Advances in Quantitative Verification for Ubiquitous Computing

Marta Kwiatkowska

Department of Computer Science, University of Oxford, Oxford, OX1 3QD

Abstract. Ubiquitous computing, where computers 'disappear' and instead sensor-enabled and software-controlled devices assist us in everyday tasks, has become an established trend. To ensure the safety and reliability of software embedded in these devices, rigorous model-based design methodologies are called for. Quantitative verification, a powerful technique for analysing system models against quantitative properties such as "the probability of a data packet being delivered within 1ms to a nearby Bluetooth device is at least 0.98", has proved useful by detecting and correcting flaws in a number of ubiquitous computing applications. In this paper, we focus on three key aspects of ubiquitous computing: autonomous behaviour, constrained resources and adaptiveness. We summarise recent advances of quantitative verification in relation to these aspects, illustrating each with a case study analysed using the probabilistic model checker PRISM. The paper concludes with an outline of future challenges that remain in this area.

1 Introduction

Ubiquitous computing, also known as pervasive computing, was envisaged by Mark Weiser in [47], where he predicted that computers will "weave themselves into the fabric of everyday life until they are indistinguishable from it". Today, powered by advances in microelectronics, mobile phone technology and cloud computing, we are witnessing a tremendous growth in device technologies for software-controlled 'smart' devices that support our daily activities and autonomously make decisions on our behalf. They can sense what is around them, remember the context and adapt to new situations. They can communicate wirelessly with other devices and humans, and are Internet-enabled. Applications are endless, from environmental and health monitoring, through home appliance networks, to self-driving cars. A related vision is the 'Internet of Things', where everyday objects (called 'everyware' by Adam Greenfield) are enhanced with information processing capabilities.

The growing dependence of society on ubiquitous computing calls for rigorous device design methodologies, which is particularly important for their embedded software that controls the device actions and whose failure can lead to costly recalls. Model-based design methodologies have the potential to improve the reliability of devices and reduce the development effort through code generation and software reuse via product lines. In particular, automated verification

via model checking provides the means to systematically verify software against correctness requirements such as "the smartphone will never reveal private data to unauthorised contacts". These techniques have been successfully applied to TinyOS sensor network software [5] with respect to safety assertions. However, when modelling ubiquitous computing devices we often need to include quantitative aspects such as probability, time delays and resource usage in the models. Probability is needed because of inherent unreliability of wireless communication technologies such as Bluetooth and ZigBee, which use randomised back off schemes to minimise collisions; also, embedded devices are frequently powered by battery and components may be prone to failure. *Quantitative verification* [31] techniques are well suited to this case, where systems are modelled as variants of Markov chains, annotated with real-time and quantitative costs/rewards. The aim is to automatically establish quantitative properties, such as "the probability of a monitoring device failing to issue alarm when a dangerous rise in pollutant level is detected", "the worst-case expected time for a Bluetooth device to discover another device in vicinity", or "the minimum expected power consumption of the smartphone while looking up directions with GPS". Quantitative, probabilistic verification has been implemented in the probabilistic model checker PRISM [34], which has been applied to a wide range of case studies from the ubiquitous computing domain, resulting in automatic detection and diagnosis of software flaws and unexpected trends.

In this paper, we provide a brief overview of quantitative verification techniques, including typical features of the models and property specification notations. We then describe a selection of recent advances, highlighting the following three key aspects of ubiquitous computing devices:

1. *autonomous behaviour*: increasingly, we are relying on ubiquitous computing devices to act autonomously on our behalf, including *safety-critical* applications such as self-driving cars, or robotic search and rescue missions;
2. *constrained resources*: the embedded processors have limited memory and CPU speed, are often battery powered, and employ unreliable communication technologies, and yet they are expected to *reliably and timely* perform critical functions such as financial transactions;
3. *adaptiveness*: ubiquitous computing devices are typically enabled and managed by cloud services, which dynamically adapt behaviours to changing requirements and environmental context, necessitating continuous monitoring and runtime verification to provide *dependability* assurance.

We illustrate each of the above with a typical case study, drawn from the ubiquitous computing domain, that used PRISM, describing the modelling approach taken and lessons learnt. The case studies involve sensor-enabled mobile devices such as autonomous robots, smartphones and healthcare monitoring. The requirements that we wish to ensure include:"the robot will successfully arrive at the exit with probability greater than 0.99, without hitting any obstacles" (autonomous behaviour); "the email protocol will ensure that the total energy cost of sending a message does not exceed a specified bound, even if the bit error rate is high (constrained resources); and "the device will maintain 0.97 minimum

probability of delivering sensor readings to the beacon within 5ms, even if the bandwidth drops from time to time" (adaptiveness). We conclude by outlining future research challenges in quantitative verification for ubiquitous computing.

The paper is organised as follows. In Section 2 we give an overview of quantitative verification techniques, focusing on Markov chain and Markov decision process models, and the corresponding temporal logics, PCTL and CSL. Section 3 demonstrates recent advances of quantitative verification by highlighting a number of case studies from ubiquitous computing, all analysed with the PRISM model checker [34]. Section 4 concludes the paper by summarising future research challenges.

2 Quantitative Verification Basics

We give a brief overview of a selection of probabilistic models and specification formalisms used in automated quantitative verification. The models have been chosen according to case studies presented in the next section. We note that all models and specification notations discussed are supported by PRISM [34].

2.1 Markov Decision Processes

In ubiquitous computing devices probabilistic behaviour typically coexists with nondeterminism. Probability is the result of a random event, for example an electronic coin toss, sensor failure or stochastic delay, and nondeterminism is used to model concurrent execution or action-based control. Both nondeterminism and (discrete) probability are present in the classical model of *Markov decision processes (MDPs)*.

Let S be a finite set; we denote the set of probability distributions over S by $\mathit{Dist}(S)$.

Definition 1 (MDP). *A* Markov decision process (MDP) *is a tuple* $\mathcal{M} = (S, \bar{s}, \mathit{Act}, \mathbf{P}, \mathit{AP}, L)$*, where*

- S *is a finite set of states and* $\bar{s} \in S$ *is the initial state;*
- Act *is a finite set of actions;*
- $\mathbf{P} : S \times \mathit{Act} \times S \in [0,1]$ *is a transition probability matrix where* $\sum_{s' \in S} \mathbf{P}(s,a,s') \in \{0,1\}$ *for any* $s \in S$ *and* $a \in \mathit{Act}$*;*
- AP *is a set of atomic propositions;*
- $L : S \to 2^{\mathit{AP}}$ *is a labelling of states with atomic propositions.*

We let $\delta(s) \subseteq \mathit{Act}$ denote the set of actions enabled in s, i.e. $a \in \delta(s)$ if $\sum_{s' \in S} \mathbf{P}(s,a,s') = 1$. The MDP executes as follows: in each state s the successor state is chosen by, first, nondeterministically selecting an enabled action $a \in \delta(s)$, and, second, choosing the successor according to the probability distribution $\mathbf{P}(s,a)$. A *path* of $\mathcal{M}$ is of the form $\pi = s_0 a_0 s_1 a_1 s_2 \cdots$ where $a_i \in \delta(s_i)$ and $\mathbf{P}(s_i, a_i, s_{i+1}) > 0$ for each $i \geq 0$.

To reason formally about the behaviour of MDPs, we use the notion of *strategies* (also called *adversaries* or *policies*), which resolve all the nondeterministic choices in a model. Formally, a strategy is a function σ that maps a finite path ending in s to an action in $\delta(s)$ based on the history of choices made so far. Under a particular strategy, the behaviour of an MDP is fully probabilistic and we can define a probability space over the possible paths through the model using standard construction [30].

A *discrete-time Markov chain (DTMC)* is an MDP $\mathcal{M} = (S, \bar{s}, Act, \mathbf{P}, AP, L)$ with $|Act| = 1$, where $\sum_{s' \in S} \mathbf{P}(s, a, s') = 1$ for all $s \in S$. Thus, a DTMC can be viewed as a single transition probability matrix $\mathbf{P} : S \times S \in [0, 1]$, with all rows summing up to 1. We omit Act from the tuple.

MDP properties are typically expressed in temporal logics. The logic PCTL (Probabilistic Computation Tree Logic) [29], a probabilistic extension of the temporal logic CTL is defined below.

Definition 2. *The syntax of PCTL is given by:*

$$\Phi ::= \mathit{true} \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \mathrm{P}_{\sim p}[\,\psi\,]$$
$$\psi ::= \mathrm{X}\,\Phi \mid \Phi\,\mathrm{U}^{\leq k}\,\Phi \mid \Phi\,\mathrm{U}\,\Phi$$

where a is an atomic proposition, $\sim \in \{<, \leq, \geq, >\}$, $p \in [0, 1]$ and $k \in \mathbb{N}$.

PCTL formulae Φ are interpreted over the states of an MDP. As path formulae we allow $\mathrm{X}\,\Phi$ ("Φ is satisfied in the *next* step") $\Phi_1\,\mathrm{U}^{\leq k}\,\Phi_2$ ("Φ_2 is satisfied within k steps and Φ_1 is true *until* that point") and $\Phi_1\,\mathrm{U}\,\Phi_2$ ("Φ_2 is eventually satisfied and Φ_1 is true *until* then"). The usual derived variants $\mathrm{F}\,\Phi$, $\mathrm{G}\,\Phi$ are also permitted. We say that a state $s \in S$ *satisfies* a PCTL formula Φ, denoted $s \models \Phi$, if it is true for s, which means that the probability of a path formula ψ being true in a state satisfies the bound $\sim p$ for all strategies. We can also use PCTL in *quantitative* form, e.g. $\mathrm{P}_{min=?}[\ \psi\ ]$, which returns the minimum/maximum probability of satisfying ψ. Examples of PCTL properties are:

- $\mathrm{P}_{\max=?}\,[\,\mathrm{F}\,\mathit{lost}\,]$ - "the maximum probability, over all possible strategies, of the protocol losing a message";
- $\mathrm{P}_{\min=?}\,[\,\mathrm{F}^{\leq 10}\,\mathit{deliver}\,]$ - "the minimum probability, over all possible strategies, of the protocol delivering a message within 10 time steps";
- $\mathrm{P}_{\geq 1}\,[\,\mathit{near\_supplies}\,\mathrm{U}\,\mathit{exit}\,]$ - "under all possible strategies, with probability 1, the robot always remains near supplies until exiting".

Model checking PTCL over MDPs requires a combination of graph-based algorithms and numerical solution techniques. Typically, we are interested in the best- or worst-case behaviour and compute the minimum or maximum probability that some event occurs, quantifying over all possible strategies. The minimum and maximum probabilities can be computed by solving linear optimisation problems, which is often implemented using dynamic programming.We can also *synthesise* the strategy that achieves the minimum/maximum probability. For the case of DTMCs, it suffices to solve linear equation systems. More expressive

logics such as LTL or PCTL* can also be defined, albeit their model checking becomes more expensive. The usual approach is to convert the LTL formula to a deterministic Rabin automaton and perform verification on the *product* of this automaton and the original MDP; see e.g. [2,22].

2.2 Continuous-Time Markov Chains

In MDPs, the progress of time is modelled by discrete time steps, one for each transition of the model. For many applications, it is preferable to use a *continuous* model of time, where the delays between transitions can be arbitrary real values. A natural extension of MDPs with real-time (not discussed here) is probabilistic timed automata [37]. We focus on the simpler, classical model of *continuous-time Markov chains (CTMCs)*, which have no nondeterminism, and extend DTMCs with real-time by modelling transition delays with exponential distributions.

Definition 3. *A continuous-time Markov chain (CTMC) is* $\mathcal{C} = (S, \bar{s}, \mathbf{P}, E, AP, L)$ *where:*

- $(S, \bar{s}, \mathbf{P}, AP, L)$ *is a DTMC;*
- $E : S \to \mathbb{R}_{\geq 0}$ *is the* exit rate.

In a CTMC $\mathcal{C}$, the residence time of a state $s \in S$ is a random variable governed by an exponential distribution with parameter $E(s)$. Therefore, the probability to exit state s in t time units is given by $\int_0^t E(s) \cdot e^{-E(s)\tau} d\tau$. To take the transition from s to another state s' in t time units, the probability equals $\mathbf{P}(s, s') \cdot \int_0^t E(s) \cdot e^{-E(s)\tau} d\tau$.

Intuitively, the CTMC executes as follows: in each state s, it stays in this state for time t, drawn from exponential distribution with parameter $E(s)$, and then moves to state s' with probability $\mathbf{P}(s, s')$. A *timed path* of $\mathcal{C}$ is a finite or infinite sequence $s_0 t_0 s_1 t_1 s_2 \cdots t_{n-1} s_n \ldots$, where $t_i \in \mathbb{R}_{>0}$ for each $i \geq 0$. As for DTMCs, a probability space over the paths through a CTMC can be defined [3], where events correspond to certain sets of paths.

To specify quantitative properties of CTMCs, the logic CSL [3] has been proposed, which is syntactically similar to PCTL, except that it now includes continuous versions of the step-bounded path operators, that is, $\Phi_1 \, \mathrm{U}^{[t,t']} \, \Phi_2$, where $t, t' \in \mathbb{R}_{\geq 0}$, is true for a path if Φ_1 is satisfied until Φ_2 becomes true at a time point belonging to the interval $[t, t']$. Then, for example, $\mathrm{P}_{=?} \, [\, \mathrm{F}^{[0,t]} \, \Phi \,]$ denotes the transient probability of satisfying Φ by time t. A steady state operator S is also added, which can express the long-run probability of residing in a state satisfying some state formula. Examples of CSL properties are:

- $\mathrm{P}_{=?} \, [\, \mathrm{F}^{[5,5]} \, \neg \mathit{empty} \,]$ - "the probability of the robot's battery not being depleted at time 5 mins";
- $\mathrm{P}_{=?} \, [\, \mathit{near\_supplies} \, \mathrm{U}^{[0,5.5]} \, \mathit{exit} \,]$ - "the probability of the robot remaining near supplies before exiting safely within 5.5 mins";
- $\mathrm{S}_{\geq 0.99} \, [\, \neg \mathit{fail} \,]$ - "the long-run probability of the robot being operational is at least 0.99".

Model checking for CSL over CTMCs proceeds by discretisation into a DTMC. The steady-state operator is computed by solving a linear equation system, whereas the probabilistic operator reduces to transient probability calculation, and is typically implemented using *uniformisation*, an efficient iterative numerical method. For more information see e.g. [32]. More expressive, durational properties for CTMCs can also be defined and automatically verified [14].

2.3 Adding Costs & Rewards

The above probabilistic models can be augmented with *reward* information (also referred to as *cost*), which enables the computation of expected reward values. For simplicity, we only show the extension for DTMCs. For $\mathcal{D} = (S, \bar{s}, \mathbf{P}, AP, L)$, we define a *reward structure* $(\underline{\rho}, \boldsymbol{\iota})$, where: $\underline{\rho} : S \to \mathbb{R}_{\geq 0}$ assigns rewards to states, and $\boldsymbol{\iota} : S \times S \to \mathbb{R}_{\geq 0}$ assigns rewards to transitions. The state reward vector $\underline{\rho}(s)$ is the reward acquired in state s per time step, and the transition reward $\boldsymbol{\iota}(s, s')$ is acquired each time a transition between states s and s' occurs.

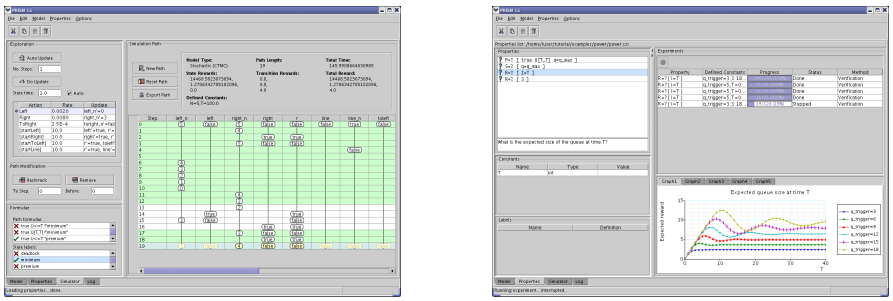
Reward structures can be used to represent a variety of different aspects of a system model, for example "number of packets dropped by the protocol" or "the expected energy consumed in the start-up phase". To express reward-based properties for DTMCs, the logic PCTL can be extended [32] with additional operators:

$$\mathrm{R}_{\sim r}[\,\mathrm{C}^{\leq k}\,] \;\big|\; \mathrm{R}_{\sim r}[\,\mathrm{I}^{=k}\,] \;\big|\; \mathrm{R}_{\sim r}[\,\mathrm{F}\,\Phi\,]$$

where $\sim \in \{<, \leq, \geq, >\}$, $r \in \mathbb{R}_{\geq 0}$, $k \in \mathbb{N}$ and Φ is a PCTL formula. The formula $\mathrm{R}_{\sim r}[\,\psi\,]$ is satisfied in a state s if, from s, the *expected* value of reward ψ meets the bound $\sim r$. The formula ψ can take the form: $\mathrm{C}^{\leq k}$, which refers to the reward *cumulated* over k time steps; $\mathrm{I}^{=k}$, the state reward at time *instant* k (i.e. after exactly k time steps); and $\mathrm{F}\,\Phi$, the reward cumulated before a state satisfying Φ is *reached*. Similarly to the P operator, we also use the *quantitative* form $\mathrm{R}_{=?}[\,\psi\,]$, meaning the value of the expected reward. The following are examples of reward properties assuming appropriate reward structures have been defined:

- $\mathrm{R}_{=?}[\mathrm{C}^{\leq 10}]$ - "the expected power consumption within the first 10 time steps of operation";
- $\mathrm{R}_{=?}[\mathrm{I}^{=100}]$ - "the expected number of regions visited by the robot after 100 time steps have passed";
- $\mathrm{R}_{\geq 5}[\mathrm{F}\ \mathit{exit}]$ - "the expected number of regions visited by the robot until exiting is at least 5".

Model checking for the reward operators for DTMCs reduces to a combination of graph algorithms and solution of linear equations; see e.g. [32] for more information. An extension of CSL with the reward operator was formulated in [32]. Similarly, cumulative and instantaneous reward operators can be added to PCTL for MDPs [22], where minimum/maximum expected rewards, denoted by $\mathrm{R}_{max=?}[\cdot]$ in quantitative form, are computed over all strategies. Steady-state rewards, respectively long-run average in the case of MDPs, can also be defined [32,22].



(a) Model and strategy exploration (b) Experiments and graph plotting

Fig. 1. Screenshots of the PRISM graphical user interface

2.4 Quantitative Verification with PRISM

Quantitative verification techniques been implemented within PRISM [34,1], a probabilistic model checker developed at the Universities of Birmingham and Oxford. PRISM provides direct support for DTMCs, MDPs and CTMCs, as well as two additional models not discussed here, probabilistic timed automata (PTAs) and stochastic multi-player games (SMGs), the latter via the tool PRISM-games [11]. The models are specified using a high-level modelling language based on guarded command notation. Quantitative properties can be specified in the temporal logics PCTL, LTL, PCTL* and CSL, which include both probabilistic and reward operators.

PRISM is primarily a *symbolic* model checker (based on variants of Binary Decision Diagrams (BDDs)), but it also makes extensive use of *explicit* storage schemes such as sparse matrices and arrays, and implements multiple engines for efficiency and scalability. The verification algorithms can provide either *exact*, numerical solutions to the induced linear equation systems or linear programming problems, typically computed iteratively, or *approximate* the probability/expectation by sampling executions using Monte Carlo techniques and performing statistical inference to estimate the probability of satisfying the property (also known as *statistical model checking*). Parametric models are now supported [15]. It is also possible to *simulate* the model and *synthesise* the strategy that minimises/maximises the probability or reward [1].

PRISM's graphical user interface, shown in Figure 1, provides a model editor, a simulator for model debugging and strategy exploration, and graph-plotting functionality. Models and properties can be specified using parameters, and *experiments* facilitate the search for flaws or unusual trends, by plotting the values of quantitative queries as the parameters in the model or property are varied.

PRISM is free and open-source (released under the GPL license), runs on most major operating systems, and has been applied to several application domains, including distributed algorithms, security protocols, dependability, planning, workflows, and biology; see [1] for more information.

3 Quantitative Verification for Ubiquitous Computing

Quantitative verification is a powerful and widely applicable method, which has been successfully applied in the context of ubiquitous computing. We mention, for example, the modelling and analysis of performance and energy consumption of the Bluetooth device discovery protocol [16], which established for the first time that the worst-case time to hear one message is unusually long, about 2.5s. A similar analysis has been performed for the ZigBee sensor network protocol [26], recently extended with analysis of the impact of key update strategies on network performance. Further studies concerning mobile devices include performability of several dynamic power management schemes [43].

The ultimate challenge is to apply these techniques to real-world ubiquitous computing scenarios, which are expanding rapidly. In this paper, we survey some of the recent advances by focusing on three key aspects of ubiquitous computing devices: *autonomous behaviour*, *constrained resources* and *adaptivity*. Each aspect will be illustrated by means of case studies that use PRISM.

3.1 Autonomous Behaviour

A growing number of ubiquitous computing applications involves designing autonomous robotic missions, such as those used in planetary exploration, for example Mars Rover, or disaster search and rescue. A key challenge for the designers is to construct a mission so that it satisfies some high-level goals, and executes in a timely manner and without fault. In addition, technological progress towards self-parking and self-driving cars calls for software tools to support the design of safe autonomous vehicle control.

In this section, we highlight the research aimed at automated generation of control strategies for robotic vehicles in dynamic environments as reported in [41]. The approach is based on the observation that temporal logic such as CTL can be used to specify the mission goals, for example, "the robot will remain in safe regions until exiting successfully". Under the assumption that the environment is static and can be partitioned into a finite-state transition system, conventional model checking tools can be applied to analyse and generate strategies in this case. However, one has to consider *noise* on sensors induced from measurement errors; similarly, control actions can exhibit *uncertainty* due to actuator error, in the sense that the robot can move to an adjacent region or remain in the current region, and this choice is probabilistic, with probabilities that can be estimated. Natural models for such scenarios are therefore Markov decision processes (as overviewed in Section 2.1), which are partitioned into finitely many regions. If RFID tags are placed in regions, the robot can uniquely determine the region it is in; we can thus assume full observability, thus avoiding the need to consider partially-observable MDPs. Consequently, we can now employ temporal PCTL to specify mission goals, and the above goal now becomes: "what is the probability that the robot will remain in safe regions until exiting successfully?".

The problem can be stated as follows. Consider a robot moving in a partitioned environment with static obstacles modelled as a Markov decision process $\mathcal{M}$. Given a PCTL formula Φ that specifies the mission goal, determine a control strategy that *optimises* the probability of satisfying Φ. Clearly, this problem can be solved by applying quantitative verification as described in Section 2.1, namely, computing the minimum/maximum probability or expectation, and then *synthesising* the optimal strategy.

In previous work, the authors considered *safe vehicle control* in city environments, and developed a tool for synthesising control strategies from PCTL based on PRISM. Later, they extended the synthesis algorithms to include expected reward specifications to incorporate aspects such as time and energy cost, as well as Boolean combinations of PCTL formulae [41]. They also validated their approach using an experimental testbed that employs iRobot Create[1], the popular programmable robot. The following are examples of actual mission goal specifications from [41]:

- $\mathrm{P}_{max=?}[\,(\mathbf{S} \vee (\mathbf{R} \wedge \mathbf{M_1}))\,\mathrm{U}\,\mathbf{D_1}\,]$ - "reach *Destination 1* by driving through either only *Safe* regions or through *Relatively Safe* regions only if *Medical Supply 1* is available at such regions";
- $\mathrm{P}_{max=?}[(\,\mathrm{P}_{\leq 0.5}[\mathrm{X}\mathbf{R}] \wedge \neg\mathbf{U})\mathrm{U}\mathbf{D_1}\,]$ - "reach *Destination 1* by going through the regions from which the probability of converging to a *Relatively safe* region is less than 0.5 and always avoiding *Unsafe* regions".

The results reported are encouraging, with the tool able to synthesise control strategies for MDPs with 1000 states, though the construction of the MDP using Monte Carlo simulation can be expensive.

Strategy synthesis from LTL specifications has been formulated, e.g., in [48]. Probabilistic verification of coordinated foraging-and-reacting multi-robotic missions is considered in [8], where compositionality is employed to improve scalability. We remark that more complex mission goals may require *multi-objective* specifications, where the simultaneous satisfaction of more than one property, e.g. maximise the probability of reaching target and minimise travelling time, is needed. Automated verification and synthesis for multi-objective specifications for MDPs have been developed [24] and applied to synthesis of strategise for the team formation protocol. In recent work, we have developed synthesis algorithms for autonomous driving for conjunctive multi-objective specifications for stochastic game models, based on actual map data [12].

3.2 Constrained Resources

Ubiquitous computing devices are frequently wearable, for example, low-cost RFID tags and wireless body sensors, and consequently have limited memory and computational capacity. At the same time, they may need to execute computationally intensive tasks, such as authentication and communication in unreliable wireless media. Typically battery-powered, where it is either difficult or

[1] `http://verifiablerobotics.com/CreateMATLABsimulator/createsimulator.html`

inconvenient to access energy renewal sources, the need for high-quality resource management protocols for such devices is paramount. Quantitative verification has previously been applied to analyse dynamic power management schemes for mobile devices [43] and energy harvesters [46]. Here, we highlight a recent case study that applied quantitative verification with PRISM to analyse the computational and transmission cost of an electronic email protocol [4], with the view to provide tool-assisted systematic analysis of the protocol for a variety of mobile environments.

The Certified E-mail Message Delivery (CEMD) protocol studied in [4] is used on mobile devices such as smartphones and PDAs, including low-cost hardware, frequently operating in noisy environments with high bit error rate. The CEMD protocol provides a number of security features, such as fairness (neither sender nor recipient should gain advantage upon interruption), timeliness (all participants should be able to exit the protocol in a given finite time), and confidentiality (only the intended participant should learn the contents of an email message, implemented using RSA encryption). In view of the computational cost of RSA operations, it is important for the designers to understand the impact of executing the email service on CPU performance and energy consumption. To this end, a CTMC model (see Section 2.2) of the protocol is developed in PRISM, and parameterised based on the popular Texas Instruments TMS320C55x processor which performs at the low frequency end of 200MHz, hence maintaning the ability to provide services in low power modes. Then a detailed analysis is performed using the derived parameters, also taking into account the number of parallel sessions and realistic bit error rates of typical mobile communication technologies, both of which affect the performance.

The quantitative analysis is focused on *computational* and *transmission* costs of the CEMD protocol, respectively defined as a function of the CPU cycles needed to perform RSA operations, and a function of the negative acknowledgement rate and bit error rate of the wireless medium. The properties are expressed as CSL formulae, and analysed for different values of the parameters, assuming suitable reward structures:

- $\mathrm{R}_{=?}[\mathrm{C}^{\leq T}]$ - "the expected computational cost of completing all protocol's sessions in finite time T";
- $\mathrm{P}_{=?}[\mathrm{F}^{[0,T]}\, \mathit{finish}]$ - "the probability of completing all protocol's sessions in finite time T".

The analysis has revealed that the CEMD protocol is highly dependent on the specific characteristics of the environment that it runs on, with widely different behaviour, and may lead to instabilities. The derived model of the protocol provides a sound foundation for a methodology to analyse further quantitative aspects of the protocol, for example energy consumption.

Resource efficiency, particularly in relation to energy, is a major design issue for ubiquitous computing devices. Quantitative analysis of low-cost RFID authentication protocol was performed in [45]. The effect of mobility and transmission range on energy consumption was considered for mobile process calculi frameworks in [27]. The quantitative analysis of a smartgrid protocol using

PRISM-games [10,11], a PRISM extension for stochastic games, revealed a flaw, which was fixed by introducing incentives to prevent selfishness.

3.3 Adaptiveness

Many ubiquitous computing applications continuously monitor the environment by means of sensors and must adapt to new scenarios. Ubiquitous computing systems such as home networks are enabled by service-based systems, typically based on cloud computing [17], which dynamically adapt behaviours to the changing requirements and contexts. It has been argued [6] that the need to continuously provide reliability, dependability and performance guarantees for adaptive systems calls for *quantitative runtime verification*. This is different from offline quantitative verification performed at the design stage, as described in Section 2, where a model is developed and analysed pre-deployment in order to improve the design. Runtime verification, in contrast, is invoked as the system is being executed, intercepting and steering its execution to ensure that given requirements are continuously satisfied in spite of adaptation.

In [7], we have developed an extensive framework called QoSMOS which can be used to dynamically manage and optimise the performance of service-based systems. The framework has been demonstrated a typical ubiquitous computing healthcare scenario, called TeleAssistance, where patients are remotely monitored, with data being sent to a medical lab for analysis, and there is a requirement to guarantee a certain QoS level of delivering a specific service, for example to change the dosage of a drug. The system is built as a workflow of web services, and may suffer from component failures. The framework proceeds autonomically, repeatedly invoking the monitoring, analysis, planning and execution stages (so called MAPE loop) as follows:

- *monitor* the reliability, workload and response time of services, to derive an operational model;
- *analyse* performance and QoS requirements, utilising the values of parameters obtained from the monitoring phase;
- *plan* adaptation of the system based on the results of analysis, which may involve changing the resource allocation or selection of optimal service;
- *execute* the adaptation of the system.

The models used for the TeleAssistance application are DTMCs and CTMCs, and the following are example requirements:

- $\mathrm{P}_{\leq 0.13}[\,\mathrm{F}\,\mathit{failedAlarm}\,]$ - "the probability that at an alarm failure ever occurs during the lifetime of the system is less than 0.13" (PCTL property);
- $\mathrm{R}_{\leq 0.05}[\,\mathrm{F}^{[0,86400]}\,\mathit{dropped}\,]$ - "the probability of a *changeDrug* request being dropped due to the request queue being full during a day of operation is less than 0.05" (CSL property).

The QoSMOS framework implements the analysis stage using quantitative verification with PRISM. This involves executing PRISM verification tasks at runtime, which works well when the number of services is small, but may become

impractical when the size of the model or the number of tasks increases. To improve efficiency, one can employ *parametric* approaches, e.g. [21], and specifically *parameter synthesis* [28]. We consider parametric probabilistic models, where probabilities are specified in terms of expressions over parameters, rather than concrete values. Then, the parameter synthesis problem aims to determine the valuations for parameters which guarantee the satisfaction of a given property, and can be solved by means of constraint solving. In recent work, we have applied sampling-based and swarm intelligence techniques to heuristically search for a good valuation of parameters for parametric models [15] (parametric MDPs and DTMCs/CTMCs). We note that this approach only finds one such valuation, rather than all the valuations for parameters. However, it may results in performance improvement by orders of magnitude, and is therefore particularly well suited to runtime verification scenarios.

The parameter synthesis methods, both those based on constraint solving as well as heuristic search, have recently been implemented within PRISM [15]. Alternative approaches to improve efficiency of quantitative runtime verification include incremental model construction and incremental verification [25], which avoid the need to rerun a verification task by reusing results from previous verifications.

3.4 Further Advances

We briefly summarise further developments in quantitative verification that have shown promise.

Compositional probabilistic verification. The size and complexity of ubiquitous computing communities demand improvements in the capacity of quantitative verification tools. Compositional assume-guarantee techniques have the potential to improve the scalability of model checking by subdividing the verification into separate tasks for each component of the system being analysed. For example, to verify property G on a two-component system $M_1 \| M_2$ we (i) check that, under the assumption that some property A holds, M_2 is guaranteed to satisfy G; and (ii) check that M_1 always satisfies the assumption A under any context. In recent work [35,23], *compositional assume-guarantee* techniques have been developed for MDPs, for both quantitative safety and liveness properties. Several proof rules have been developed to support compositional probabilistic model checking of MDPs, and implementation in terms of multi-objective probabilistic model checking [18] has been provided as an extension of PRISM, which yields substantial improvement over monolithic methods. The process can be fully automated for safety properties by applying automata learning techniques [19,20] to generate assumptions.

Cooperative and competitive behaviour. Ubiquitous computing involves communities of self-interested agents, who may need to cooperate to achieve certain goals. Traditionally, cooperative behaviour has been analysed using game theory. Building upon this, we develop quantitative verification methods for *multi-player stochastic games*, which model communities of agents that can exhibit probabilistic behaviour, for examples as a result of random choices. Prop-

erty specifications are stated in terms of a probabilistic and reward extension of the well-known logic ATL, called rPATL [10], which can express properties such as: "does the coalition have a strategy to meet a quantitative goal, irrespective of the strategies of the other players?". The framework has been applied to the analysis of a smartgrid protocol, collective decision making for sensor networks and user-centric networks, where we discovered and corrected flaws in existing protocols [10,39]. The techniques have been implemented as an extension of PRISM [11], and include both verification, as well as strategy synthesis.

Probabilistic real-time protocols. Many ubiquitous computing applications require consideration of probability and continuous real-time, in conjunction with nondeterminism that is used to model distributed computation. The model of *probabilistic timed automata* (PTAs) [37] can be viewed as a Markov decision process extended with real-valued clocks or, alternatively, an extension of the well-known timed automata formalism with discrete probabilistic choice. PTAs naturally model distributed randomised algorithms with timing, for example the ZeroConf protocol, as well as the medium access protocols for wireless networks, including WiFi, Bluetooth and ZigBee; all have been analysed with PRISM [1]. A number of techniques have been developed for quantitative verification of PTAs, including the *digital clocks* [36] approach; forwards [37] and backwards reachability [38] based on *zones*; and *game-based quantitative abstraction-refinement* [33]. Strategy synthesis is also possible [44]. PRISM provides native support for PTAs, via the techniques of [33] and [36].

Medical devices. Embedded software is increasingly often used in medical devices, which monitor and control physical processes such as electrical signal in the heart or dosage of insulin. For example, an implantable cardiac pacemaker device reads electrical signals from sensors placed on the heart muscle, monitors the timing between heart beats, and, if any are missed, generates signals to stimulate the heart, maintaining the rate of 60-100 beats per minute. Quantitative verification can provide automated means to verify safety properties of the pacemaker, but the models must incorporate *continuous dynamics*, necessary to model the electrical signal in the heart, in addition to timing and probabilities. A further difficulty is the need to verify the properties against realistic heart models. Recently, we developed two physiologically relevant heart models, one based on ECG signals [9] and the other on a network of cardiac cells [13]. We have composed the heart models with timed automata models of pacemaker software, and subjected the composed system to quantitative verification. We are able to verify basic safety properties, for example whether the pacemaker corrects the slow beat of the heart, as well as more complex properties, such as providing a detailed analysis of energy consumption.

A specific challenge of medical devices is that they need to interface to biological systems. Quantitative modelling and verification technology has already been applied to DNA computing devices [42], where it was able to automatically discover and diagnose a design error. These methods are applicable to molecular sensing devices at the nanoscale.

4 Challenges and Future Directions

Ubiquitous computing was conceived over 20 years ago by Marc Weiser [47] and has been unstoppable since. Smartphones have far outstripped the sales of desktop PCs. Enhanced with a multitude of sensors, smartphones and tablets are being used for a variety of tasks, from email, through looking up restaurants nearby, to monitoring of the heart rate and air pollution.

The emergence of ubiquitous computing has posed new challenges for software and device designers. Ubiquitous computing was recognised in the UK as a Grand Challenge [40], subdivided into: the engineering of ubiquitous computing devices, their scientific understanding, and human interaction mechanisms. The research on quantitative verification reported in this paper contributes to the scientific understanding of ubiquitous computing led by Robin Milner, and is related to the Verified Software initiative of Tony Hoare. Quantitative verification research is very much inspired by their vision. It naturally complements the core activities of the two initiatives, focusing on practical, algorithmic solutions, that have the potential to drive the development of industrially-relevant methodologies and software tools to support the design of ubiquitous computing devices.

Much progress has been made in quantitative verification for ubiquitous computing, as reported here and elsewhere, and supported by effective software tools. Successes include synthesising safe strategies for autonomous vehicles; analysing quantitative trends of low-level network protocols; finding and correcting flaws in smartgrid energy distribution protocols; development of methodologies for the verification of medical devices; and adaptive service-based frameworks which can continuously guarantee the satisfaction of given QoS properties. Key limitations of current techniques are poor scalability of quantitative verification; lack of effective methods for integrating discrete, continuous and stochastic dynamics; and poor efficiency of quantitative runtime verification. The scale and complexity of the ubiquitous computing scenarios are so great that the challenges that remain seem prohibitive. We anticipate that following topics will be particularly difficult: scalability of quantitative verification; compositional quantitative frameworks; effective runtime steering; quality assurance for embedded software; efficiency of strategy synthesis for autonomous control in dynamic scenarios; and quantitative verification for stochastic hybrid systems.

Acknowledgements. This research is supported in part by ERC Advanced Grant VERIWARE and the Institute for the Future of Computing, Oxford Martin School.

References

1. PRISM website., `www.prismmodelchecker.org`
2. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
3. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time Markov chains. IEEE Transactions on Software Engineering 29, 524–541 (2003)

4. Basagiannis, S., Petridou, S.G., Alexiou, N., Papadimitriou, G.I., Katsaros, P.: Quantitative analysis of a certified e-mail protocol in mobile environments: A probabilistic model checking approach. Computers & Security 30(4), 257–272 (2011)
5. Bucur, D., Kwiatkowska, M.: On software verification for TinyOS. Journal of Software and Systems 84(10), 1693–1707 (2011)
6. Calinescu, R., Ghezzi, C., Kwiatkowska, M., Mirandola, R.: Self-adaptive software needs quantitative verification at runtime. Communications of the ACM 55(9), 69–77 (Sep 2012)
7. Calinescu, R., Grunske, L., Kwiatkowska, M., Mirandola, R., Tamburrelli, G.: Dynamic QoS management and optimisation in service-based systems. IEEE Transactions on Software Engineering 37(3), 387–409 (2011)
8. Chaki, S., Giampapa, J.: Probabilistic verification of coordinated multi-robot missions. In: Proc. SPIN'13 (2013), to appear.
9. Chen, T., Diciolla, M., Kwiatkowska, M., Mereacre, A.: Quantitative verification of implantable cardiac pacemakers. In: Proc. 33rd Real-Time Systems Symposium (RTSS). IEEE Computer Society (2012)
10. Chen, T., Forejt, V., Kwiatkowska, M., Parker, D., Simaitis, A.: Automatic verification of competitive stochastic systems. Formal Methods in System Design (2013), to appear
11. Chen, T., Forejt, V., Kwiatkowska, M., Parker, D., Simaitis, A.: PRISM-games: A model checker for stochastic multi-player games. In: Proc. TACAS'13. LNCS, Springer (2013)
12. Chen, T., Kwiatkowska, M., Simaitis, A., Wiltsche, C.: Synthesis for multi-objective stochastic games: An application to autonomous urban driving. In: Proc. QEST 2013 (2013), to appear.
13. Chen, T., Diciolla, M., Kwiatkowska, M.Z., Mereacre, A.: A Simulink hybrid heart model for quantitative verification of cardiac pacemakers. In: Proc. HSCC'13. pp. 131–136 (2013)
14. Chen, T., Diciolla, M., Kwiatkowska, M.Z., Mereacre, A.: Symbolic model checking for probabilistic timed automata. ACM Transactions on Computational Logic (2013), to appear.
15. Chen, T., Hahn, E.M., Han, T., Kwiatkowska, M., Qu, H., Zhang, L.: Model repair for markov decision processes. In: Proc. TASE'13. IEEE (2013), to appear
16. Duflot, M., Kwiatkowska, M., Norman, G., Parker, D.: A formal analysis of Bluetooth device discovery. Int. Journal on Software Tools for Technology Transfer 8(6), 621–632 (2006)
17. Egami, K., Matsumoto, S., Nakamura, M.: Ubiquitous cloud: Managing service resources for adaptive ubiquitous computing. In: PerCom Workshops. pp. 123–128 (2011)
18. Etessami, K., Kwiatkowska, M., Vardi, M., Yannakakis, M.: Multi-objective model checking of Markov decision processes. Logical Methods in Computer Science 4(4), 1–21 (2008)
19. Feng, L., Kwiatkowska, M., Parker, D.: Compositional verification of probabilistic systems using learning. In: Proc. 7th International Conference on Quantitative Evaluation of SysTems (QEST'10). pp. 133–142. IEEE CS Press (2010)
20. Feng, L., Kwiatkowska, M., Parker, D.: Automated learning of probabilistic assumptions for compositional reasoning. In: Giannakopoulou, D., Orejas, F. (eds.) Proc. FASE'11. LNCS, vol. 6603, pp. 2–17. Springer (2011)
21. Filieri, A., Ghezzi, C., Tamburrelli, G.: Run-time efficient probabilistic model checking. In: Taylor, R.N., Gall, H., Medvidovic, N. (eds.) ICSE. pp. 341–350. ACM (2011)

22. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated verification techniques for probabilistic systems. In: Bernardo, M., Issarny, V. (eds.) Formal Methods for Eternal Networked Software Systems (SFM'11). LNCS, vol. 6659, pp. 53–113. Springer (2011)
23. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D., Qu, H.: Quantitative multi-objective verification for probabilistic systems. In: Abdulla, P., Leino, K. (eds.) Proc. TACAS'11. LNCS, vol. 6605, pp. 112–127. Springer (2011)
24. Forejt, V., Kwiatkowska, M., Parker, D.: Pareto curves for probabilistic model checking. In: Chakraborty, S., Mukund, M. (eds.) Proc. ATVA'12. LNCS, vol. 7561. Springer (2012)
25. Forejt, V., Kwiatkowska, M., Parker, D., Qu, H., Ujma, M.: Incremental runtime verification of probabilistic systems. In: Proc. 3rd International Conference on Runtime Verification (RV'12). LNCS, Springer (2012)
26. Fruth, M.: Probabilistic model checking of contention resolution in the IEEE 802.15.4 low-rate wireless personal area network protocol. In: Proc. 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISOLA'06) (2006)
27. Gallina, L., Han, T., Kwiatkowska, M., Marin, A., Rossi, S., Spano, A.: Automatic energy-aware performance analysis of mobile ad-hoc networks. In: Proc. IFIP Wireless Days (WD'12) (2012)
28. Hahn, E.M., Han, T., Zhang, L.: Synthesis for pctl in parametric markov decision processes. In: NASA Formal Methods. pp. 146–161 (2011)
29. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6, 512–535 (1994)
30. Kemeny, J., Snell, J., Knapp, A.: Denumerable Markov Chains. Springer (1976)
31. Kwiatkowska, M.: Quantitative verification: Models, techniques and tools. In: Proc. ESEC/FSE'07. pp. 449–458. ACM Press (September 2007)
32. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic model checking. In: Bernardo, M., Hillston, J. (eds.) Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07). LNCS (Tutorial Volume), vol. 4486, pp. 220–270. Springer (2007)
33. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic games for verification of probabilistic timed automata. In: Proc. FORMATS'09. LNCS, vol. 5813, pp. 212–227. Springer (2009)
34. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proc. CAV'11. LNCS, vol. 6806, pp. 585–591. Springer (2011)
35. Kwiatkowska, M., Norman, G., Parker, D., Qu, H.: Assume-guarantee verification for probabilistic systems. In: Esparza, J., Majumdar, R. (eds.) Proc. TACAS'10. LNCS, vol. 6105, pp. 23–37. Springer (2010)
36. Kwiatkowska, M., Norman, G., Parker, D., Sproston, J.: Performance analysis of probabilistic timed automata using digital clocks. Formal Methods in System Design 29, 33–78 (2006)
37. Kwiatkowska, M., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. Theoretical Computer Science 282, 101–150 (2002)
38. Kwiatkowska, M., Norman, G., Sproston, J., Wang, F.: Symbolic model checking for probabilistic timed automata. Information and Computation 205(7), 1027–1077 (2007)

39. Kwiatkowska, M., Parker, D., Simaitis, A.: Strategic analysis of trust models for user-centric networks. In: Proc. 1st International Workshop on Strategic Reasoning (SR'13). EPTCS, vol. 112, pp. 53–60 (2013)
40. Kwiatkowska, M., Rodden, T., Sassone, V. (eds.): From computers to ubiquitous computing, by 2020, vol. 366(1881) (2008)
41. Lahijanian, M., Andersson, S.B., Belta, C.: Temporal logic motion planning and control with probabilistic satisfaction guarantees. IEEE Transactions on Robotics 28(2), 396–409 (2012)
42. Lakin, M., Parker, D., Cardelli, L., Kwiatkowska, M., Phillips, A.: Design and analysis of DNA strand displacement devices using probabilistic model checking. Journal of the Royal Society Interface 9(72), 1470–1485 (2012)
43. Norman, G., Parker, D., Kwiatkowska, M., Shukla, S., Gupta, R.: Using probabilistic model checking for dynamic power management. Formal Aspects of Computing 17(2), 160–176 (2005)
44. Norman, G., Parker, D., Sproston, J.: Model checking for probabilistic timed automata. Formal Methods in System Design (2012), to appear
45. Paparrizos, I.K., Basagiannis, S., Petridou, S.G.: Quantitative analysis for authentication of low-cost RFID tags. In: LCN. pp. 295–298 (2011)
46. Susu, A.E., Acquaviva, A., Atienza, D., Micheli, G.D.: Stochastic modeling and analysis for environmentally powered wireless sensor nodes. In: Proc. WiOpt. pp. 125–134. IEEE (2008)
47. Weiser, M.: The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev. 3(3) (Jul 1999)
48. Wongpiromsarn, T., Topcu, U., Murray, R.M.: Receding horizon temporal logic planning. IEEE Trans. Automat. Contr. 57(11), 2817–2830 (2012)