Journal of Logical and Algebraic Methods in Programming 84 (2015) 611–639

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

A calculus of quality for robustness against unreliable communication

Hanne Riis Nielson, Flemming Nielson, Roberto Vigo*

Department of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads, building 324, DK-2800, Kongens Lyngby, Denmark

ARTICLE INFO

Article history:
Received 24 July 2014
Received in revised form 16 April 2015
Accepted 27 April 2015
Available online 15 May 2015

Keywords:
Availability
Denial-of-service
Process calculus
Robustness
SAT-solving

ABSTRACT

A main challenge in the development of distributed systems is to ensure that the components continue to behave in a reasonable manner even when communication becomes unreliable. We propose a process calculus, the Quality Calculus, for programming software components where it becomes natural to plan for default behaviour in case the ideal behaviour fails due to unreliable communication and thereby to increase the quality of service offered by the system. The development is facilitated by a SAT-based robustness analysis to determine whether or not the code is vulnerable to unreliable communication. The framework is illustrated on the design of a fragment of a wireless sensor network, and is substantiated by formal proofs of correctness of the analysis, which relate the original reduction semantics of the calculus to a new semantics with explicit substitutions.

1. Introduction

One of the main challenges in the development of distributed systems is to ensure that the distributed components continue to behave in a reasonable manner even when communication becomes unreliable. This is especially important for safety-critical software components in embedded systems and for software components that control part of our physical environment. With the advent of cyber-physical systems, in which software components are distributed throughout a physical system, the challenges will gain more and more relevance.

Considerable focus has been placed on how to ensure the integrity, confidentiality and authenticity of data communicated between components. In embedded systems this is easiest when communication takes place over cables shielded from other applications and used only for this purpose. However, increasingly cables are shared between many applications, including for example the infotainment system on cars, and often wireless communication needs to be employed as well, as when the control system needs to communicate with the pressure meter installed in the tyres. In healthcare applications there also is a trend to use wireless communication for interconnecting measuring apparatus with patient monitoring systems and with systems that dispense oxygen, saline, or morphine. Solutions generally include the proper use of cryptographic communication protocols that can be proved secure using state-of-the-art analysis tools.

Less focus has been placed on how to ensure that the expected communication actually takes place. This is hardly surprising given the much more challenging nature of this problem. One dimension of the problem is to ensure that other control components continue to operate and for this it often suffices to use model checking techniques for proving the

* Corresponding author. Tel.: +45 45253726.
E-mail addresses: hrni@dtu.dk (H.R. Nielson), fnie@dtu.dk (F. Nielson), rvig@dtu.dk (R. Vigo).

http://dx.doi.org/10.1016/j.jlamp.2015.04.002
2352-2208/© 2015 Elsevier Inc. All rights reserved.

absence of deadlock and livelock. Another dimension is to ensure that messages sent are in fact received and this is much harder. Over the Internet the possibility of denial-of-service attacks is well-known – simply flooding the connection with messages beyond the capacity of the recipient thereby masking the proper messages. Wireless communication is open to the same attacks as well as interference with the frequency band and physically shielding the antennas of sender and receiver as they are distributed throughout a cyber-physical system. Indeed, it might seem that this problem cannot be solved by merely using computer science techniques.

What is feasible using computer science techniques is to ensure that software systems are hardened against the unreliability of communication. This calls for programming software components of distributed systems in such a way that one has programmed a default behaviour to be enacted when the ideal behaviour is denied due to the absence of expected communication. To this end we propose

- a process calculus, the Quality Calculus, for programming software components and their interaction, natively equipped with the notion of absence of communication, and
- a SAT-based analysis to determine the vulnerability of the processes against unreliable communication.

The Quality Calculus is developed in Sections 2, 3 and clearly inherits from calculi such as CCS [27] and the π-calculus [29]. Its main novelty is a binder specifying the inputs to be performed before continuing. In the simplest case it is an input guard $t?x$ describing that some value should be received over the channel t and should be bound to the variable x. Increasing in complexity, we may have binders of the form $\&_q(t_1?x_1, \cdots, t_n?x_n)$ indicating that several inputs are *simultaneously* active and a quality predicate q that determines when sufficient inputs have been received to continue.

As a consequence, when continuing with the computation some variables might not have obtained proper values as the corresponding inputs might have not been performed. To model this we distinguish between data and optional data, much like the use of option data types in programming languages like Standard ML. The construct $\mathsf{case}\ e\ \mathsf{of}\ \mathsf{some}(y) \colon P_1\ \mathsf{else}\ P_2$ will evaluate the expression e; if it evaluates to $\mathsf{some}(c)$ we will execute P_1 with y bound to c; if it evaluates to none we will execute P_2. The expressiveness of the Quality Calculus is considered in Section 4 and an example in the context of a wireless sensor network is presented in Section 5.

The SAT-based [24] robustness analysis is developed in Section 6. It is based on the view that processes must be coded in such a way that error configurations are not reached due to unreliable communication; rather, default data should be substituted for expected data in order to provide meaningful behaviour in all circumstances. Of course, this is not a panacea – default data is not as useful as the correct data, but often better *quality* of service may be obtained when basing decisions on default or old data, rather than simply stopping in an error state. As an example, if a braking system does not get information about the spinning of the wheels from the ABS system, it should not simply stop braking, rather it should continue to brake – perhaps at reduced effect to avoid blocking the wheels.

The analysis attaches propositional formulae to all points of interest in the processes; such formulae characterise the combinations of optional data that could be missing. This is useful for showing that certain error configurations cannot be reached; indeed, if a propositional formula is unsatisfiable then the corresponding program point cannot be reached. The availability of extremely efficient SAT-solvers makes this a very precise analysis method with excellent scalability.

As a matter of fact, denial-of-service attacks are successful in the real world, as witnessed increasingly even by the general public. Therefore, there is a need for principled approaches to cope with the consequences of unavailability. In particular, dealing with the consequences of denial-of-service leads to a framework where different sources – such as nature, misfortune, and adversarial behaviour including physical destruction – are treated in a uniform manner. A main such consequence is the absence of expected communication; the Quality Calculus allows to model seamlessly such concern, and the robustness analysis enforces availability considerations taking advantage of the expressiveness of the calculus primitives.

Our investigation leads to a world in which components are aware of being part of a system and are determined to operate even if their ideal partners become unavailable and do not provide expected information, perhaps because such partners are undergoing a denial-of-service attack. The overall perspective is then lifted from a self-centred, muscular approach to a view which admits the existence of denial-of-service and tries to circumvent it by means of default data. Therefore, dealing with the consequences of *successful denial-of-service* at component level, the Quality Calculus defies unavailability at system level to the extent possible. As a matter of fact, full resilience against denial-of-service can only be obtained by fighting the sources of unavailability from the target side as well as being ready to cope with its consequences at the system level. In this sense, our work is to be understood as complementary to the study of sources of unavailability, adversarial behaviours, and reactive countermeasures. In other words, the calculus advocates the design of systems where the absence of communication is remedied by means of default data, such as estimates and historical data. In the (limited) cases where only actual data can be used, reactive countermeasures have to be adopted to prevent such key components from denying service to their partners.

Section 8 positions our approach with respect to the literature on process calculi and denial-of-service. Finally, we conclude and present our outlook on future work in Section 9. Appendices A and B contain detailed proofs.

The robustness analysis is to be understood as an enforcement mechanism guiding the programmers, as envisioned by Dijkstra [13]: