# Performance Analysis for LTE Networks with Markov Decision Process

Tony Tsang, *Member, IEEE,*

***Abstract*—Long Term Evolution (LTE) has been proposed as a promising radio access technology to bring higher peak data rates and better spectral efficiency. However, scheduling and resource allocation in LTE still face huge design challenges due to their complexity. In this paper, the optimization problem of scheduling and resource allocation for separate streams is first formulated. By separating streaming scheduling and packet sorting, the scheduler is aware of probabilistic state information, fairness among the streams, and the frame weight. Our algorithm thus reduces an Markov Decision Processes(MDP) to a fully probabilistic Markov chain on which Statistical Model Checking (SMC) may be applied to give an approximate solution to the problem of checking the probabilistic Bounded Linear Temporal Logic (BLTL) property. We integrate our algorithm in a parallelized modification of the PRISM simulation framework. Extensive validation with both new and PRISM benchmarks demonstrates that the approach scales very well in scenarios where symbolic algorithms fail to do so. Simulations results with video sequences show that significant gains could be observed by our scheme in terms of spectrum efficiency, QoS of packet delay, and video quality while maintaining the fairness among the streams.**

***Index Terms*—Long Term Evolution; Markov Decision Processes; Statistical Model Checking; QoS**

## I. Introduction

Recent advance research has developed a large variety of smart mobile devices, which are powerful enough to support a wide range of multimedia traffic (e.g. VoIP, video streaming, multiplayer interactive gaming) and also legacy mobile services (e.g. voice, SMS, MMS). These new multimedia applications require high data rates and power to provide better Quality of Service (QoS). However, due to the low transmission rate and high service costs, the 3G (third generation) technology has been unsuccessful in delivering ubiquitous/high-speed mobile broadband.

To address the mobile broadband requirements, the 3GPP introduced the new radio access technology Long Term Evolution (LTE) which has the capability to move towards fourth generation (4G) wireless systems. LTE is designed to be a high data rate and low latency system that aiming to support different types of services, including web browsing, FTP, HD video streaming, VoIP, online multi-user interactive gaming and real time video. However, the use of enriched 4G services is still limited because the receivers of these services require computationally complex circuitry that drains the user equipments (UE) battery power quickly.

In our study, we first formulate the optimization problem of resource allocation for separate streams. Then, we show that it is reduced to the problem of packet scheduling. Various packet scheduling strategies for video transmission over wireless have been discussed including [1–3]. Regarded as a delay-limited capacity problem, The Earliest Deadline First (EDF) strategy is put forward to satisfy the delay constraints in [1]. Moreover, in content-aware schemes, the importance of the scheduled packet for decoders is considered as well [2, 3], i.e., Minimization Cost (MC) strategy. Nevertheless, these strategies dont refer to e-Multimedia Broadcast/Multicast Servicee (MBMS) system due to the following considerations. (I). each OFDM-based frame including multi-subcarriers is apt to be scheduled to the data from more than one stream. Obviously, it is inappropriate for multicast in view of power consumption, since each terminal needs to decode more frames including the data for its desired contents. [4]. (II). as for MBMS over a Single Frequency Network (MBSFN), the data entity is separated from the control entity. The control entity which is responsible for allocating resources has no idea of the related factors used by packet scheduling [5].

To resolve the problems above, we propose a suboptimum scheduling scheme, called the QoS-aware two-layer scheduling. The innovations lie in (I). it is up to specification of e-MBMS that a frame is allocated to one stream, thus the terminal is enabled to turn into sleep mode during several frames, when its undesired streams are being transmitted [6]. (II). the process of resource allocation is divided into two layers. In the longterm scheduling, we add the QoS-aware Scheduling Module (QASM) to the control entity, and it is able to acquire the information of queue state from the data entity, such as the packet urgency and fairness, to help decide the transmission order of streams. After that, the data entity ensures the prior transmissions of more important packets based on the frame weight in the short-term scheduling.

Model Checking [7] (MC) is a successful set of techniques aimed at providing formal guarantees (usually expressed in some form of temporal logic) for models that can be specified as transition systems. There has been a lot of interest in the MC community for extensions of the classical algorithms to probabilistic settings, which are more expressive but significantly harder to analyse. These extensions study the Probabilistic Model Checking (PMC) problem, where the goal is to find the probability that a property holds in some stochastic model.

When solving the PMC problem, it is often possible to trade-off correctness for scalability. There is extensive work on how the PMC problem can be solved through exact techniques [8–10], which compute correct probability bounds. Exact techniques do, however, rely on reasoning about the entire state space, which is widely considered to be the limiting factor in

T. Tsang is with the Hong Kong Polytechnic University, Hong Kong, e-mail:ttsang@ieee.org

their applicability to large problems. The complementary approach is known as Statistical Model Checking (SMC), which is based on selectively sampling traces of the system until enough statistical evidence has been found. Although it trades away the iron clad guarantees of PMC for statistical claims, SMC requires comparatively little memory, thus circumventing the most pressing limitation of classical PMC techniques. In addition, sampling is usually very efficient even for large systems.

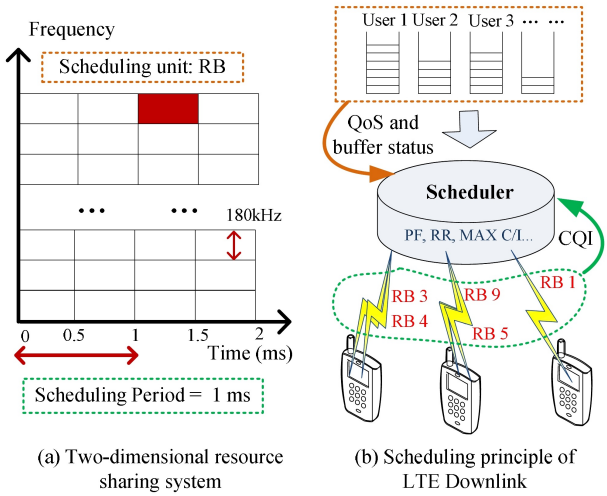
We develop and study the QoS-aware two-layer scheduling algorithm to enable the application of SMC in Markov decision processes (MDPs), the *de facto* standard for modelling discrete systems exhibiting both stochastic and nondeterministic behaviour. The main difficulty for the PMC problem in MDPs is that it requires properties to hold in *all* resolutions of nondeterminism, or *schedulers*. Properties, expressed in temporal logic and interpreted over traces, often check for bad behaviour in the modelled system. In this case, one would check that, for all schedulers, the probability of bad behaviour occurring is less than some small value.

PRISM [8] is a state-of-the-art probabilistic model checker. We implemented our algorithm in Java, using a parallelised version of PRISMs simulation framework for trace generation. This allows us to seamlessly use PRISMs specifications for MDPs. We take care to ensure that our multi-threaded modification of the framework remains statistically unbiased. We apply our algorithm to both the PRISM benchmark suite as well as to new benchmarks and perform an extensive comparison. The results show that the algorithm is highly scalable and efficient. It also runs successfully on problems that are too large to be tackled by PRISMs exact engine.

Simulations results show that QoS-aware two-layer scheduling scheme performs well in exhaustive QoS metrics including spectrum efficiency, packet delay, and video quality, while maintaining the adequate fairness among the streams.

## II. SCHEDULING AND RESOURCE ALLOCATION IN LTE DOWNLINK

Orthogonal Frequency Division Multiplexing (OFDM) radio technology has been selected as LTE downlink radio access scheme owing to its high bandwidth scalability, simple equalization, high robustness against multi-path fading and high spectral efficiency. Concerning resource allocation, OFDM-based LTE downlink can be seen as a time-frequency two-dimensional resource sharing system, as described in Fig.1 (a). Such two-dimensional resource is divided into multiple resource blocks (RBs). An RB, which last 0.5 ms in the time domain and 12 consecutive subcarriers in the frequency domain, is considered as the minimum scheduling unit. Each LTE frame lasts 10 ms and it is divided into ten equally size sub-frame, called Transmission Time Interval (TTI). Evolved NodeBs (eNodeBs), the base stations in LTE, executes the RBto- user assignment at its medium access control (MAC) layer according to the selected scheduling algorithm every TTI, as shown in Fig. 1(b).




(a) Two-dimensional resource sharing system

(b) Scheduling principle of LTE Downlink

Fig. 1. Scheduling and Resource Allocation in LTE Downlink

## III. PROBABILISTIC MODEL CHECKING FOR MARKOV DECISION PROCESS

In this section we lay the necessary formal foundations to define the probabilistic model checking problem.

### A. Real Time Modeling

The basic elements of Real Time Model are its actions, which represent activities carried out by the systems being modeled, and its operators, which are used to real time descriptions.

*Time point*

A time point is a time instant with respect to the global clock of the system; it does not have duration. It specifies the starting and stopping times of an action. Using a time point, we can instruct the system to generate an action at a particular point in time. Time point progresses consistently in all parts of the system. More formally, the time point is defined by using a discrete time domain, which contains the following properties:

$\forall t\ \exists t'\ t < t' \wedge \forall t'' : t < t'' \Rightarrow t' \leq t''$

We assume a fixed set of clocks $t = \{t_0, \ldots, t_i\}$. The special time point $t_0$, which is called the start time point, always has the value 0.

*Time Constraint*

An action can exist for a period of time; this duration is called the time constraint of the action. A time constraint has a starting and an ending point. It consists of a lower-bound and an upper-bound time point, where the lower-bound time point enables an action in a module, and the upper-bound time point disables the action at that point in time. Formally, we define a time constraint in the following:

$\mathcal{T}_i = \{[\tau_{i_{min}}, \tau_{i_{max}}] \mid \forall t_i \in T\}$ with $0 \leq \tau_{i_{min}} \leq \tau_{i_{max}}$.

### B. State Labeled Markov Decision Processes with Time Constraint

Markov decision processes [11] are a popular choice to model discrete state transition systems that are both probabilis-

tic and nondeterministic. Standard statistical model checking does not handle nondeterminism and thus cannot be directly applied to these models. Schedulers are functions used to resolve the nondeterminism in Markov decision processes (MDP). A MDP in which nondeterminism has been resolved becomes a fully probabilistic system known as a Markov chain. In the setting of Probabilistic Model Checking (PMC), it is customary to assume the existence of a state labelling function $\mathcal{L}$ that associates each state with a set of propositions that are true in that state with time constraint.

*1) Definition 1 ( Real Time Markov Decision Process):* A State Labeled Real Time Markov Decision Process (RT-MDP) is a tuple $\mathcal{M} = \langle S, \bar{s}, A, \tau, \mathcal{L}, \mathcal{T} \rangle$ where $S$ is a (finite) set of states, $\bar{s} \in S$ is an initial state, A is a (finite) set of actions, $\tau : S \times A \times S \to [0,1]$ is a transition function such that for $s \in S, a \in A$ , either $\sum_{s' \in S} \tau(s,a,s') = 1$ (a is enabled) or $\sum_{s' \in S} \tau(s,a,s') = 0$ (a is disabled), for each $s \in S$ there exists at least one action enabled from $s$ and $\mathcal{L} : S \to 2^{AP}$ is a labelling function mapping each state to the set of atomic propositions true in that state at time constraint $\mathcal{T}$ .

For each state s and enabled action $\mathcal{M} = \langle S, \bar{s}, A, \tau, \mathcal{L}, \mathcal{T} \rangle$ $a, \tau(s,a,s')$ gives the probability of taking action a in state s and moving to state $s'$ at time $t'$ . At least one action needs to be enabled at each state. The transitions are assumed to take one time step so there has a notion of real time $t_i$ . Because of this, RT-MDPs are particularly suited for reasoning about the ordering of events with being explicit about their timing $\mathcal{T}$ .

*2) Definition 2 (Markov Chain with Time Constraint):* A State Labeled discrete time Markov chain is a tuple $\mathcal{M} = \langle S, \bar{s}, A, \tau, \mathcal{L}, \mathcal{T} \rangle$ where $S$ is a (finite) set of states,$s \in S$ is an initial state, A is a (finite) set of action names. $P : S \times A \times S \to [0,1]$ is a transition function such that for $s \in S$ , $\sum_{a \in A} \sum_{s' \in S} P(s,a,s') = 1$and $\mathcal{L} : S \to 2^{AP}$ is a labelling function mapping each state to a set of atomic propositions that are true in that state at time constraint $\mathcal{T}$ .

There is a set of paths associated with each Markov chain $\mathcal{M}$. A path in $\mathcal{M}$, denoted $\pi \in \mathcal{M}$, is an infinite sequence $\pi = \bar{s} \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \cdots$of states s.t. for all $i \in \mathcal{N}$ , $P(s_i, a_i, s_{i+1}) > 0$ . Given a path $\pi$ with time constraint $\mathcal{T}$ , the $n$-th state of $\pi$, denoted $\pi^n$ , is $s_n$ ; the $k-$*prefix* of $\pi$ , denoted $\pi \mid_k$ is the finite subsequence of $\pi$ that ends in $\pi_k$ ; and the $k-$*suffix* of $\pi$ , denoted $\pi \mid^k$ is the infinite subsequence of $\pi$ that starts in $\pi^k$ with time constraint $\mathcal{T}^k$ .

The transition function $P$ induces a canonical probability space over the paths of $\mathcal{M}$ as follows. We define the function $Pr_f$ , over finite prefixes: for prefix $\hat{\pi} = \bar{s} \xrightarrow{a_0} s_1 \cdots \xrightarrow{a_{k-1}} a_k$ , $Pr_f(\hat{\pi}) \cong 1$ if $k = 0, Pr_f(\hat{\pi}) \cong P(\bar{s}, a_0, s_1) P(s_1, a_1, s_2) \cdots P(s_{k-1}, a_{k-1}, s_k)$ otherwise. This function extends to a unique measure $Pr$ over the set of (infinite) paths of $\mathcal{M}$ with time constraint $\mathcal{T}$ .

A real time scheduler for a RT-MDP resolves the nondeterminism in each state $s$ by providing a distribution over the set of actions enabled in $s$ within time constraint.

*3) Definition 3 (Real Time Scheduler):* A *memoryless scheduler* for a MDP $\mathcal{M}$ is a function $\sigma : S \times A \to [0,1]$ s.t. $\sum_{a \in A} \sigma(s,a) = 1$ and $\sigma(s,a) > 0$ only if $a$ is enabled in $s$ at time $t$ .

A real time scheduler for which either $\sigma(s,a) = 1$ or $\sigma(s,a) = 0$ for all pairs $(s,a) \in S \times A$ is called *deterministic* . In this work, by scheduler, we mean memoryless scheduler. For a discussion on this design decision, see [11].

*4) Definition 4 (Markov chain induced by a real time scheduler):* Given a MDP $\mathcal{M} = \langle S, \bar{s}, A, \tau, \mathcal{L}, \mathcal{T} \rangle$ and a scheduler for $\mathcal{M}$ , $\sigma$ , the *Markov chain* induced by $\sigma$ , is the Markov chain $\mathcal{M}^\sigma = \langle S, \bar{s}, A, P, \mathcal{L}, \mathcal{T} \rangle$ where $P(s,a,s') \cong \sigma(s,a)\tau(s,a,s')$ within time constraint $\mathcal{T}$ .

This resolution of nondeterminism will enable us to apply SMC techniques to RT-MDPs, provided we find a suitable scheduler.

## IV. Real Time Bounded Linear Temporal Logic

Linear Temporal Logic (LTL) [12, 13] is a formalism used to reason about the ordering of events without introducing time explicitly. It is interpreted over sequences of states. Each state represents a point in time in which certain propositional assertions hold. Once an event changes the truth value of these assertions, the system moves to a new state.

### A. Linear Temporal Logic Syntax and Semantics

We use linear temporal logic (LTL) to formally specify system properties. Standard LTL is built upon a finite set of atomic propositions, logical operators $\neg$ (negation) and $\vee$ (disjunction), and the temporal modal operators $\bigcirc$ (next) and $\mathcal{U}$ (until).

Formally, given a set of atomic propositions $\prod$ , the set of LTL formulas over $\prod$ can be defined inductively as follows
(1) any atomic proposition $\pi \in \prod$ is an LTL formula;
(2) if $\varphi$ and $\psi$ are LTL formulas, so are $\neg\varphi, \bigcirc\varphi, \varphi \vee \psi$ and $\varphi \mathcal{U} \psi$ .
Additional logical operators, such as $\wedge$ (conjunction), $\to$ (material implication), and temporal modal operators $\Diamond$ (eventually), and $\Box$ (always), are defined by:
(a) $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$ ;
(b) $\varphi \to \psi := \neg\varphi \vee \psi$ ;
(c) $True := p \vee \neg p$ ; where $p \in \prod$ ;
(d) $\Diamond\varphi := True \mathcal{U} \varphi$ ;
(e) $\Box\varphi := \neg\Diamond\neg\varphi$ .
A *propositional formula* is one that does not include any temporal operators.

**Continuous Semantics of LTL** : An LTL formula for the continuous-time of non linear system is interpreted over its trajectories $(x, \sigma)$ .

$\dot{x} = f_\sigma(x,d)$ ,
where $x(t) \in X \subseteq \mathcal{R}^n$ is the state at time $t$ and $d(t) \in D \subseteq \mathcal{R}^d$ is exogenous disturbance, $P$ is a finite index set , and $\{f_\sigma : p \in P\}$ is a family of nonlinear vector fields satisfying the usual conditions to guarantee the existence and uniqueness of solution for each of the subsystems.

Formally, given an LTL formula $\varphi$ without the next operator $\bigcirc$ , we can recursively define the satisfaction of $\varphi$ over a trajectory $(x(t), \sigma(t))$ at time $t$ , written $(x(t);\ \sigma(t)) \models \varphi$ , as follows:
(1) for any atomic proposition $\pi \in \prod, (x(t), \sigma(t)) \models \pi$ if and only if $\pi \in h(x(t), \sigma(t))$ ;

(2) $(x(t),\sigma(t)) \models \neg\varphi$ if and only if $(x(t),\sigma(t)) \not\models \varphi$ ;
(3) $(x(t),\sigma(t)) \models \varphi \vee \psi$ if and only if $(x(t),\sigma(t)) \models \varphi$ or $(x(t),\sigma(t)) \models \psi$ ; and
(4) $(x(t),\sigma(t)) \models \varphi \mathcal{U} \psi$ if and only if there exists $t' \geq t$ such that $(x(t'),\sigma(t')) \models \psi$ and $(x(s),\sigma(s)) \models \varphi$ for all $s \in [t,t')$ .
A trajectory $(x,\sigma)$ starting at $t_0$ is said to satisfy $\varphi$ , written $(x;\ \sigma) \models_{t_0} \varphi$ , if $(x(t_0),\sigma(t_0))) \models \varphi$ . If the initial time is not significant, we simply write $(x,\sigma) \models \varphi$ .

**Discrete Semantics of LTL** An LTL formula for a switched system given by the family of transition systems is interpreted over its switching executions.

$\{\mathcal{T}_p := (\mathcal{Q}, \mathcal{Q}_0, \xrightarrow{p}) : p \in P\}$ .

Given an LTL formula $\varphi$ , we can recursively define the satisfaction of $\varphi$ over a switching execution $(q,p) = (q_0,p_0)(q_1,p_1)(q_2,p_2)\cdots$ at position $i$, written $(q_i,p_i) \models \varphi$ as follows:
(1) for any atomic proposition $\pi \in \prod, (q_i,p_i) \models \pi$ if and only if there exists $x_i \in \mathcal{T}^{-1}(q_i)$ such that $\pi \in h(x_i,p_i)$ ;
(2) $(q_i,p_i) \models \neg\varphi$ if and only if $(q_i,p_i) \not\models \varphi$ ;
(3) $(q_i,p_i) \models \bigcirc\varphi$ if and only if $(q_{i+1},p_{i+1}) \models \varphi$ ;
(4) $(q_i,p_i) \models \varphi \vee \psi$ if and only if $(q_i,p_i) \models \varphi$ or $(q_i,p_i) \models \psi$ ;
(5) $(q_i,p_i) \models \varphi \mathcal{U} \psi$ if and only if there exists $j \geq i$ such that $(q_j,p_j) \models \psi$ and $(q_k,p_k) \models \varphi$ for all $k \in [i,j)$ .
A switching execution $(q,p) = (q_0,p_0)(q_1,p_1)(q_2,p_2)\cdots$ is said to satisfy $\varphi$ , written $(q,p) \models \varphi$ if $(q_0,p_0) \models \varphi$ .

### B. Real Time Bounded Linear Temporal Logic

Sampling and checking of paths needs to be computationally feasible. Since LTL may require paths of arbitrary size, we instead use *Real Time Bounded* RT-LTL, which requires only paths of bounded size [14]. In addition, for each path, we may identify a smallest prefix that is sufficient to satisfy or refute the property, which we will call the *minimal sufficient prefix* of the path. This notion is useful in practice to avoid considering unnecessarily long paths. The syntax and semantics of RT-BLTL are summarized in Table I.

| **Syntax:** $\varphi := p \mid \neg\varphi \mid \varphi \vee_{\mathcal{T}} \varphi \mid F_{\mathcal{T}}^{\leq n}\varphi \mid G_{\mathcal{T}}^{\leq n}\varphi \mid \varphi \mathcal{U}_{\mathcal{T}}^{\leq n}\varphi$ | |
|---|---|
| **Semantics:** $\pi \models \varphi$ iff $\cdots$ | |
| **if $\varphi$ is...** | **Samantics** |
| $p$ | $p \in \mathcal{L}(\pi^0)$ |
| $\neg\varphi_1$ | $\pi \not\models \varphi_1$ |
| $\varphi_1 \vee_{\mathcal{T}} \varphi_2$ | $\pi \models \varphi_1$ or $\pi \models \varphi_2$ with $\mathcal{T}$ |
| $F_{\mathcal{T}}^{\leq n}\varphi_1$ | $\exists_{i<n}$ : $\pi\,\vert^i \models \varphi_1$ with $\mathcal{T}$ |
| $G_{\mathcal{T}}^{\leq n}\varphi_1$ | $\forall_{i<n}$ : $\pi\,\vert^i \models \varphi_1$ with $\mathcal{T}$ |
| $\varphi_1 \mathcal{U}_{\mathcal{T}}^{\leq n}\varphi_2$ | $\exists_{i\leq n}\ \forall_{k\leq i}$ : $\pi\,\vert^k \models \varphi_1$ and $\pi\,\vert^i \models \varphi_2$ with $\mathcal{T}$ |

where $\pi = \pi^0 \xrightarrow{a_0} \pi^1 \xrightarrow{a_1} \pi^2 \cdots$ is a path. $\pi\,\vert^i$ is the suffix of $\pi$ starting at $\pi^i$ . $\mathcal{L}$ is given and maps states, $\pi^i$ , to the subset of atomic propositions that are true in that state with time constraint $\mathcal{T}_i$ .

Informally, $F_{\mathcal{T}}^{\leq n}\varphi_1$ means "$\varphi_1$ will become true within $n$ transitions in time constraint $\mathcal{T}_n$"; $G_{\mathcal{T}}^{\leq n}\varphi_1$ means "$\varphi_1$ will be remain true for the next $n$ transitions in time constraint $\mathcal{T}_n$" and $\varphi_1 \mathcal{U}^{\leq n}\varphi_2$ means "$\varphi_2$ will be true within the next $n$ transitions and $\varphi_1$ remains true until then in time constraint $\mathcal{T}_n$ ". The classical connectives follow the usual semantics.

### C. Probabilistic and Statistical Model Checking

Let $\mathcal{M}$ be a RT-MDP, $\varphi$ be a RT-BLTL property and $0 < \theta < 1$ be a rational number in time constraint $\mathcal{T}$ . The problem of PMC for these parameters, denoted $P_{\leq\theta}(\varphi)$ , lies in deciding whether $\forall_\sigma : Pr(\{\pi : \pi \in \mathcal{M}^\sigma\ ,\ \pi \models \varphi\}) \leq \theta$ that is, "Is the probability of the set of paths of $\mathcal{M}^\sigma$ that satisfy $\varphi$ at most $\theta$ for all schedulers $\sigma$?"

The formula $\varphi$ usually encodes an undesirable property, e.g. reaching an error state or violating a critical condition. If we can find the scheduler that maximises the probability of satisfying $\varphi$, then we can compare that probability with $\theta$ to answer the PMC query, since all other schedulers will achieve a lower value. It can be easily shown that *deterministic* schedulers are sufficient for achieving this maximum probability.

Some state-of-the-art techniques for the PMC problem in RT-MDPs [8, 9] usually rely on symbolic methods to encode the state-action graph of the RT-MDP in compact representations [15, 16] . Using this representation, such approaches compute the exact maximum probability of satisfying the property through an iterative method that propagates information throughout the state space.

Fully probabilistic models, like Markov chains with time constraint, exhibit probabilism but not nondeterminism. These models admit only the trivial scheduler that selects the single available distribution at each state. The PMC problem for fully probabilistic systems then reduces to deciding whether the probability of satisfying $\varphi$ under that scheduler is greater than $\theta$ . For solving this problem, there exists an efficient sampling based technique known as Statistical Model Checking (SMC).

SMC comes in two flavours: *hypothesis testing* solves the PMC problem stated above; independent traces of a system are analysed until a meaningful decision can be reached about the hypothesis "probability of satisfaction of $\varphi$ is smaller than $\theta$ ". Without going into much detail, a quantity that measures the relative confidence in either of the hypotheses, called the *Bayes factor* (or the likelihood ratio in the case of the SPRT [17] ) , is dynamically recomputed until enough statistical evidence has been gathered to make a decision. The other kind of SMC is *interval estimation* , where traces are sampled until a probability of satisfaction can be estimated within some confidence interval [18] . This value is then compared against $\theta$ . Hypothesis testing is often faster than interval estimation, whereas interval estimation finds the actual probability of satisfying $\varphi$ . The suitability of either of the techniques, naturally, depends on the specific problem at hand.

In conclusion, since SMC solves the PMC problem statistically on Markov chains, SMC for MDPs reduces to the problem of finding an optimal scheduler for the PMC problem.

## V. Probabilistic and Statistical Model Checker Tool

We present the probabilistic model checker PRISM [19] which exploits the computation power of (general purpose) graphics processing units (GPUs).

Probabilistic model checking [20] is a branch of model checking which has been successfully used for the analysis of models that have a probabilistic/stochastic nature. These

models cover a broad spectrum of applications ranging from communication protocols like FireWire and Bluetooth, to biological networks that model gene expression.

In traditional model checking one usually aims at proving absolute logical correctness of the analyzed model against a given property. In probabilistic model checking the correctness of the properties is quantified with some probability. The properties are expressed in extensions of the traditional temporal logics such that the quantitative probabilistic aspects are captured.

PRISM is a probabilistic model checker which was developed initially at the University of Birmingham and currently is being developed at the University of Oxford. PRISM is an open source tool and written in Java and C++. During the years the tool has gained a significant popularity and it has been tested on various case studies. A quite comprehensive summary of PRISM applications can be found on the tool web page [21] .

PRISM supports three types of models: discrete- and continuous Markov chains (DTMCs and CTMCs), and Markov decision processes (MDPs), The models are specified using the PRISM modeling language which is based on the Reactive Modules formalism. Systems are described as a set of modules executed in parallel. Each module contains transitions to which probabilities are associated in various ways, depending on the model type.

Properties are specified in the logics PCTL and CSL, which are probabilistic extensions of the logic CTL. PCTL is used to specify properties of DTMC models, whereas CSL is used in the context of CTMCs.

### A. Functionality Overview

We begin with a brief overview of the current functionality of the PRISM tool. Items in boldface denote new or improved features in version 4.0, which are described in more detail in the remainder of the paper.

- Modelling and construction of many types of probabilistic models, now including **probabilistic timed automata** ; all can be augmented with costs or rewards, in the case of PTAs yielding the model of **priced probabilistic timed automata**;
- Model checking of a wide range of quantitative properties, expressed in a language that subsumes the temporal logics PCTL, CSL, LTL and PCTL*, as well as extensions for quantitative specifications and costs/rewards;
- Multiple model checking engines, both symbolic (BDD-based) and **explicit state**; and a variety of probabilistic verification techniques, such as symmetry reduction and **quantitative abstraction refinement**;
- A **discrete-event simulator**, with support for **statistical model checking** methods, including confidence-level approximation and acceptance sampling;
- Model import options, e.g. from Systems Biology Markup Language (SBML);
- **Optimal adversary/strategy** generation for nondeterministic models;
- A GUI, with model editor, simulator and graphing, or command-line tool;
- A **benchmark suite** of probabilistic models and associated properties.

### B. Probabilistic Timed Automata (PTAs)

Probabilistic timed automata (PTAs)[23, 24] are finite-state automata enriched with real-valued clocks, in the style of timed automata, and with discrete probabilistic choice, in the style of Markov decision processes (MDPs).

Clocks are real-valued variables, whose values increase simultaneously over time. Predicates over clock variables called *guards* and *invariants* are assigned to transitions and states, respectively, imposing restrictions on when transitions can occur and how long can be spent in a state. For ease of modelling, we can also add finite-ranging *data variables* to a PTA. Transitions between states can reset clocks (to integer values) and update data variables. This is done probabilistically: the target state, clock resets and variable updates are specified by a discrete probability distribution. The choice between multiple transitions, as well as the elapse of time (subject to invariant satisfaction) are both *nondeterministic* .

PTAs can be augmented with information about costs incurred or, equivalently, rewards gained (PRISM uses the latter terminology). This model, often known as *priced* PTAs, allows reasoning about a wide range of additional prop- erties, such as energy consumption or resource usage. PRISM supports *linearly* priced PTAs, where costs/rewards are accumulated at a rate proportional to the elapse of time, with the rate depending on the current state (and data variables).

Finally, we mention that PTAs also support *parallel composition* (as for timed automata and MDPs), in which multiple PTAs operate concurrently, synchronizing on transitions with matching labels. For precise details, see [25] .

PRISM uses a uniform modelling language for all the probabilistic models that it supports, including PTAs. This is a textual language, based on guarded command notation. To support PTAs, PRISM 4.0 adds a new clock data type. Clock variables can appear (as convex expressions) in guards, on the left-hand side of a command, and can be reset, like any other variable, with an update on the right-hand side. A new *invariant* keyword is introduced to allow expression of invariants.

PRISM analyzes two main classes of properties for PTAs: (i) the minimum/maximum probability of reaching a target, possibly within a time bound; and (ii) the minimum/maximum expected reward accumulated until a target is reached. Two verification methods are implemented:

- Quantitative abstraction refinement [26] constructs and analyzes a series of probabilistic abstractions, automatically refining at each step to produce more precise results.
- Digital clocks [25] performs an automatic model translation to an equivalent finite-state, discrete-time model (with integer-valued clocks)1 and then uses PRISM's existing MDP model checking techniques.

### C. Other PRISM Components and Features

*1) Quantitative abstraction refinement toolkit:* As described above, PRISM's default PTA verification technique

uses *quantitative abstraction refinement* [26]. This can be seen as a quantitative analogue of classical counterexample-guided abstraction refinement. It provides a fully automatic approach to verification of large or infinite-state probabilistic systems, by iteratively building, analyzing and refining increasingly precise probabilistic abstractions. In addition to PTAs [27] , the same approach has been applied to verification of probabilistic software (using predicate abstraction and SAT) [28] and to finite-state MDPs [26] . While these implementations all build abstractions of MDPs as stochastic two-player games, the same approach can be used to, for example, build abstractions of Markov chains as Markov decision processes. Quantitative abstraction refinement is implemented in PRISM in the form of an extensible toolkit, with support for multiple model types, refinement strategies and configurable optimizations.

*2) Explicit-state probabilistic model checking library:* PRISM already features several model checking engines (called "MTBDD", "sparse", and "hybrid"), all either fully or partially symbolic (i.e. BDD-based). The tool now incorporates a new, entirely explicit-state probabilistic model checking library, implemented in Java and based on sparse matrix data structures. It supports stochastic two-player games, Markov decision processes and discrete- and continuous- time Markov chains. The code is designed to serve as a general purpose library, either for inclusion in other techniques or for prototyping new model checking algorithms. For example, the library is used in the abstraction-refinement toolkit, in which probabilistic models need to be constructed and modified on-the-fly, a task not well-suited to symbolic implementations.

*3) Simulation engine and statistical model checking:* Version 4.0 of PRISM incorporates a newly rewritten version of its *discrete-event simulation engine*. This provides efficient random generation of paths through PRISM models, both for the purposes of debugging models and to support so-called *statistical* (or ap- proximate) model checking techniques [17, 29] . PRISM now offers two types of such analysis. For *quantitative properties* (e.g. $P =?[\cdot]$ in PRISM notation), it either generates a confidence interval (based on a given confidence level) or a probabilistic guarantee of correctness, using the Chernoff-Hoeffding bound [29] . For bounded properties (e.g. $P < 0.1[\cdot]$ ), it uses *acceptance sampling* [17] implementing Wald's sequential probability ratio test (SPRT). Statistical model checking offers significantly improved scalability, in comparison to conventional probabilistic model checking techniques, and applies to a broader class of models.

*4) Optimal adversary (strategy) generation:* PRISM's MDP verification implementation now includes the ability to generate optimal adversaries (also known as strategies). This means that, when PRISM computes the minimum or maximum value for a probabilistic reachability (or expected reward) property, it can also generate an adversary (resolution of non-determinism in the model) that produces it. This can be used to debug or analyze the results of model checking, for example in order to generate probabilistic counterexamples, or to produce an optimal solution for a scheduling problem. Furthermore, by using the digital clocks engine, optimal adversaries can also be generated for PTAs.

*5) The PRISM benchmark suite:* There are a large number of existing PRISM case studies, distributed with the tool, included in publications and on the tool website [21] . These are widely used, for example to evaluate new model checking techniques, or to compare model checking implementations and tools. Unfortunately, there are often several different variants of each model and it is not always easy to locate a particular class of models or properties. The PRISM benchmark suite [22] aims to provide a comprehensive source of freely-available benchmarks for probabilistic model checking. It includes a large selection of probabilistic models, of varying types and sizes, and corresponding properties for model checking, grouped by type.

## VI. QoS-aware Two-Layer Scheduling Scheme

The conceived novel scheduling strategy targets real time service provisioning in the LTE downlink. It has been built on two distinct levels (see Fig. 2) that interact together in order to dynamically assign radio resources to user equipment (UE). They take into account the channel state, the data source behaviors, and the maximum tolerable delays.

At the highest level, an innovative resource allocation algorithm, frame level scheduler, namely FLS, defines frame by frame the amount of data that each real time source should transmit to satisfy its delay constraint. To solve the problem using a solution with a low computational complexity, FLS exploits a discrete-time linear control loop [30]. Once FLS has accomplished its task, the lowest layer scheduler, every transmission time interval (TTI), assigns resource blocks (RBs) using the proportional fair (PF) algorithm [31] by considering bandwidth requirements of FLS.

In other words, FLS defines on the long run (i.e., in a single frame) how much data should be transmitted by each data source. The lowest layer scheduler, instead, allocates resource blocks in each TTI to achieve a trade-off between fairness and system throughput. It is important to note that FLS does not take into account the channel status. On the contrary, the lowest layer scheduler assigns RBs first to flows hosted by UEs experiencing the best channel quality and then (i.e., when these flows have transmitted the amount of data imposed by FLS) it considers the remaining ones. In particular, the lowest layer scheduler decides the number of TTIs/RBs (and their position in the time/frequency domains) in which each real time source will actually transmit its packets. It is very important to remark that the proposed approach is very general and it is independent on the model used for describing incoming data. For this reason, we do not need stochastic flow models. In fact, the control theoretic approach describes a flow as a signal modelling the bit-rate produced by the application layer.

### A. Frame Streaming Scheduling

A QoS-ware two-layer scheduling scheme is devised, where $w_k$ is divided into the streaming weight $ws_k$ and frame weight $I_{k,m}$ . In the first layer frame streaming scheduling, streaming weight is determined by Multi-cell/Multicast Coordination Entity (MCE) at Multicast Channel Scheduling Period (MSP)

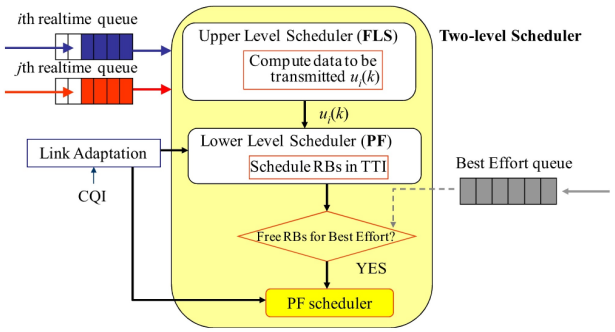



Fig. 2. QoS-aware Two-level Scheduling Algorithm

level. And then, evolved Node Bs (eNB) performs the packet sorting based on the results of streaming scheduling at TTI-level.

With the help of QoS-aware Scheduling Module (QASM), the following parameters offered by eNBs at the end of MSP, would help MCE to decide the transmission order for the next MSP. A certain eNB is enough since the action of each eNB is identical. Considering the cost of additional signaling, Delay Tolerance Factor (DT) and Fairness Penalty Factor (FP) are included to guarantee the throughput and fairness among the streams. Such scheduling is called Time-out-Based Scheduling Strategy (TBS) here.

$$DT_k = \frac{\mathcal{T}_{delay_{HoL}}}{\mathcal{T}_{PDB_k}} \tag{1}$$

$$\mathcal{T}_{delay_{k,HoL}} = t - T_{k,HoL}\,, \tag{2}$$

$$FP_k = \frac{1}{\frac{scheduled\_total_{\mathcal{T}_k}}{received\_total_{\mathcal{T}_k}}}\,. \tag{3}$$

where $\mathcal{T}_{delay_{k,HoL}}$ is the period from the time spot $T_{k,HoL}$, i.e., when the head of line (HOL) packets arrived at the queue, to the current time spot $t$ for the streaming $k$. $\mathcal{T}_{PDB_k}$ is the Packet Delay Budget for video streaming $k$ indicated by QCI.

The fairness is earliest proposed in unicast systems [32], we adopt it into the e-Multimedia Broadcast/Multicast Service (e-MBMS) system. $scheduled\_total_{\mathcal{T}_k}$ is the throughput of streaming $k$ during a period. $received\_total_{\mathcal{T}_k}$ is the amount of received packets in this period for streaming $k$.

After acquiring DT and FP, QASM would determine the streaming weight as

$$ws_k = \frac{received\_total_{\mathcal{T}_k}}{scheduled\_total_{\mathcal{T}_k}} \times exp(\frac{\mathcal{T}_{delay_{HoL}}}{\mathcal{T}_{PDB_k}})\,. \tag{4}$$

Finally, the transmission order in the bundle is determined along with QCI for non-multiplexed streams. The stream in the bundle with a larger $ws_k$ is prior transmitted.

### B. Packet Sorting

To improve the video quality at receivers, the scheduler in eNB performs the packet sorting at TTI-level after streaming scheduling, called *Cost-based Sorting Strategy* (CSS).

The binary indicator $\delta_k$ is used to show whether the streaming $k$ is scheduled completely or not, that is, whether there is any packet left in the queue to be scheduled.

$$\delta_k = \begin{cases} 0, \text{if the streaming} k \text{is scheduled completely} \\ 1, \text{otherwise} \end{cases} \tag{5}$$

The corresponding streaming $k^*$, which is to be scheduled could be determined by the following equation

$$k^* = arg\,max_{k=1,2,\ldots,K}\, ws_k \cdot \delta_k\,. \tag{6}$$

Despite being sufficient to achieve maximum probabilities, deterministic schedulers are a poor choice for exploring the state space through simulation: sampling with a deterministic scheduler provides information *only* for the actions that it chooses. Probabilistic schedulers are more flexible, explore further, and enable reinforcement of different actions. Thus, we always use probabilistic schedulers in the exploration part of our algorithm.

Ideally,$\sigma$ converges to a near-deterministic scheduler, but due to our commitment to exploration, it will never do so completely. Before using Statistical Model Checking (SMC) to answer the Probabilistic Model Checking (PMC) question, we thus greedily determinise $\sigma$. More precisely, we compute a scheduler that always picks the best estimated action at each state. Formally, DETERMINISE($\sigma^*$) is a new scheduler with the help of equation (6), for a determined streaming $k^*$, CSS could be described as (7)

$$\sigma^* = arg\,max_{\sigma}\, I_{k^*,\sigma}\,, \tag{7}$$

where the more important packet with a higher frame weight is prior allocated in CSS during the MSP.

We thus hope to redirect the residual probabilities of choosing bad actions to the promising regions of the state space. In practice, this step makes a significant difference. Generally, QoS-aware Two-Layer Scheduling can be described as follows:

| QoS-aware Two-layer Scheduling Scheme | | |
|---|---|---|
| 1) | | Initialization |
| | a) | Set $\delta_k = 1$ for all $k \in \{1, 2, \cdots, K\}$ . |
| | b) | Set $\omega_{n,k,t} = 0$ for all $n \in \{1, 2, \cdots, N\}$, $k \in \{1, 2, \cdots, K\}$, and $t \in \{1, 2, \cdots, T\}$ |
| | c) | Set $i = 1$ and $j = 1$ |
| 2) | | Determine $ws_k$ in MCE, where $ws_k$ is defined as (4) for all $k \in \{1, 2, \cdots, K\}$ . Then, MCE informs eNBs of the results of resource allocation. |
| 3) | | eNBs receive the MCH Scheduling Information (MSI). |
| 4) | | While $j \leq \mathcal{T}$ or $\delta_k = 0, \forall k$ , in eNBs |
| | a) | While $i \leq N$ |
| | | i. Find $k^*$where it is defined as (6). |
| | | ii. Find the $\sigma^*$as (7) for a given $k^*$, then the selected packet is allocated to the pair $i$ of RBs in TTI $\mathcal{T}_j$ . |
| | | iii. Update $\delta_k, \forall k$ , according to (5), |
| | | iv. Update $i = i + 1$ . |
| | b) | Update $j = j + 1$ . |
| | c) | Set $i = 1$ . |
| 5) | | The procedure of resource allocation is complete. |

Since the interval time $\mathcal{T}$ of scheduling is enlarged, our scheme is suboptimum in the case of conventional scheduling strategies at TTI-level. However, from the view of realization, it ensures that one TTI $\mathcal{T}_i$is allocated to one stream. Moreover, differing from the current semi-dynamic scheduling in LTE system, QoS of packet delay, fairness and the frame weight are considered in the long-term and short-term scheduling respectively, to aim to approach the performances achieved by the conventional strategies.

### C. Number of Runs

Although we will show that the scheduler packet sorting stage converges towards frame streaming schedulers, at any given point we cannot quantify how close to frame streaming the candidate scheduler is. Statistical claims are possible, however. If the current candidate is sufficient to settle the original Probabilistic Model Checking (PMC) query, the algorithm can stop immediately. If it is not, it may be restarted after a reasonable number of improvement iterations. These restarts help our algorithm finding and focusing on more promising parts of the state space it might have missed before. Algorithms like this are called biased Monte Carlo algorithms. Given a confidence parameter ($p$) on how likely each run is to converge, we can make a statistical claim up to arbitrary confidence ($\eta$) on the number of times we have to iterate the algorithm, $T_{\eta,p}$ :

*Bounding Theorem* [33] : For a false-biased, $p$-correct Monte Carlo algorithm (with $0 < p < 1$) to achieve a correctness level of $(1-\eta)$, it is sufficient to run the algorithm at least a number of times:

$$T_{\eta,p} = \frac{\log_2 \eta}{\log_2(1-p)} \tag{8}$$

This result guarantees that, even in cases where the convergence of the scheduler learning procedure in one iteration is improbable, we will only need to run the procedure a relatively small number of times to achieve much higher confidence. Taking all these considerations into account, the main Statistical Model Checking (SMC) procedure for Markov decision processes (MDPs) is laid out in Algorithm QoS-aware Two-layer Scheduling Scheme. An important requirement of this algorithm and Bounding Theorem is that we have a positive probability of convergence to an frame streaming scheduler during scheduler learning.

## VII. Performance Evaluation

We evaluate our procedure on several well-known benchmarks for the PMC problem. First, we use one easily parametrisable case study to present evidence that the algorithm gives correct answers and then we present systematic comparisons with PRISM [8]. Our implementation extends the PRISM simulation framework for sampling purposes. Because we use the same input language as PRISM, many off-the-shelf models and case studies can be used with our approach.

*Reinforcement Heuristics* : Our approach allows us to tune the way in which we compute quality and reinforcement information without destroying guarantees of convergence (under easily enforced conditions) but netting significant speedups in practice. These optimizations range from negatively reinforcing failed paths to reinforcing actions differently based on their estimated quality.

### A. Simulation Results

The performance of the proposed algorithm will be evaluated and compared with three traditional scheduling algorithms Proportional Fair (PF), Round Robin (RR) and Best CQI (B-CQI) in normal mode - no Discontinuous Reception DRX). The evaluation and comparison are done in same simulation environment and parameter. Evaluation will be done on key performance evaluation parameters; as described in above subsection.

All the schedulers are used in the same simulation setup as presented in the following Table. The receivers of all the UE are switched-on all the time that means no power is being saved by the UEs. The traditional schedulers which are designed to work in non-DRX environment are being compared with Proposed Scheme. However, the Proposed Scheme specially considers active and normal modes of UEs. Therefore, other schedulers may overwhelm the Proposed Scheme in one or more performance evaluation parameters.

| Parameters | Values |
|---|---|
| eNodeB radius | 250 m |
| Number of sectors per eNodeB | 3 |
| Target area | Single sector |
| Number of UEs | 0-100 |
| eNodeB total TX power | 20 W |
| Number of antennas (MIMO) | 4 TX, 3 RX |
| Fading models | Fast fading |
| UE Speed | 5 km/h |
| Operating frequency band | 2 GHz |
| System channel bandwidth | 5 MHz |
| Number of RBs | 25 |
| GBR | 25 kbps |
| CQI reporting Every | TTI |
| Traffic model | Video |
| Video packet generation interval | 20 ms |
| Video delay threshold | 100 ms |
| Power saving mechanism | DRX Sleep |
| DRX on duration | 1 TTI |
| DRX In-Active duration | 4 TTIs |

Figure 3, shows systems throughput performance when the simulation is run for 5000 TTI. The results show that Best CQI (B-CQI) scheduler performed the best because it chooses the UEs, which have the best channel conditions in the uplink through CQI feedbacks. The PF scheduler performed the second best in this regard because it tries to balance the system throughput with the fairness. The Proposed Scheme performed not as good as B-CQI and PF scheduler because it is not designed to maximize system throughput rather, it designed to provide good QoS. The three markers point to the time when the Proposed Schemes system throughput performance degraded significantly. The reason is the throughput of some UEs had started to go below the GBR limit due to bad channel condition, and the scheduler tried to compensate it by assigning more resources. The RR scheduler performed not so well, but its throughput is more stable than any other scheduler because it treats all the UEs equally regardless of their channel conditions or requirements.

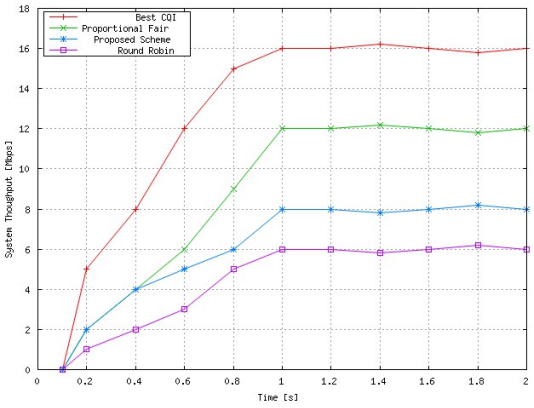



Fig. 3. Downlink System Thoughput vs. Time

Figure 4, shows the effect of number of users on packet delays for all four schedulers. The packet delay threshold for video is 100 ms according to LTE QCI otherwise the packet will be discarded. When the number of users increases, the most of the time UE switched off which result in packets start to get delayed. Figure 4, shows that RR performed best and Proposed Schemes performed second best. Both of these curves followed a linear pattern while the PF initially started well, but its performance degraded significantly after 20 ms duration. The B-CQI performed worst in terms of packet delays because it is designed to achieve maximum systems throughput thus it disregards fairness and delay constraints.

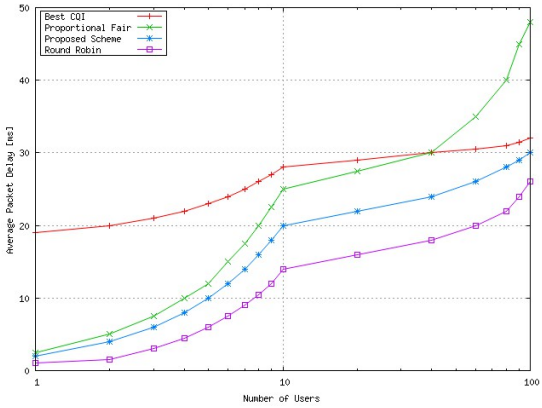



Fig. 4. Average Packet Delay vs. Number of Users

## VIII. Conclusion

In this paper, we proposed a new QoS-aware Two-layer downlink scheduling algorithm for delay sensitive traffic (Video). QoS-aware Two-layer scheduling algorithm is divided into the streaming scheduling and packet sorting by introducing dynamic QoS-related factors, such as the packet urgency and fairness among the streams. Streaming scheduling determines the transmission order of the multi-streams in MCE. And then, packet sorting ensures the transmissions of more important packets in eNB.

Combining classical SMC and reinforcement learning techniques, we have proposed what is, to the best of our knowledge, the first algorithm to solve the PMC problem in probabilistic nondeterministic systems by sampling methods. We have implemented the algorithm within a highly parallel version of the PRISM simulation framework. This allowed us to use the PRISM input language and its benchmarks.

The Proposed Scheme endeavors to provide better QoS by decreasing packet delay, improve fairness among the UE and considering the QoS requirement of multimedia service. It has the capability to assure QoS in non-power saving environment. The Proposed Scheme is compared with the traditional schemes according to different QoS attributes through simulations. In a normal power environment, the Proposed Scheme performs well in terms of throughput among the UEs with acceptable packet delay.

In future work, a longer simulation environment will be used with multiple eNodeBs. The mobility effect on QoS

will be evaluated by considering the handover procedure. The performance of Proposed Scheme will be examined with Deep Sleep mode of operation and its comparison with DRX Light Sleep mode.

## REFERENCES


[1] Sun, Siyue, Yu, Qiyue; Meng, Wei-Xiao; Li, Cheng, A configurable dual-mode algorithm on delay-aware low-computation scheduling and resource allocation in LTE downlink, Wireless Communications and Networking Conference (WCNC), 2012.

[2] Mushtaq, M.S., Shahid, A., Fowler, S., "QoS-Aware LTE Downlink Scheduler for VoIP with Power Saving", IEEE 15th International Conference on Computational Science and Engineering (CSE), Pages 243 - 250, 5-7 Dec. 2012.

[3] P. V. Pahalawatta, R. Berry, etc., Content-aware Resource Allocation and Packet Scheduling for Video Transmission over Wireless Networks, IEEE J.Select. Areas Commun., vol. 25, no. 4, pages. 749V759, May 2007.

[4] P. Hosein and T. Gopal, Radio Resource Management for Broadcast Services in OFDMA-Based Networks, in Proc. IEEE ICC, pp.271-275, May 2008.

[5] Y. Chen, Statistical Multiplexing for LTE MBMS in Dynamic Service Deployment, in Proceeding. IEEE VTC, pp.2805-2809, May 2008.

[6] 3GPP TS 36.300, Evolved Universal Terrestrial Radio Access (EUTRA) and Evolved Universal Terrestrial Radio Access Network (EUTRAN); Overall description; Stage 2 (Release 10), v10.0.0, June 2010.

[7] Edmund M. Clarke Jr., Orna Grumberg, and Doron A. Peled. Model Checking. The MIT Press, 1999.

[8] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Prism 4.0: Verification of probabilistic real-time systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, CAV, volume 6806 of Lecture Notes in Computer Science, pages 585-591. Springer, 2011.

[9] Christel Baier, Edmund M. Clarke, Vassili Hartonas-Garmhausen, Marta Z. Kwiatkowska, and Mark Ryan. Symbolic model checking for probabilistic processes. In ICALP, pages 430V440, 1997.

[10] Frank Ciesinski and Marcus Größer. On probabilistic computation tree logic. In Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, Joost-Pieter Katoen, and Markus Siegle, editors, Validation of Stochastic Systems, volume 2925 of Lecture Notes in Computer Science, pages 147V188. Springer, 2004.

[11] David Henriques, João Martins, Paolo Zuliani, André Platzer, and Edmund Clarke. Statistical model checking for Markov decision processes. Technical Report CMU-CS-12-122, Computer Science Department, Carnegie Mellon University, 2011.

[12] Amir Pnueli, The temporal logic of programs. In FOCS, pages 46-57. IEEE Computer Society, 1977.

[13] Z. Manna and A. Pnueli, The Temporal Logic of Reactive and Concurrent Systems: Specification. Springer, vol. 1, 1992.

[14] Paolo Zuliani, Andr é Platzer, and Edmund M. Clarke. Bayesian statistical model checking with application to simulink/stateflow verification. In HSCC, pages 243-252, 2010.

[15] E.M. Clarke, M. Fujita, P.O. McGeer, K.L. McMillan, and J.C. Yang. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. In Int. Workshop on Logic Synthesis, 1993.

[16] Luca de Alfaro, Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Roberto Segala. Symbolic model checking of probabilistic processes using mtbdds and the kronecker representation. In Susanne Graf and Michael I. Schwartzbach, editors, TACAS, volume 1785 of Lecture Notes in Computer Science, pages 395V410. Springer, 2000.

[17] Hakan L. S. Younes and Reid G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In Ed Brinksma and Kim Guldstrand Larsen, editors, CAV, volume 2404 of Lecture Notes in Computer Science, pages 223-235. Springer, 2002.

[18] Richard Lassaigne and Sylvain Peyronnet. Probabilistic verification and approximation. Ann. Pure Appl. Logic, 152(1-3):122V131, 2008.

[19] M.Z. Kwiatkowska, G. Norman, D. Parker, PRISM: Probabilistic Symbolic Model Checker, Computer Performance Evaluation, Modelling Techniques and Tools 12th International Conference, TOOLS 2002, volume 2324 of Lecture Notes in Computer Science, pages 200-204, Springer, 2005.

[20] M. Kwiatkowska, G. Norman, D. Parker. Stochastic Model Checking, Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation, volume 4486 of Lecture Notes in Computer Science, pages 220-270, Springer, 2007.

[21] http://www.prismmodelchecker.org.

[22] http://www.prismmodelchecker.org/benchmarks/

[23] Jensen, H.: Model checking probabilistic real time systems. In: Proc. 7th Nordic Workshop on Programming Theory, pages 247-261, 1996.

[24] Kwiatkowska, M., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. TCS 282, pages 101-150, 2002.

[25] Kwiatkowska, M., Norman, G., Parker, D., Sproston, J.: Performance analysis of probabilistic timed automata using digital clocks. FMSD 29, pages 33-78, 2006.

[26] Kattenbelt, M., Kwiatkowska, M., Norman, G., Parker, D.: A game-based abstrac- tion-refinement framework for Markov decision processes. FMSD 36(3), 2010.

[27] Kwiatkowska, M., Norman, G., Parker, D.: Stochastic games for verification of probabilistic timed automata. In: Proc. FORMATS'09. pages. 212-227, Lecture Notes in Computer Science, 2009.

[28] Kattenbelt, M., Kwiatkowska, M., Norman, G., Parker, D.: Abstraction refinement for probabilistic software. In: Proc. VMCAI'09. pages 182-197, Lecture Notes in Computer Science, 2009.

[29] Herault, T., Lassaigne, R., Magniette, F., Peyronnet, S.: Approximate probabilistic model checking. In: Proc. VMCAI'04. pages 307-329, Lecture Notes in Computer Science, 2004.

[30] K. J. Astrom and B. Wittenmark, Computer Controlled Systems: Theory and Design, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[31] E. Dahlman, S. Parkvall, J. Skold, and P. Beming, 3G Evolution HSPA and LTE for Mobile Broadband. New York: Academic, 2008.

[32] P. Svedam, S. Wilson, etc., A QoS-aware Proportional Fair Scheduling for Opportunistic OFDM, in Proc. IEEE VTC, pp.558-562, September 2004.

[33] Gilles Brassard and Paul Bratley. Algorithmics - theory and practice. Prentice Hall, 1988.


**Tony Tsang** received the BEng degree in Electronics & Electrical Engineering with First Class Honours in U.K., in 1992. He received the Ph.D from the La Trobe University (Australia) in 2000. He was awarded the La Trobe University Postgraduation Scholarship in 1998. He is a Lecturer at the Hong Kong Polytechnic University. Prior to joining the Hong Kong Polytechnic University, Dr. Tsang earned several years of teaching and researching experience in the Department of Computer Science and Computer Engineering, La Trobe University. His research interests include mobile computing, networking, protocol engineering and formal methods. Dr. Tsang is a member of the ACM and the IEEE.