profiles` and `Applications` objects, hence it is the same. We have illustrated this in Figure 5.17.

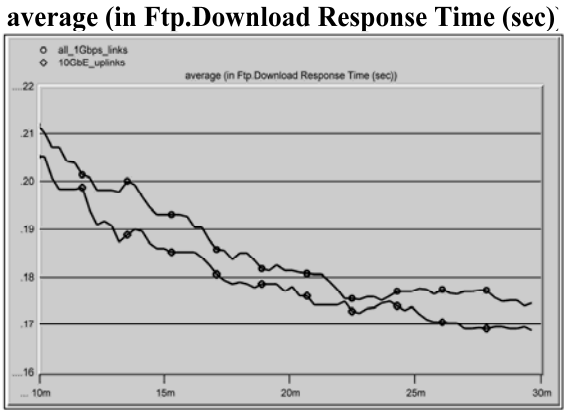


Figure 5.15: Uplink bottlenecks: ftp download response time

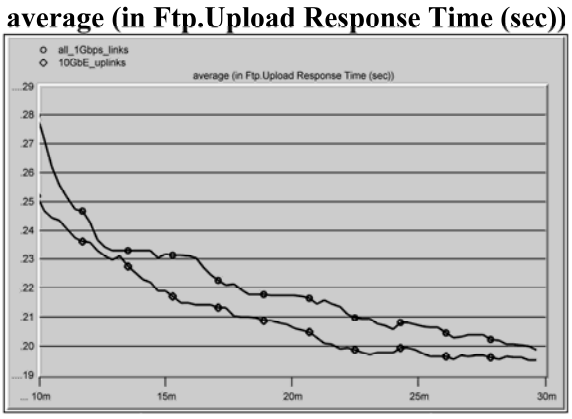


Figure 5.16: Uplink bottlenecks: ftp upload response time

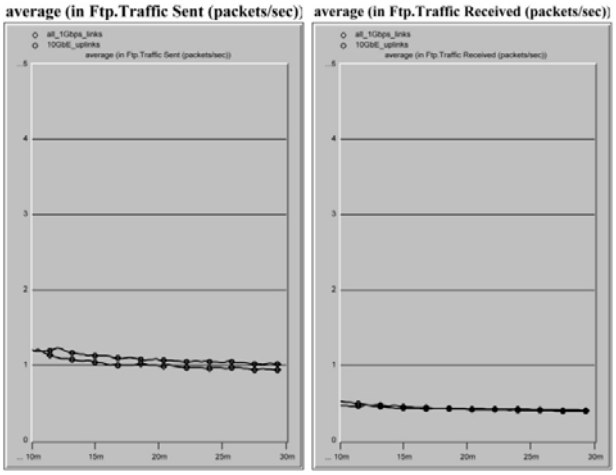


Figure 5.17: Uplink bottlenecks: ftp traffic

D.Fault tolerance and Redundancy

Essentially, fault tolerance refers to the ability of a network to resist against the various kinds of problems ranging from hardware failures to virus attacks and hacking attempts23]. It is one of the most crucial considerations of a communication networks [HYPERLINK \l "Sea01" 24]. To introduce fault tolerance in a network, the first thing to do is identifying single points of failure in the network. A *single point of failure* is a point in a network whose failure can bring the whole network down. These single points of failure are eliminated mostly through redundancy to keep the network from going down in case of any fault occurrence at this single point of failure25]. This makes the network very much transparent namely to the users, if a fault occurred at a single point of failure where no redundancy was deployed, it will bring the whole network down, disconnecting all the users from their communications. While, if redundancy was deployed at that point, the communication will automatically shift to the redundant resources, leaving user's communications intact and unaware of any failures.

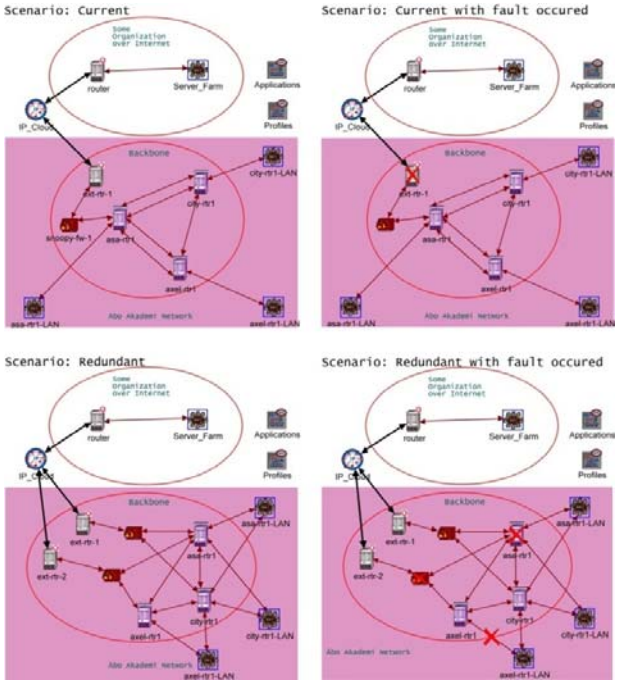


Figure 5.18: Simulation scenarios: fault tolerance

In practice, it is not sufficient to eliminate the single points of failure for the entire network. In addition, other points of failure also need to be eliminated for keeping a major part of the network from going down. Consider the failure of `city-rtr1` in Figure 5.9: this would not let the whole Åbo Akademi network down, but still it would disturb a major portion of the network and its users. Hence, it would be useful to eliminate this point of failure as well as other points of failure like this in the network.

Åbo Akademi network has only one place in the entire network where redundancy is deployed. This redundancy is in links that connect `city-rtr1` and `axel-rtr1` to `asa-rtr1`. There is no redundancy at devices at all; there is no redundancy at links elsewhere at all. Practically, there is no fault tolerance in the network. The de facto solution to this fault tolerance problem is creating a partial mesh [HYPERLINK \l "Joh01" 20].

Suggested Solution, Simulations and Results

In Figure 5.18, we illustrate out suggested solution for the problem of fault tolerance with redundant devices and a partial mesh of links between them. The two upper scenarios in Figure 5.20 are based on the current situation of the network. In the left part, the scenario simulates the situation in which there is no failure in the network. In the right part, the scenario is showing the state when there is a fault occurrence at `ext-rtr-1` device. In the lower two scenarios, the suggested solution is implemented with no fault occurrence at left, and with simultaneous fault occurrence at two devices and one link on the right side.

All of the four scenarios contain similar devices and links between them. The configuration of `Applications`, `Profiles` and thus the traffic is the same. The simulation was run for a total duration of 30 minutes. We now show the result of these simulations.

Effect of Failure on Devices

The first two scenarios that model the current network show the effect of the failure of a device. The last two scenarios show the effect of the failure of two devices and one link. In Figure 5.19, we show the counts of IP traffic sent and received through the edge router in all four scenarios.

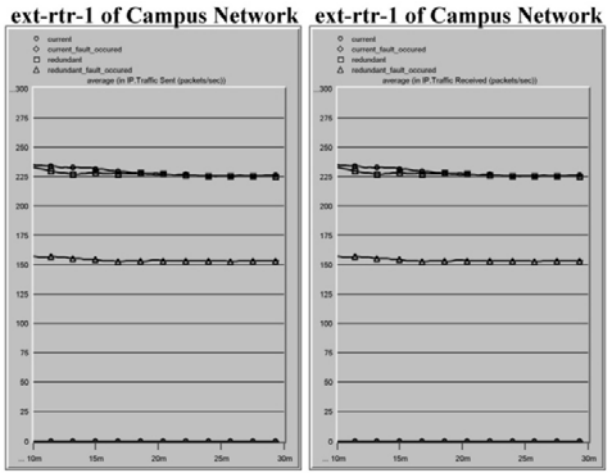


Figure 5.19: Fault tolerance: IP traffic

In the case of current scenario without a fault the traffic sent and received is around 225 packets per second. But when a fault occurred in the current scenario, the traffic sent and received goes to zero packets per second. It shows that the entire network is no more able to communicate with the Internet. One the other hand, in suggested scenarios, when there is no fault, the traffic sent and received is just the same as in the case of current network without a fault occurrence. But as soon as a fault occurs in the suggested scenario, the traffic drops to around 150 packets per second. This is due to the reason that in the case of a fault occurrence, the devices have to converge once again and calculate new paths. During this time, the communication hangs. But still this process is transparent to the edge devices.

Effect of Failure on Links

The effect of the failure on links in the current network is fatal. If a device goes down the main links become idle. In Figure 5.20 and 5.21, we show the effects of the failure of `ext-rtr-1` on the link between `ext-rtr-1` and the `ip_cloud` which is representing the Internet. The figures illustrate that in the case of a fault in current the scenario, the utilization and the throughput of the link becomes zero, as there is no path for the traffic. On the other hand, if the fault occurs in the suggested scenario, the utilization and throughput goes slightly down, because of the aforementioned fact that the network needs to converge again. During this time—which is a matter of fraction of time—the communication is disrupted. So the overall average utilization and throughput of the links goes down.

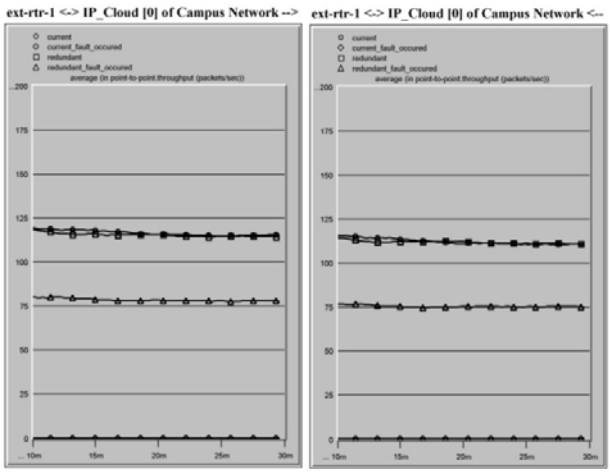


Figure 5.20: Fault tolerance: throughput of link

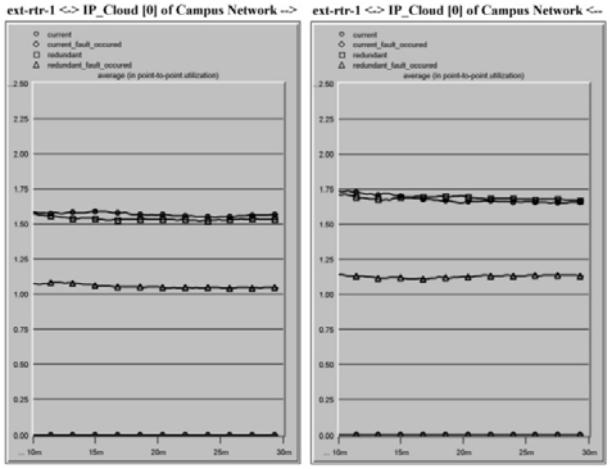


Figure 5.21: Fault tolerance: utilization of link

Effect of Failure on Overall Network Performance

We have recorded the statistics for email applications in the scenarios. In Figure 5.22 and 5.23, we illustrate that there are no statistics for the current scenario with the occurrence of a fault. This is due to the reason that the current scenario has no tolerance against fault.

On the other hand, if a fault occurs in the suggested scenario, the response time becomes almost equal to the current scenario. The response time of current scenario with no fault and the suggested scenario with fault occurred is the same because both scenarios are now running on a single link and a single connecting device to the Internet. While, in the case of the suggested scenario with no fault occurrence, the response time is less because there are two connecting links and devices to the Internet. Due to this fact, the performance of the application is better.

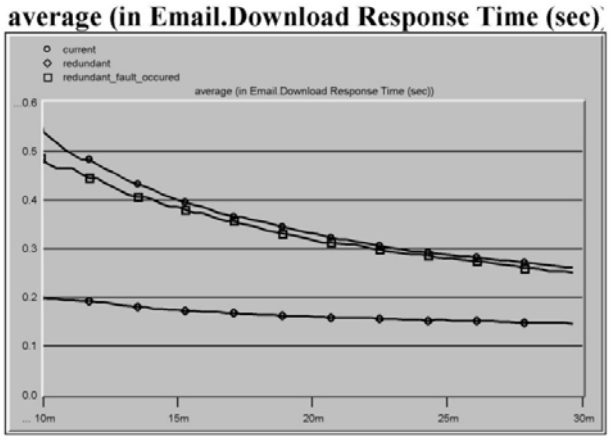


Figure 5.22: Fault tolerance: email download response time

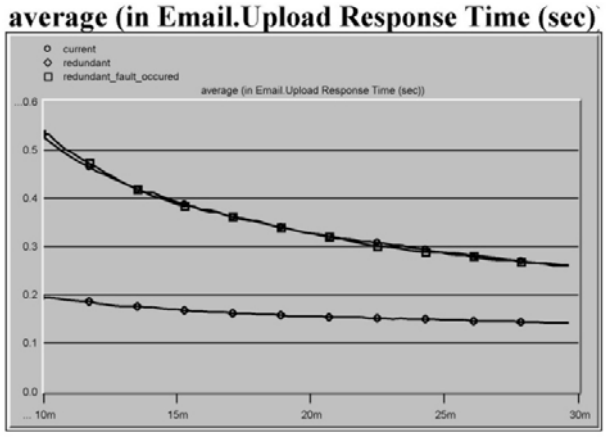


Figure 5.23: Fault tolerance: email upload response time

VI. Proposed Network Design

We have presented in the previous sections that Åbo Akademi network has several problems including the naming convention, the extended hierarchy, the uplink bottlenecks and the (lack of) fault tolerance. Here we propose a backbone architecture that can solve these problems.

The first aspect is the solution of bottlenecks. The proposed solution's entire backbone is built on a 10GbE technology. The second aspect is addition of fault tolerance. This architecture proposes a partial mesh topology to eliminate the points which can bring the whole network down in case of their own failure. A new firewall structure is proposed to eliminate the bottlenecks and single point of failure created by the current firewall setup. A new external router is added to eliminate the chances that the failure of a single external router would cut off the network from the rest of the world. We illustrate this design in Figure 6.1.

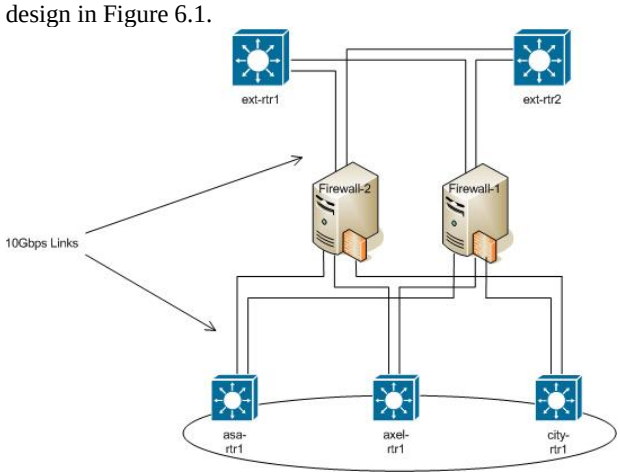


Figure 6.1: proposed backbone architecture

Fault tolerance is not the only aspect that is to be acquired through redundancy. Redundancy is also used to enhance the performance. Current network standards and protocols allow us to use redundant device to balance network traffic load. For example, the network load can be equally shared between both of the firewalls, and if one firewall is down, the other can alone handle all the network traffic coming in and going out.

As we have noted before, current Åbo Akademi network at several places violates the conventional layered network architecture by extending the hierarchy to fourth or fifth level. This is totally unacceptable according to a designer perspective. The second major problem at aggregation layer in the current network is the lack of fault tolerance. In Figure 6.2, we present the proposed aggregation and edge layer architecture that can solve these problems.

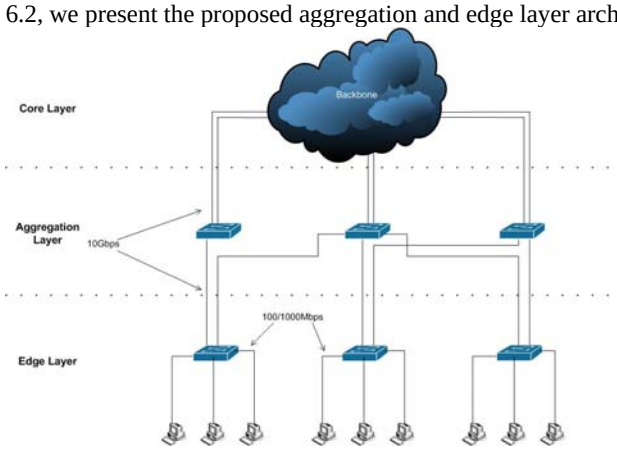


Figure 6.2: proposed aggregation and edge layers

In the proposed design, the failure of a single link or switch at aggregation layer can be of no harm to the network. It would be even unnoticeable by users. It is because if one link fails, there is always another link to keep communication uninterrupted. And if a switch at aggregation layer fails, there is always another connection to another switch. The only points of failure left in the network are at edge the layer now. These points are not eliminated through redundancy because

there scope is very limited. A failure of an edge switch can affect only a number of computers which are connected to it. It can not affect a considerable part of the network, as in the case of a failure at core layer or aggregation layer.

VII. Conclusions and Future Work

The Åbo Akademi network is a switched network built on Ethernet technology. It was established many years ago and had considerably grown during the past years. Our analysis in this work shows that the network has accumulated some complexity. We can speculate about the reasons contributing to this complexity, such as arbitrary growth, limited resources, lack of stringent security needs, and of course, the fact that its main skeleton was designed many years ago. The network's problems revolve around the issues like switched network architecture, aggregation of bandwidth, the extension of hierarchy, bottlenecks, lack of fault tolerance, naming conventions and the old technology backbone. All of these constitute obstacles for having an efficient network.

Our suggested solution and the results of the simulations justify that the backbone should be upgraded to meet the requirements of the new resource hungry applications. Second, further depths in the hierarchical structure currently exist. Simulations and theory show that these further depths in the hierarchal structure exhibit efficiency problems in switching networks. So, the suggested solution is to remove these extensions and connect the hierarchy extending switches directly to the aggregation layer.

One of the biggest problems in the network is the lack of fault tolerance. We have suggested a partial mesh topology with the addition of some devices to introduce fault tolerance. Simulations of the suggested scenario show that our proposed solution does not only introduce fault tolerance but also introduces performance enhancement in the network. Along with this, the presence of a single firewall at the end of the network is another point to reconsider. This device is a dangerous single point of failure in the network. It is also creating a bottleneck because all the traffic from inside the network destined to outside world has to pass through it. We have proposed to install another firewall device. This will facilitate the network with better performance because of the load sharing between two firewalls and with fault tolerance as well, in the case of a device failure.

Another problem is the absence of a structured naming convention. This problem is very crucial on the administrative side of the network. It makes the network more person dependant. We have suggested some naming conventions which can be deployed in the network. Some examples of such naming conventions are quoted from different universities networks.

On the whole, we conclude that there are several problems in the Åbo Akademi network. Some of these problems are very crucial in their nature and demand an immediate solution. We have also proposed solutions to the identified problems and have justified our suggested solutions via simulations.

Future Work Besides the suggested solutions for improving the efficiency of the ÅA network, we see future work on the topic more on the logical level. We plan to investigate the real traffic patterns being originated from the edge of the network as well as the traffic circulation in the network. Analyzing this type of information would allow us to gain a more concrete insight of the bottlenecks and problems in the network.

Along with this, we intend to work on the VLAN architecture deployed in the network. The major issues in this direction would be the definition points of VLAN as well as the inter-VLAN routing. This study will give us important insight into bringing the network to a stricter core-distribution-access type of design.

References

[1] R. Breyer and S. Riley, *Switched, Fast, And Gigabit Ethernet* , 3rd ed. , USA : Macmillan Technical Publishing , 1999.

[2] W. Stallings, *Data and Computer Communications*, International ed. New Jersey, USA: Prentice-Hall, Inc., 1997.

[3] A. S. Tanenbaum, *Computer Network*, 3rd ed. New Jersey, USA: Prentice-Hall, Inc., 1996.

[4] CISCO Systems, Inc., Internetworking Technology Handbook - Ethernet. [Online]. http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Ethernet.html

[5] F. Halsall, *Data Communications, Computer Networks and Open systems* , 4th ed. USA: Addison-Wesley Publishing Company , 1996.

[6] CISCO Systems, Inc., Internetworking Technology Handbook - LAN Switching. [Online]. http://www.cisco.com/en/US/docs/internetworking/technology/handbook/LAN-Switching.html

[7] IEEE, IEEE Standards for Local Area Network. [Online]. http://ieeexplore.ieee.org/xpls/abs_all.jsp?tp=&isnumber=1002&arnumber=26520&punumber=2565

[8] (2008, Mar.) IEEE, IEEE P802.3ba Objectives. [Online]. http://grouper.ieee.org/groups/802/3/ba/PAR/P802.3ba_Objectives_0308.pdf

[9] W. Stallings, *High Speed Networks: TCP/IP and ATM Design Principles* . New Jersey , USA: Prentice-Hall, Inc., 1998.

[10] T. Boyles and D. Hucaby, *CCNP Switching Exam Certification Guide*. Indianapolis, USA: Cisco Press, 2001.

[11] CISCO Systems, Inc., Internetwork Design Guide - Appendix E: Broadcasts in Switched LAN Internetworks. [Online]. http://www.cisco.com/en/US/docs/internetworking/design/guide/nd20e.html

[12] CISCO Systems, Inc., Internetwork Design Guide - Internetworking Design Basics. [Online]. http://www.cisco.com/en/US/docs/internetworking/design/guide/nd2002.html

[13] J. J. Roese, *Switched LANs: Implementation, Operation, Maintenance* . USA: McGraw-Hill, 1998.

[14] CISCO Systems, Inc., Internetwork Design Guide - Introduction [Internetworking]. [Online]. http://www.cisco.com/en/US/docs/internetworking/design/guide/nd2001.html

[15] K. D. Stewart III and A. Adams, *Designing and Supporting Computer Networks* . Indianapolis , USA: Cisco Press , 2008.

[16] W. Lewis, *Multilayer Switching Comapnion Guide* . Indianapolis, USA: Cisco Press , 2003.

[17] University of Waterloo, Network device naming standard. [Online].

https://strobe.uwaterloo.ca/~twiki/bin/view/ISTNS/NetworkDeviceNamingStandard
[18] Rutgers: The State University of New Jersey, Rutgers: Telecommunications Division. [Online]. http://www.td.rutgers.edu/documentation/Reference/RUNet_Network_Device_Naming_Convention/index.html
[19] University of Maryland, Baltimore. [Online]. www.umaryland.edu/cits/docs/Campus%20Device%20Naming%20Conventions.doc
[20] J. Swartz and T. Lammle, *CCIE : Cisco certified internetwork expert : study guide.* San Francisco , USA: Sybex, 2001.
[21] D. Sadot and I. Elhanany, "Optical switching speed requirements for terabit/second packet overWDM networks," *Photonics Technology Letters, IEEE*, vol. 12, no. 4, pp. 440-442, Apr. 2000.
[22] J. Davidson, J. Peters, and B. Gracely, *Voice over IP fundamentals*. Indianapolis, IN, USA: Cisco Press , 2000.
[23] S. Mueller and T. W. Ogletree, *Upgrading and Repairing Networks*, 4th ed. Indianapolis, IN, USA: Pearson Education , 2003.
[24] S. Odom and H. Nottingham, *Cisco switching black book*. Scottsdale, AZ, USA: Coriolis Group Books, 2001.
[25] S. M. Ballew, *Managing IP networks with Cisco routers*. Sebastopol, CA , USA: O'Reilly & Associates, 1997.

Random Walk Gossip: A Manycast Algorithm for Disaster Area Networks

Mikael Asplund, Simin Nadjm-Tehrani
Department of Computer and Information Science, Linköping University
SE-581 83 Linköping, Sweden
{mikas,simin}@ida.liu.se

When the communication infrastructure is needed the most - in the event of a disaster - it is the most likely that it is not available. We suggest that rescue personnel working in such conditions can be supported by networking protocols which are tolerant to disconnectivity and unstructured topologies. Moreover, since communication devices will probably be battery driven and power is not easily available, protocols need to very restrictive in communication to save power. In particular, we are interested in reliable multicast operations in which a sender wants to send a message that can be relied upon to reach at least a portion of a certain group of receivers (i.e., manycast).

Epidemic algorithms in mobile networks can be broadly categorised as using localised gossipping [2] or anti-entropy [4]. Both mechanisms have drawbacks; while the former approach suffers from a complicated balance between wasting resources and the risk of messages not being propagated, the latter provides full coverage but generally results in slow propagation as well as a high bandwidth usage. Recently, Khelil et al. [3] proposed to use a combination of these approaches called hyper-gossipping to achieve best-effort broadcasts in partitioned networks.

We believe that cooperation in post-disaster areas requires a new kind of protocol which is efficient (short delay, low bandwidth), capable of dealing with disrupted communication, reliable, and which does not require knowledge about which nodes are currently operating in the network.

In this abstract we describe the basics of a protocol called Random Walk Gossip (RWG), described earlier and experimentally evaluated [1]. This protocol meets the need for energy-efficient reliable communication in an intermittently connected environment. The protocol relies heavily on hashing of node addresses instead of keeping track of all the nodes in the system. This way we take the middle way between best-effort algorithms requiring no knowledge at all and fully reliable protocols requiring full knowledge. RWG terminates when at least k nodes will be reached by the message.

The protocol has two modes: gossipping and waiting. During the gossipping phase, the message spread in the network. If a holder of a message (custodian) detects that the network is partitioned, it puts the message on hold. This will cause nodes to be silent when no new nodes can be reached, and thus reducing energy consumption. Eventually, the node will discover that uninformed nodes are nearby and resume propagation of the message.

Algorithm 1 outlines the basic behaviour during the gossipping phase of the algorithm. Each message has a bit vector (*informed*) whose role it is to keep track of the nodes that have received the message. The index of each node is decided by taking the hash of the node ID. For example, the vector $[0, 1, 0, 0, 1, 0]$ indicates that the nodes with hash 2 and 4 have received the message.

Algorithm 1 Random walk gossip

When a message m is heard by node i from neighbour j:
  wake up all messages not seen by j
  set *m.informed[hash(i)]*=1
  if forwarder:
    Send m to a random neighbour (if possible)

When k nodes have been reached (easily seen in *m.informed*):
  stop forwarding and propagate $beSilent(m)$

A message is made inactive if there is no neighbour that can forward it, thereby indicating a possible network partition. The decision to make a message inactive is specific to that message alone, since other messages might already have seen the parts of the network which are deemed to be currently unreachable. As soon as a node discovers a neighbouring node that has not seen the inactive message, it is reactivated, and the random walk starts anew.

The algorithm has been extensively evaluated in a simulation environment built on top of ns-3. Current work includes implementing the algorithm on a range of hand held devices (Linux-based tablets and laptops, as well as Android telephones).

Acknowledgements

This work was supported by the Swedish Civil Contingencies Agency (MSB) and the Swedish Research Council (VR).

References

[1] M. Asplund and S. Nadjm-Tehrani. A partition-tolerant manycast algorithm for disaster area networks. In *28th International Symposium on Reliable Distributed Systems*. IEEE, Sept. 2009.

[2] Z. Haas, J. Halpern, and L. Li. Gossip-based ad hoc routing. *IEEE/ACM Trans. Networking*, 14(3):479–491, June 2006.

[3] A. Khelil, P. J. Marrón, C. Becker, and K. Rothermelns. Hypergossiping: A generalized broadcast strategy for mobile ad hoc networks. *Ad Hoc Netw.*, 5(5):531–546, 2007.

[4] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical report, Duke University, 2000.

Modelling Fault Tolerance and Parallelism in Communicating Systems

Linas Laibinis[1], Elena Troubitsyna[1], and Sari Leppänen[2]

[1] Åbo Akademi University, Finland
[2] Nokia Research Center, Finland
{Linas.Laibinis, Elena.Troubitsyna}@abo.fi
Sari.Leppanen@nokia.com

Abstract. Telecommunication systems should have a high degree of availability, i.e., high probability of correct provision of requested services. To achieve this, correctness of software for such systems and system fault tolerance should be ensured. In this paper we show how to formalise and extend Lyra – a top-down service-oriented method for development of communicating systems. In particular, we focus on integration of fault tolerance mechanisms into the entire Lyra development flow.

1 Introduction

Modern telecommunication systems are usually distributed software-intensive systems providing a large variety of services to their users. Development of software for such systems is inherently complex and error prone. However, software failures might lead to unavailability or incorrect provision of system services, which in turn could incur significant financial losses. Hence it is important to guarantee correctness of software for telecommunication systems.

Nokia Research Center has developed the design method Lyra [6] – a UML2-based service-oriented method specific to the domain of communicating systems and communication protocols. The design flow of Lyra is based on the concepts of decomposition and preservation of the externally observable behaviour. The system behaviour is modularised and organised into hierarchical layers according to the external communication and related interfaces. It allows the designers to derive the distributed network architecture from the functional system requirements via a number of model transformations.

From the beginning Lyra has been developed in such a way that it would be possible to bring formal methods (such as program refinement, model checking, model-based testing etc.) into more extensive industrial use. A formalisation of the Lyra development would allow us to ensure correctness of system design via automatic and formally verified construction. The achievement of such a formalisation would be considered as significant added value for industry.

In our previous work [5, 4] we proposed a set of formal specification and refinement patterns reflecting the essential models and transformations of Lyra. Our approach is based on stepwise refinement of a formal system model in the

B Method [1] – a formal refinement-based framework with automatic tool support. Moreover, to achieve system fault tolerance, we extended Lyra to integrate modelling of fault tolerance mechanisms into the entire development flow. We demonstrated how to formally specify error recovery by rollbacks as well as reason about error recovery termination.

In this paper we show how to extend our Lyra formalisation to model parallel execution of services. In particular, we demonstrate how such an extension affects the fault tolerance mechanisms incorporated into our formal models. The extension makes our formal models more complicated. However, it also gives us more flexibility in choosing possible recovery actions.

2 Previous Work

In this section we give a brief overview of on our previous results [5, 4] on formalising and verifying the Lyra development process. This work form the basis for new results presented in the next section.

2.1 Formalising Lyra

Lyra [6] is a model-driven and component-based design method for the development of communicating systems and communication protocols, developed in the Nokia Research Center. The method covers all industrial specification and design phases from pre-standardisation to final implementation.

Lyra has four main phases: *Service Specification*, *Service Decomposition*, *Service Distribution* and *Service Implementation*. The *Service Specification* phase focuses on defining services provided by the system and their users. In the *Service Decomposition* phase the abstract model produced at the previous stage is decomposed in a stepwise and top-down fashion into a set of service components and logical interfaces between them. In the *Service Distribution* phase, the logical architecture of services is distributed over a given platform architecture. Finally, in the *Service Implementation* phase, the structural elements are integrated into the target environment. Examples of Lyra UML models from the Service Specification phase of a positioning system are shown on Fig.1.

To formalise the Lyra development process, we choose the B Method as our formal framework. The B Method [1] is an approach for the industrial development of highly dependable software. Recently the B method has been extended by the Event B framework [2, 7], which enables modelling of event-based systems. Event B is particularly suitable for developing distributed, parallel and reactive systems. The tool support available for B provides us with the assistance for the entire development process. For instance, Atelier B [3], one of the tools supporting the B Method, has facilities for automatic verification and code generation as well as documentation, project management and prototyping.

The B Method adopts the top-down approach to system development. The basic idea underlying stepwise development in B is to design the system implementation gradually, by a number of correctness preserving steps called *refinements*. The refinement process starts from creating an abstract specification

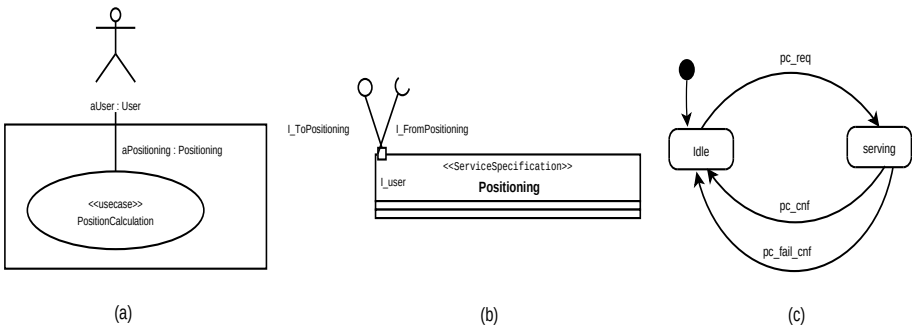


Fig. 1. (a) Domain Model. (b) Class Diagram of Positioning. (c) State Diagram.

and finishes with generating executable code. The intermediate stages yield the specifications containing a mixture of abstract mathematical constructs and executable programming artefacts.

While formalising Lyra, we single out a generic concept of a communicating service component and propose B patterns for specifying and refining it. In the refinement process a service component is decomposed into a set of service components of smaller granularity specified according to the proposed pattern. Moreover, we demonstrate that the process of distributing service components between network elements can also be captured by the notion of refinement. Below we present an excerpt from an abstract B specification pattern of a communicating service component.

The proposed approach to formalising Lyra in B allows us to verify correctness of the Lyra decomposition and distribution phases. In development of real systems we merely have to establish by proof that the corresponding components in a specific functional or network architecture are valid instantiations of these patterns. All together this constitutes a basis for automating industrial design flow of communicating systems.

MACHINE ACC

...

VARIABLES $\mathsf{in\_data}, \mathsf{out\_data}, \mathsf{res}$

INVARIANT $\mathsf{in\_data} \in \mathsf{DATA} \land \mathsf{out\_data} \in \mathsf{DATA} \land \mathsf{res} \in \mathsf{DATA}$

INITIALISATION $\mathsf{in\_data}, \mathsf{out\_data}, \mathsf{res} := \mathsf{NIL}, \mathsf{NIL}, \mathsf{NIL}$

```
EVENTS
input =
  ANY param
  WHERE param ∈ DATA ∧ ¬(param = NIL) ∧ in_data = NIL
  THEN
    in_data := param
  END;

calculate =
  WHEN ¬(in_data = NIL) ∧ out_data = NIL
  THEN
    out_data :∈ DATA − {NIL}
  END;

output =
  WHEN ¬(out_data = NIL)
  THEN
    res := out_data ||
    in_data, out_data := NIL, NIL
  END
```

A B specification, called an *abstract machine*, encapsulates a local state (program variables) and provides operations on the state. In the Event B framework[1], such operations are called *events*. The events can be defined as

WHEN g **THEN** S **END**

or, in case of a parameterised event, as

ANY vl **WHERE** g **THEN** S **END**

where vl is a list of new local variables (parameters), g is a state predicate, and S is a B statement describing how the program state is affected by the event.

The events describe system reactions when the given **WHEN** or **WHERE** conditions are satisfied. The **INVARIANT** clause contains the properties of the system (expressed as predicates on the program state) that should be preserved during system execution. The data structures needed for specification of the system are defined in a separate module called *context*. For example, the abstract type $DATA$ and constant NIL used in the above specification are defined in the context ACC_Data, which can be accessed ("seen") by the abstract machine ACC.

[1] This work has been done using the Atelier B tool, supporting the Event B extension

The presented specification pattern is deliberatively made very simple. It describes a service component in a very abstract way – a service component simply receives some request data as the input, non-deterministically calculates non-empty result, which is then returned as the output. Using this specification as the starting point of our formal development gives us sufficient freedom to refine it into different kinds of service components. In particular, both the service components providing single services and the service components responsible for orchestrating service execution (called service directors) can be developed as refinements of the presented specification. Moreover, the defined specification and refinement patterns can be repeatedly used to gradually unfold the hierachical structure of service execution.

The proposed approach to formalising Lyra in B allows us to verify correctness of the Lyra decomposition and distribution phases. In development of real systems we merely have to establish by proof that the corresponding components in a specific functional or network architecture are valid instantiations of these patterns. All together this constitutes a basis for automating industrial design flow of communicating systems.

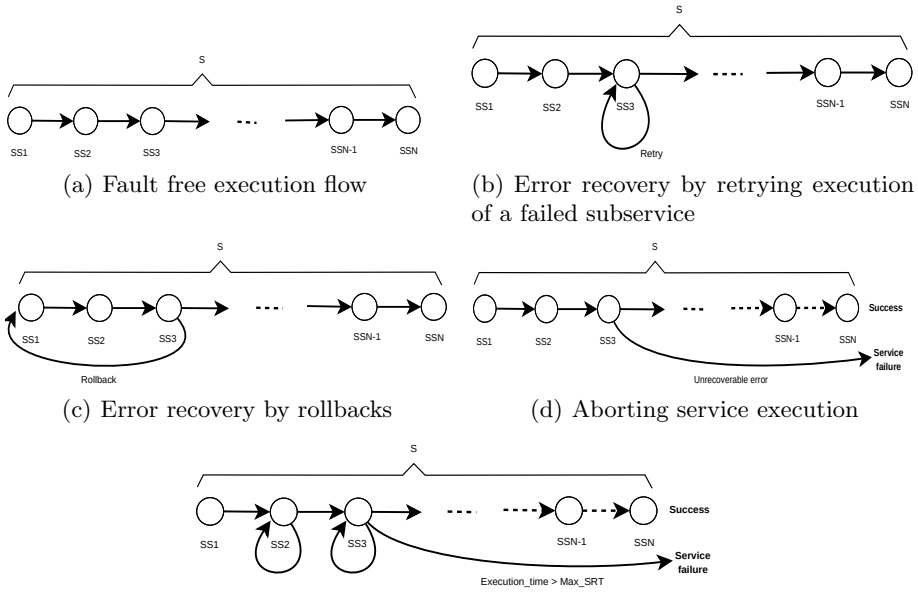
2.2 Introducing Fault Tolerance in the Lyra Development Flow

Currently the Lyra methodology addresses fault tolerance implicitly, i.e., by representing not only successful but also failed service provision in the Lyra UML models. However, it leaves aside modelling of mechanisms for detecting and recovering from errors – the fault tolerance mechanisms. We argue that, by integrating explicit representation of the means for fault tolerance into the entire development process, we establish a basis for constructing systems that are better resistant to errors, i.e., achieve better system dependability. Next we will discuss how to extend Lyra to integrate modelling of fault tolerance.

In the first development stage of Lyra we set a scene for reasoning about fault tolerance by modelling not only successful service provision but also service failure. In the next development stage – *Service Decomposition* – we elaborate on representation of the causes of service failures and the means for fault tolerance.

In the *Service Decomposition* phase we decompose the service provided by a service component into a number of stages (subservices). The service component can execute certain subservices itself as well as request other service components to do it. According to Lyra, the flow of the service execution is managed by a special service component called *Service Director*. *Service Director* co-ordinates the execution flow by enquiring the required subservices from the external service components.

In general, execution of any stage of a service can fail. In its turn, this might lead to failure of the entire service provision. Therefore, while specifying *Service Director*, we should ensure that it does not only orchestrates the fault-free execution flow but also handles erroneous situations. Indeed, as a result of requesting a particular subservice, *Service Director* can obtain a normal response



(a) Fault free execution flow

(b) Error recovery by retrying execution of a failed subservice

(c) Error recovery by rollbacks

(d) Aborting service execution

(e) Aborting the service due to timeout

Fig. 2. Service decomposition: faults in the execution flow

containing the requested data or a notification about an error. As a reaction to the occurred error, *Service Director* might

- retry the execution of the failed subservice,
- repeat the execution of several previous subservices (i.e., roll back in the service execution flow) and then retry the failed subservice,
- abort the execution of the entire service.

The reaction of *Service Director* depends on the criticality of an occurred error: the more critical is the error, the larger part of the execution flow has to be involved in the error recovery. Moreover, the most critical (i.e., unrecoverable) errors lead to aborting the entire service. In Fig.2(a) we illustrate a fault free execution of the service S composed of subservices $S_1, \ldots, S_N$. Different error recovery mechanisms used in the presence of errors are shown in Fig.2(b) - 2(d).

Let us observe that each service should be provided within a certain finite period of time – the *maximal service response time Max_SRT*. In our model this time is passed as a parameter of the service request. Since each attempt of subservice execution takes some time, the service execution might be aborted even if only recoverable errors have occurred but the overall service execution time has already exceeded *Max_SRT*. Therefore, by introducing *Max_SRT* in our model, we also guarantee termination of error recovery, i.e., disallow infinite retries and rollbacks, as shown in Fig.2(e).

3 Fault Tolerance in the Presence of Parallelism

Our formal model briefly described in the previous section assumes sequential execution of subservices. However, in practice, some of subservices can be executed in parallel. Such simultaneous service execution directly affects the fault tolerance mechanisms incorporated into our B models. As a result, they become more complicated. However, at the same time it provides additional, more flexible options for error recovery that can be attempted by *Service Director*.

3.1 Modelling Execution Flow

The information about all subservices and their required execution order becomes available at the Service Decomposition phase. This knowledge can be formalised as a data structure

$$Task \ : \ seq(\mathcal{P}(SERVICE))$$

Here $SERVICE$ is a set of all possible subservices. Hence, $Task$ is defined as a sequence of subsets of subservices. It basically describes the control flow for the top service in terms of required subservices. At the same time, it also indicates which subservices can be executed in parallel.

For example,

$$Task \ = \ < \{S1, S2\}, \ \{S3, S4, S5\}, \ \{S6\} >$$

defines the top service as a task that should start by executing the services $S1$ and $S2$ (possibly in parallel), then continuing by executing the services $S3$, $S4$, and $S5$ (simultaneously, if possible), and, finally, finishing the task by executing the service $S6$.

Essentially, the sequence $Task$ defines the data dependencies between subservices. Also, $Task$ can be considered as the most liberal (from point of view of parallel execution) model of service execution. In the Service Distribution phase the knowledge about the given network architecture becomes available. This can reduce the parallelism of service control flow by making certain services that can be executed in parallel to be executed in a particular order enforced by the provided architecture.

Therefore, $Task$ is basically the desired model of service execution that will serve as the reference point for our formal development. The actual service execution flow is modelled in by the sequence $Next$ which is of the same type as $Task$:

$$Next \ : \ seq(\mathcal{P}(SERVICE))$$

Since at the Service Decomposition phase we do not know anything about future service distribution, $Next$ is modelled as an abstract function (sequence), i.e., without giving its exact definition. However, it should be compatible with $Task$. More precisely, if $Task$ requires that certain services S_i and S_j should be executed in a particular order, this order should be preserved in the sequence

$Next$. However, $Next$ can split parallel execution of given services (allowed by $Task$) by sequentially executing them in any order.

So the sequence $Next$ abstractly models the actual control flow of the top service. It is fully defined (instantiated) only in the refinement step corresponding to the Service Distribution phase. For example, the following instantiation of $Next$ would be correct with respect to $Task$ defined above:

$$Next \;\; = \; < \{S2\},\ \{S1\},\ \{S4\},\ \{S3, S5\},\ \{S6\} >$$

Also, we have to take into account that *Service Director* itself can become distributed, i.e., different parts of service execution could be orchestrated by distinct service directors residing on different network elements. In that case, for every service director, there is a separate $Next$ sequence modelling the corresponding part of the service execution flow. All these control flows should complement each other and also be compatible with $Task$.

3.2 Modelling Recovery Actions

As we described before, a *Service Director* is the service component responsible for orchestrating service execution. It monitors execution of the activated subservices and attempts different possible recovery actions when these services fail. Obviously, introducing parallel execution of subservices (described in the previous subsection) directly affects the behaviour of *Service Director*.

Now, at each execution step in the service execution flow, several subservices can be activated and run simultaneously. *Service Director* should monitor their execution and react asynchronously whenever any of these services sends its response. This response can indicate either success or a failure of the corresponding subservice.

The formal model for fault tolerance presented in Section 2.2 is still valid. However, taking into account parallel execution of services presents *Service Director* with new options for its recovery actions. For example, getting response from one of active subservices may mean that some or all of the remaining active subservices should be stopped (i.e., interrupted). Also, some of the old recovery action (like retrying of service execution) are now parameterised with a set of subservices. The parameter indicates which subservices should be affected by the corresponding recovery actions.

Below we present the current full list of actions that *Service Director* may take after it receives and analyses the response from any of active subservices. Consequently, *Service Director* might

- **Continue** to the next service execution step. In case of successful termination of all involved subservices (complete success).
- **Wait** for response from the remaining active subservices. In case of successful termination of one of few active subservices (partial success).
- **Abort** the entire service and send the corresponding message to the user or requesting component. In case of an unrecoverable error or the service timeout.

– **Cancel** (a set of subservices) by sending the corresponding requests to interrupt their execution (partial abort). In case of a failure which requires to retry or rollback in the service execution flow.
– **Retry** (a set of subservices) by sending the corresponding requests to re-execute the corresponding subservices. In case of a recoverable failure.
– **Rollback** to a certain point of the service execution flow. In case of a recoverable failure.

Service Director makes its decision using special abstract functions needed for evaluating responses from service components. These functions should be supplied (instantiated) by the system developers at a certain point of system development.

Here is a small excerpt from the B specification of *Service Director* specifying the part where it evaluates a response and decides on the next step:

```
handle =
    ...
    resp := Eval(curr_task, curr_state);
    CASE resp OF EITHER
      CONTINUE THEN
        IF curr_task = size(Next) THEN finished := TRUE
        ELSE active_serv, curr_task := Next(curr_task + 1), curr_task + 1 END
      WAIT THEN skip
      RETRY THEN active_serv := active_serv ∪ Retry(curr_task, curr_state)
      CANCEL THEN active_serv := active_serv ∪ Cancel(curr_task, curr_state)
      ROLLBACK THEN curr_task := Rollback(...); active_serv := Next(curr_task)
      ABORT THEN finished := TRUE
    END
    ...
```

where the abstract functions Next, Retry, Cancel, and Rollback are defined (typed) as follows:

$$\mathsf{Next} : \mathsf{seq}(\mathcal{P}(\mathsf{SERVICE}))$$
$$\mathsf{Eval} : 1..\mathsf{size}(\mathsf{Next}) * \mathsf{STATE} \rightarrow \{\mathsf{CONTINUE}, \mathsf{WAIT}, \mathsf{RETRY}, \mathsf{CANCEL}, \mathsf{ROLLBACK}, \mathsf{ABORT}\}$$
$$\mathsf{Retry} : 1..\mathsf{size}(\mathsf{Next}) * \mathsf{STATE} \nrightarrow \mathcal{P}(\mathsf{SERVICE})$$
$$\mathsf{Cancel} : 1..\mathsf{size}(\mathsf{Next}) * \mathsf{STATE} \nrightarrow \mathcal{P}(\mathsf{SERVICE})$$
$$\mathsf{Rollback} : 2..\mathsf{size}(\mathsf{Next}) * \mathsf{STATE} \nrightarrow 1..\mathsf{size}(\mathsf{Next}) - 1$$

4 Conclusions

In this paper we proposed a formal approach to development of communicating distributed systems. Our approach formalises and extends Lyra [6] – the UML2-based design methodology adopted in Nokia. The formalisation is done within

the B Method [1] and its extension EventB [2] – a formal framework supporting system development by stepwise refinement. The proposed approach establishes a basis for automatic translation of UML2-based development of communicating systems into the refinement process in B. Such automation would enable smooth integration of formal methods into existing development practice.

In particular, in this paper we focused on integrating fault tolerance mechanisms into the formalised Lyra development process. A big challenge is formal modelling of parallel service execution and its effect on system fault tolerance. The ideas presented in this paper are implemented by extending our previously developed B models. The formalised Lyra development is verified by completely proving the corresponding B refinement steps using the Atelier B tool. At the moment, we are in the process of moving this development to new Event B language developed within the EU RODIN project [8].

Acknowledgements

This work has been supported by IST FP6 RODIN Project.

References

1. J.-R. Abrial. *The B-Book*. Cambridge University Press, 1996.
2. J.-R. Abrial. Extending B without Changing it (for Developing Distributed Systems). *Proceedings of 1st Conference on the B Method*, pp.169-191, Springer-Verlag, November 1996, Nantes, France.
3. Clearsy. *AtelierB: User and Reference Manuals*. Available at http://www.atelierb.societe.com/index_uk.html.
4. L. Laibinis, E. Troubitsyna, S. Leppänen, J.Lilius, and Q. Malik. Formal Service-Oriented Development of Fault Tolerant Communicating Systems. *Rigorous Development of Complex Fault-Tolerant Systems, Lecture Notes in Computer Science*, Vol.4157, chapter 14, pp.261-287, Springer-Verlag, 2006.
5. L. Laibinis, E. Troubitsyna, S. Leppänen, J. Lilius, and Qaisar Malik. Formal Model-Driven Development of Communicating Systems. Proceedings of 7th International Conference on Formal Engineering Methods (ICFEM'05), LNCS 3785, Springer, November 2005.
6. S. Leppänen, M. Turunen, and I. Oliver. Application Driven Methodology for Development of Communicating Systems. *Forum on Specification and Design Languages*, Lille, France, 2004.
7. Rigorous Open Development Environment for Complex Systems (RODIN). Deliverable D7, Event B Language, online at http://rodin.cs.ncl.ac.uk/.
8. Rigorous Open Development Environment for Complex Systems (RODIN). IST FP6 STREP project, online at http://rodin.cs.ncl.ac.uk/.