Control-based event-driven bandwidth allocation scheme for video-surveillance systems

Gautham Nayak Seetanadi, Karl-Erik Årzen and Martina Maggio

Department of Automatic Control, Lund University, Lund, Sweden

1. Introduction

Large number of interconnected devices in complex computational infrastructures has increased the need to share common system resources (e.g., CPU, network bandwidth, storage). Sharing of resources leads to contention within the connected devices for access to the constrained resource. In these constrained dynamical systems, performance problems are mitigated mainly through two methods. The first is through over-provisioning of the constrained resource causing under-utilisation at non-peak loads and is not an economical solution. Over-provisioning is also not possible in some systems due to lack of...
accurate knowledge about resource requirements. The second preferred method is by employing entities that can adapt their utilisation to use only a fraction of the constrained resource. But this scenario where the entities adjust their requirements in the presence of multiple independent loops can cause disruptive effects and lead to system instability [1].

This paper considers one such constrained dynamical system, a video-surveillance system composed of a network manager and cameras that share a communication channel. The cameras capture a stream of images and encode each captured image using a quality factor that in turn regulates its instantaneous bandwidth utilisation. The encoded frame is then transmitted over the network to a central node, the network manager.

The network manager is a decision node that allocates a fraction of total available bandwidth to all the cameras connected in the network. The manager ensures that the bandwidth is allocated to the cameras in a fair manner based on the errors between the bandwidth allocated by it and the actual bandwidth utilisation by the cameras. Bandwidth reallocation to the cameras can be performed either periodically or in an event-based manner.

The periodically triggered manager is fast and predictable but has the drawback of unnecessary communication due to frequent messages and non-trivial triggering period selection. If the period is too large then the manager reacts slowly to changes in the camera bandwidth requirements. On the other hand, a small period causes large communication overhead due to the large number of control messages sent to the cameras. The guarantees on the performance of the periodic manager are inherently obtained as shown in [2]. The event based manager is triggered based on changes related to the scene captured by the cameras. This minimises network reconfigurations leading to few calculations but complicates guarantees provided on the behaviour of the system.

The cameras in the network are adaptive in nature and regulate their bandwidth utilisation by encoding the captured images. Images captured directly by the camera are large in size and unsuitable for direct transmission. The size of images after encoding them is dependent on the encoding technique used and the amount of compression decided by the quality factor. Image quality denotes the amount of information retained in the image after encoding with a higher quality generally leading to a clearer image but with a large frame size. The resulting frame size is also dependent on the artefacts in the images captured, arising due to lighting, movement, nature among others [3–5], leading to variation in sizes for images encoded with the same quality factor.

The presence of these multiple, event-based and time-triggered control strategies can affect the performance and stability of the camera system. Model checking has shown promise in verification of such complex systems with entities displaying stochastic behaviours. It has already been applied successfully to validate and prove relevant properties of real-time systems [6,7]. Applying it to the camera network model allows us to prove properties
like stability, convergence, number of manager interventions and number of dropped frames [8].

We use a probabilistic model checker PRISM [9], to verify different properties of the system by including probabilistic disturbances in the camera model. We also show that using Statistical Model Checking (SMC) helps to reduce the impact of state-space explosion at the expense of relaxed guarantees on system performance.

The contributions of this paper are as follows:

• Realistic camera model implementation by incorporating dynamics that arise during image capture
• Comparison of system behavior with varying camera parameters for model with disturbances
• Evaluation of statistical model checking of the camera network with relaxed guarantees
• Expanded experiments on the real camera network with larger number of cameras for larger number of triggering thresholds

This paper expands on the results from [8] in several directions, both from the model checking (theoretical) perspective and from the implementation (practical) standpoint. On the theoretical side, we verify properties of probabilistic camera models using both classical model checking and statistical model checking. On the experimental evaluation side, we analyse system performance for a larger number of triggering thresholds of the event based manager. Finally, the paper also analyses the effect of threshold selection on the number of manager interventions and frames dropped by the cameras.

**Paper Organisation:** The rest of the paper is organised as follows. First we introduce the camera network architecture in Section 2 and discuss the behaviour of the cameras. The section also describes the two different triggering strategies of the network manager. Section 3 gives an introduction to model checking and describes the different types of model verification. Section 4 describes the implementation of the camera network in PRISM [9], the model checking tool used in this paper. Sections 5 and 6 discuss the results obtained using model checking and the camera test-bed respectively. Finally, we discuss related work in seven and conclude the paper in Section 8.

2. System model

This section describes the architecture and modelling of the camera surveillance network. We also introduce the notation used in the remainder of the paper in this section.
2.1. System architecture

The system consists of a set of cameras connected to a central manager via a shared Ethernet network. At each time instant $t$, we define a set of active cameras $C_t = \{c_1, \ldots, c_n\}$ where $n$ is the number of active cameras. Each camera in the network captures a stream of images, encodes, and transmits them to the network manager. The camera encodes each image ensuring that the resulting frame size is less than or equal to the bandwidth allocated by the network manager. The manager allocates bandwidth to all the cameras in the network at each instant it is invoked. Bandwidth allocation to all the cameras is the network is recalculated by evaluating the camera requirements and its relative error compared to other cameras in the network.

The architecture enables the use of two independent control strategies to achieve a good match between the resource allocated to each camera and their resource consumption:

- **Image fitting at camera $p$:** To fit a frame $w$ to the allocated bandwidth, $B_{p,w}$, by choosing the appropriate image quality, $q_{p,w}$.

- **Bandwidth allocation at the manager:** The amount of bandwidth to allocate to each of the cameras while ensuring it does not exceed the total amount of available bandwidth $H$.

**Figure 1:** System Overview

Figure 1 shows an example of the camera network with four cameras, $(c_1, c_2, c_3, c_4)$ connected to the network manager through a router. The unidirectional blue arrows denote the control signals sent from the manager to the cameras. The control signals contain information about the amount of bandwidth allocated to each cameras and their appropriate communication slot. The black arrows denote images sent from the cameras to the network manager over the shared Ethernet connection.

The manager enforces bandwidth restrictions for each camera using flexible time-triggered Switched Ethernet (FTT-SE) [10]. FTT-SE dynamically allocates...
Ethernet bandwidth to all the cameras independently and frames that are larger in size compared to the allocated bandwidth are dropped.

2.2. Camera behaviour

The cameras in the network capture a set of frames and encode them using the chosen encoding technique (In this paper, we use Motion JPEG (MJPEG)). The images are then transmitted to the manager over a common Ethernet connection during each transmission period. Figure 2 shows an original image and an image after encoding it with a quality factor. The original frames captured by the camera consist of redundant information and are unsuitable for direct transmission. Most encoding techniques use a quality factor $q_{p,w}$ to encode images and reduce the amount of redundant information contained. Higher values of $q_{p,w}$ leads to frames of larger sizes with fewer artefacts. Motion JPEG (MJPEG) generally has quality factor bounded such that $q_{p,w} \in (0, 100]$.

The camera generates a frame of size $s^*_{p,w}$ as shown in Equation 1. $s_{p,max}$ is the estimated maximum frame size that is dependent on the number of pixels in the image. The frame size is then saturated to set global limits for the maximum and minimum frame sizes. $\delta s_{p,w}$ indicates the stochastic disturbance that occurs due to changes in the scene being captured such as movement, illumination, number of people in frame to name a few.

$$
\hat{s}_{p,w}^* = 0.01 \cdot q_{p,w} \cdot s_{p,max} + \delta s_{p,w},
$$

$$
\hat{s}_{p,w} = \max\{s_{p,min}, \min\{s_{p,max}, \hat{s}_{p,w}^*\}\}.
$$

Figure 2 Camera Encoding
The camera frame size controller is synthesised using Equation 2. The frame size \( s_{p,w} \) is determined by regulating the quality \( q_{p,w} \) with a simple PI controller. The controller minimises the error \( e_{p,w} \), between the allocated bandwidth \( B_{p,w} \), and the resulting frame-size \( s_{p,w} \) as shown in Equation 3. \( e_{p,w} \) is then normalised, limiting the error between \(-1\) and \(1\) in most operating conditions.

This control design achieves a zero-steady state error (when \( \delta = 0 \)) ensuring that the size of the encoded frame is equal to the amount of bandwidth allocated by the manager. The quality parameter \( q_{p,w} \) is the control signal for the camera controller and it roughly corresponds to a compression level of the frame. The values of \( k_p \) and \( k_i \) are the proportional and integral gains of the camera controller respectively.

\[
e_{p,w} = \frac{B_{p,w-1} - s_{p,w-1}}{B_{p,w-1}} \tag{3}
\]

\[
q^*_p = k_p \cdot e_{p,w} + k_i \cdot \sum_{t=1}^{w-1} e_{p,t} \tag{4}
\]

\[
q_{p,w} = \max\{q_{\min}, \min\{q_{\max}, q^*_p\}\} \tag{5}
\]

*Figure 2.*: Camera Encoding

### 2.3. Manager behaviour

The network manager \( \mathcal{M} \) acts on a global scale deciding the optimal bandwidth to allocate to the cameras \( C_t \) connected in the network. The manager also ensures that the total allocated bandwidth does not exceed the fixed capacity \( \mathcal{H} \) of the network. \( \mathcal{M} \) selects the bandwidth allocation vector \( b_{s,w} \) at each instant \( t \) that it is invoked such that,

\[
\forall t, \mathcal{M} \text{ selects } b_{s,t} = [b_{1,t}, \ldots, b_{n,t}] \text{ such that } \sum_{p=1}^{n} b_{p,t} = 1 \tag{6}
\]

Each element of \( b_{s,t} \) determines the fraction of the bandwidth assigned to the respective camera in the network. The bandwidth assignment is done periodically during each allocation period.

Equations (7) and (8) form the core of bandwidth calculations in the manager. \( B_{p,w} \) denotes the amount of bandwidth allocated to camera \( p \) for frame \( w \) which is a function of the \( b_{p,t}M \) \( w \), the fraction of bandwidth (between 0 and 1).

\[
B_{p,w} = b_{p,t} \cdot M \cdot w \cdot \mathcal{H} \tag{7}
\]
If the frame size is greater than the amount of bandwidth allocated, i.e. $s_{p,w} > B_{p,w}$, the frame is dropped as it is outdated and the corresponding transmission slot is lost.

$$b_{p,t+1} = b_{p,t} + \varepsilon \cdot \{ -\lambda_{p,t} \cdot f_{p,t} + b_{p,t} \cdot \sum_{i=1}^{n} [\lambda_{i,t} \cdot f_{i,t}] \} \quad (8)$$

The fraction of bandwidth allocated to each camera is calculated using Equation 8. $\lambda_{p,t} \in [0,1]$ denotes the relative importance of camera $c_p$ in the network and the extent of bandwidth reallocation by the manager. $\varepsilon$ is a constant that limits the change in bandwidth trading off between the responsiveness of the manager and its robustness to disturbances.

The matching function $f_{p,t}$ determines a match between the frame size $s_{p,w}$ and the allocated bandwidth $B_{p,w}$ for the $w$-th frame of camera $p$. In the current implementation, the matching function is chosen to be the same as the normalised error used in the camera given by Equation 3. In order for the system to adhere to properties such as starvation avoidance, balance, convergence and stability, the matching function has to satisfy the following properties:

(P1a) $f_{p,t,w} > 0$ if $B_{p,w} > s_{p,w}$,
(P1b) $f_{p,t,w} < 0$ if $B_{p,w} < s_{p,w}$,
(P1c) $f_{p,t,w} = 0$ if $B_{p,w} = s_{p,w}$;
(P2a) $f_{p,t,w} \geq f_{p,t,w-1}$ if $q_{p,w} \leq q_{p,w-1}$,
(P2b) $f_{p,t,w} \leq f_{p,t,w-1}$ if $q_{p,w} \geq q_{p,w-1}$;
(P3a) $f_{p,t,w} \geq f_{p,t,w-1}$ if $b_{p,t,w} \geq b_{p,t,w-1}$,
(P3b) $f_{p,t,w} \leq f_{p,t,w-1}$ if $b_{p,w} \leq b_{p,t,w-1}$.

The manager is triggered with either a time-triggered or an event-based strategy. The time-triggered manager is activated periodically with period $\pi_M$, a multiple of $\pi_{alloc}$, the allocation period. At time 0 the manager divides the available bandwidth among the cameras equally. The following network manager interventions, happening at times $\{\pi_M, 2\pi_M, 3\pi_M, \ldots\}$, assign the bandwidth based on the relationship from Equation 8. This triggering strategy has some drawbacks. Triggering the manager has a computational overhead with linear complexity on the number of cameras. The choice of $\pi_M$ is also non-trivial. Choosing a small value of $\pi_M$ invokes the manager too frequently causing constant bandwidth reallocation. On the other hand, a large value leads to slow reactions to changes in the scenes captured by the cameras.

An event-based manager reallocates bandwidth only when necessary, avoiding frequent bandwidth recalculations [8]. We formulate the following rules to ensure the event-based manager reacts to changes in the network.

* $C_t \neq C_t$. When the number of cameras in the network changes. The manager starts by allocating equal amount of bandwidth to all the cameras and then proceeds to allocate bandwidth depending on the threshold and calculates it using on Equation 8.

• \((\exists c_p \text{ s.t. } t_{\text{thr}} < |e_{p,w}|) \land (t \mod \tau_{\text{alloc}} = 0)\). When all the cameras have finished transmission and a camera has an error greater than the absolute value of the threshold \(t_{\text{thr}}\).

The system allows cameras to dynamically join and leave the network. For convenience Tables 1 and 2 give the summary of terms used in this paper.

3. Model checking

This section gives an introduction to model checking and the two models used for verification. First we discuss the modelling checking methodology and describe the two camera models constructed. Then we define the different camera properties constructed for evaluation and model parameters used for system evaluation.

3.1. Model checking introduction and motivation

Cyber-physical systems and communication networks are growing in complexity. This rapid growth also increases the presence of errors and makes the reliability of these systems a critical issue. Model checking has shown promise in detecting errors and providing guarantees in large scale cyber-physical systems.

Traditionally, software testing has been the dominant methodology to verify software systems. However it has the drawbacks of being a time consuming, complex, and costly procedure. In some cases testing is not feasible due to the inability to cover all test cases. In general, model checking is a more powerful

| Term | Explanation |
|------|-------------|
| \(c_p\) | Camera number \(p\) |
| \(s_{p,w}\) | Size of frame \(w\) captured by camera \(p\) |
| \(q_{p,w}\) | Quality of frame \(w\) captured by camera \(p\) |
| \(\delta_{p,w}\) | Stochastic disturbance on frame \(w\) |
| \(e_{p,w}\) | Error during transmission of frame \(w\) of camera \(p\) |
| \(b_{p,w}\) | Amount of bandwidth allocated to camera \(p\) for frame \(w\) |
| \(k_p, k_i\) | Proportional and Integral gains of the controller in the camera |

**Table 1.** Explanation of camera terms.

| Term | Explanation |
|------|-------------|
| \(M\) | Manager |
| \(b_{p,w}\) | Bandwidth allocated to camera \(p\) for the transmission of the \(w\)-th frame |
| \(b_{p,t}\) | Fraction of bandwidth calculated for camera \(p\) at time \(t\) (between 0 and 1) |
| \(\tau_{\text{alloc}}\) | Allocation period |
| \(\mathcal{H}\) | Global available bandwidth |
| \(\varepsilon\) | Scaling factor that decides the aggressiveness of the manager |
| \(\lambda_{p,t}\) | Decides the importance of camera in the network [2] |
| \(f_{p,t}\) | Error of camera \(p\) at time \(t\) |
| \(\tau_{\text{thr}}\) | Threshold that triggers the manager |

**Table 2.** Explanation of manager terms.
tool compared to testing as it uses a model and provides more useful information about system behaviour.

Hardware systems have high manufacturing costs, and therefore it is crucial to catch design errors early. Hardware correctness is generally verified using emulation and simulation. Similar to software verification, exploring all the possible input combinations and verifying all outputs is often not feasible. Model checking can be used in both these scenarios.

Model checking is used to verify certain properties of a system model. In model based verification, the models are described in a mathematically precise way. Systems are modelled as Markov chains and verified using existing model checking and verification tools, some of which are PRISM [9], SPIN [11], UPPAAL [12].

Model checking also makes it possible to generate strategies for optimal operation of the modelled system while subjected to required constraints. This strategy of operation depends on precise modelling of the system. It is therefore important to eliminate any ambiguities or incompleteness in the model but include non-determinism and a characterisation of the uncertainty.

Figure 3.: Model Checking Methodology

Figure 3 shows the general work-flow of applying model checking for property verification. The system model represents the functioning of the system as an MDP consisting of states and transitions. System property is the required behaviour of the system that is evaluated. The surveillance camera network model and properties are explored in sections 3.2 and 3.3 respectively.

Figure 3 Model Checking Methodology
Probabilistic Model Checking (PMC) involves modelling of systems with stochastic behaviour and the verification of properties of such systems. In PMC, systems are modelled as state-transition systems where transitions between the different states are decided by probabilities. Discrete-Time Markov Chains (DTMC), Markov Decision Processes (MDP) and Probabilistic Timed Automata (PTA) are some of the models that support probabilistic transitions. The properties to be verified are expressed using probabilistic temporal logic, such as probabilistic reward computation-tree logic (PRCTL) [13]. The camera surveillance network is modelled using Markov Decision Processes (MDP).

A MDP is a 4-tuple \((\mathcal{S}, \mathcal{A}, P, R)\), where \(\mathcal{S}\) is a set of finite states, \(\mathcal{A}\) is a set of actions, \(P : (s, a, s') \rightarrow \{p \in \mathbb{R} \mid 0 \leq p \leq 1\}\) is a function that encodes the probability of transitioning from state \(s\) to state \(s'\) as a result of an action \(a\), and \(R : (s, a, s') \rightarrow \mathbb{N}\) is a function that encodes the reward received when the choice of action \(a\) determines a transition from state \(s\) to state \(s'\).

### 3.2. Deterministic vs probabilistic models

Deterministic models are simple system representations that do not contain any probabilistic transitions between the different states in the system model. The probability \(P\), from each state \(s\) to the next state \(s'\) of the system MDP, is equal to 1. Deterministic models also have a single outgoing transition from each state. This simplifies property verification and allows verification of larger number of sub-systems in the system model. These deterministic models are traditionally suited for verification of simple systems that do not show any stochastic behaviour [14].

However, these simple models do not capture stochastic behaviour that is generally present in real-life systems. Uncertain and unreliable behaviour in systems arises due to inherent system properties such as randomised communication algorithms, coin tosses, dice rolls with random outcomes and so on. Uncertainty in systems can also arise due to the presence of adversaries attacking the system. Uncertainty in systems can also manifest as stochastic variance in system variables such as disturbances during image capture, packet drops in network communication and so on.

This uncertainty can be modelled by attaching probabilistic distributions to the different transitions in the system model. Probabilistic models allow multiple probabilistic transitions from state \(s\) to the next state \(s'\) while ensuring the sum of all probabilities from a single state sum to 1. For example, a fair five sided die can be modelled through the transition from the initial state to the result of the die with five transitions, where each transition is attached a probability of \(1/5 = 0.2\). Similarly, a fair six sided die can be modelled using a coin toss to prevent probabilistic imbalances arising due to divisibility problems [15].
Modelling of probabilistic systems requires expert knowledge about the system compared to simple deterministic models. The complexity present in probabilistic models also leads to the problem of state space explosion [16,17] and hinders the size of system built for verification. However, due to their large application possibilities, probabilistic models have been used for verification of systems successfully in various industrial settings [18].

Uncertainties in the camera network model arise during image capture due to sudden disturbances due to movement, changes in lighting and various other factors [3]. These disturbances are captured by the factor δ in Equation (1). Previous work [8] considered a linear relationship was considered between quality and frame size and did not consider the disturbance δ during system verification. Although this was a simplification of the modelling, it was useful in proving certain properties of the system. This paper considers the disturbance δ through probabilistic transitions in the camera model.

### 3.3. Property verification

The characteristics of the system that is being verified are modelled as properties using formal language. The properties are modelled using precise mathematical equations and checked for the presence of appropriate states and paths in the state-space. The different properties are classified into Qualitative and Quantitative properties. Qualitative properties evaluate definite state reachability or full probability of the property. For example, \( P_{\geq 1} \left[ G \ H_{\text{alloc}} = H \right] \), where \( H \) indicates the total amount of bandwidth available and \( H_{\text{alloc}} \) indicates the total amount of bandwidth allocated to all the cameras connected in the network. \( G \) denotes that the property is checked globally. The property checks for complete bandwidth allocation with a probability of 1.

Quantitative properties evaluate whether the system reaches a state or exceeds a property with a threshold. They are framed similar to qualitative properties with a relaxed probability requirement. Considering the same PRTCL property as above, \( P_{\geq 1/3} \left[ G \ H_{\text{alloc}} = H \right] \) relaxes the previous strict probability to \( 2/3 \). The relaxed bound simplifies property verification and accommodates larger model sizes.

### 3.4. Classical vs statistical model checking

Model checking tools construct the whole state-space by building states and transitions, and storing it in memory. These tools have an upper limit on memory usage to limit runoff programs, restricting the size and complexity of system models. Classical model checking builds the whole state-space irrespective of the presence (or absence) of probabilities in transitions. Probabilistic models capture complex behaviours of the system but are larger in size in...
comparison to deterministic models. This state-space explosion causes memory issues during verification of complex system models.

Statistical model checking (SMC) [19] performs Monte Carlo simulations of the system with stochastic behaviours. SMC simulates the system for a finite number of runs and applies techniques from the area of statistics to evaluate it for property verification (or failure). These Monte Carlo simulations of the system provide estimates on the probabilistic measure on executions. SMC provides relaxed bound on system performance useful for verification of large complex systems, where classical model checking fails due to memory limitations [17].

3.5. Strategy generation

Strategy generation in a probabilistic system selects an optimal action or state transitions to resolve non-deterministic choices and maximise rewards (or minimise cost). PRTCL reward minimisation operator $R_{\min=}^r$ selects a strategy that resolves $r$ where $r$ is the reward.

For example, $R_{\min=}^r[F \phi]$ minimises the total reward $r$ along paths that lead to state satisfy the state formula $\phi$. In the camera network, some relevant strategies are ‘minimise the number of manager interventions’, ‘minimise the total number of dropped frames’, ‘minimise a defined cost’ or maximise the bandwidth utilisation’. Strategy generation is considered to be out of scope of this paper.

4. System implementation

This section describes the modelling of the camera surveillance system and disturbances during image capture using PRISM model checker. We also describe the construction of different camera properties and model parameters used for model checking.

4.1. System modelling

The different entities of the system are modelled as a set of modules in PRISM. In the camera network model, the different modules are namely the manager, the scheduler and the cameras. The cameras are modelled identically to each other but have their own independent modules. The scheduler is introduced into the camera model to determine the order in which the different modules in the system are invoked. This module is necessary as the network manager is modelled to act only when required in an event-based manner. The camera modules are scheduled in a turn-based manner and then the scheduler evaluates the necessity of invoking the event-based manager. An example schedule is show below.
[Turn1] scheduler → manager → cam₁ → cam₂ → ... camₙ

scheduler → manager → cam₁ → cam₂ → ... camₙ

scheduler → cam₁ → cam₂ → ... camₙ

... ...

... ...

... ...

scheduler → manager → cam₁ → cam₂ → ... camₙ

The scheduler invokes the manager and all the cameras in order during Turns 1 and 2. Turn 3 shows a scenario when the cameras have reached equilibrium (the frame sizes match the allocated bandwidths) and the manager is not invoked. It is invoked again during Turn X when the allocated bandwidth requires recalculation.

4.2. Disturbance modelling

Disturbances occur in the camera due to changes in the scene captured by the camera. Equation 1 models the behaviour of a camera with disturbances where δ denotes the disturbance. This uncertainty is modelled in PRISM using probabilities on the different transitions. The camera module is composed of two different formulas for calculation of the frame size. The first formula considers the value of δ to be 0. The second formula includes δ during frame size calculations and results in a frame of larger size. The current model assigns a probability of 0.7 to the linear formula and 0.3 to the formula for frame size with disturbances to ensure that the total probability is 1.

4.3. Properties

Model checking involves verifying relevant properties of a given system model. These properties are generally the desired/undesired behaviours of the model. Real-life requirements are converted to formal language and input into the model checker as shown in figure 3. The following properties for the camera model are verified

1. Convergence:

   \[ P_{\text{min}} = \text{?}[F(G!(\text{any_change_event}))] \]

Convergence is defined as ‘nothing happens in the model after some time’ for model checking. The PRCTL formula above returns the minimum probability
$P_{\text{min}}$ of the model finally $F$, globally $G$ such that no changes occur in the system $!\{\text{any change event}\}$. $!\{\text{any change event}\}$ is a variable that records changes in the bandwidth allocations and can be used as a measure to gauge if the system has converged.

(2) **Number of Resource Manager Interventions:**

$$R_{\text{max}}^{\text{rm calls}} = ?[F(\text{rounds} = \text{max rounds})]$$

This property is used to quantify the number of manager interventions during one verification run of the model. The reward operator $R_{\text{max}}^{\text{rm calls}}$ here counts the maximum number of resource manager calls $\text{rm calls}$ finally $F$ after a certain number of rounds $\text{max rounds}$ have passed.

(3) **Number of dropped frames:**

$$R_{\text{max}}^{\text{dropped frames}} = ?[F(\text{rounds} = \text{max rounds})]$$

Similar to the previous property, this property quantifies the number of dropped frames. The reward operator $R_{\text{max}}^{\text{dropped frames}}$ counts the maximum number of dropped frames $\text{dropped frames}$, finally $F$ after a certain number of rounds $\text{max rounds}$ have passed.

(4) **Total cost:**

$$R_{\text{max}}^{\text{total cost}} = ?[F(\text{rounds} = \text{max rounds})]$$

total_cost calculates the cost incurred by a triggering policy, i.e. a chosen threshold. The property is tuned differently depending upon actions that are penalising to a system. For example, dropping of frames, manager interventions, or addition/deletion of new cameras affect the camera network differently dependent upon the configuration of the system. In this paper, we penalise dropped frames and manager interventions by 10 and 1, respectively. In PRISM this can be defined as $\text{total cost} = 10 \cdot \text{dropped frames} + 1 \cdot \text{rm calls}$.

### 4.4. Model parameters

The performance of the system is evaluated with respect to different parameter values using model checking. This enables faster parameter evaluation and tuning with varying disturbances in the model. For the camera system the following parameters are considered,

$\varepsilon$ is the scaling factor that denotes the aggressiveness of the system. It is a trade off between responsiveness and stability of the manager. Higher values of $\varepsilon$ increase convergence speed of the system but may cause overshoots, whereas small values of $\varepsilon$ result in a slower but more stable behaviour during network re-configurations. In the current model, $\varepsilon$ is set to 0.5.

$\lambda \in (0, 1)$ denotes the importance of the camera in the network. When $\lambda = 0$, the network manager does not reallocate bandwidth to the camera and the onus is on the camera to adjust its bandwidth without manager interventions.
When \( \lambda = 1 \), the network manager adjusts the bandwidth allocated to the camera at a higher rate with no quality changes expected from the camera. This methodology grants the architecture the ability to incorporate multiple types of cameras in the same network. For example, if camera \( c_p \) is incapable of changing its encoding quality, then \( \lambda_p \) is set to 1.

The triggering threshold \( \tau_{thr} \in (0.1, 0.9) \), determines the activation of the manager for network re-configurations. The choice of \( \tau_{thr} \) is non-trivial and is discussed in detail in [8]. Small values of \( \tau_{thr} \) lead to quicker manager activation but cause a higher number of network re-configurations. Large values of \( \tau_{thr} \) reduce the number of manager interventions and network re-configurations at the cost of information loss.

\( k_{k,p} \) and \( k_{i,p} \) are the proportional and integral gains of camera \( p \) that determine the aggressiveness with which the camera regulates its quality. Large controller gains generally lead to faster camera adaptations and larger quality changes.

\( \tau_{thr} \) and \( k_{i,p} \) are analysed in more detail in the following sections.

5. Verification results

This section discusses the verification results for the camera models built in PRISM. The section also shows the state-space evolution and property verification results for the two different types of the model.

**Model I**: The first model of the camera system describes the system with no probabilities. The camera modules use a linear equation to derive the frame size \( \hat{s}_{p,w} \) given a quality \( q_{p,w} \). Equation 1 with \( \delta = 0 \) disregards the effect of disturbance during image capture. This simplification in frame size calculation results in a smaller state-space that enables performance evaluation of a larger number of cameras. Model II also provides stronger property verification guarantees using classical model checking compared to Model II.

**Model II**: The second model of the camera network incorporates disturbances during image capture as probabilistic transitions. The probabilities are added during the frame size calculation as shown in Equation 1. A realistic model of the camera model is realised using two independent equations. The first equation is linear and already described in Model I. A high probability is attached to this equation. The second equation generates a larger frame size \( \hat{s}_{p,w} \) for a given quality \( q_{p,w} \) compared to the linear equation. A lower probability is attached to this equation. The probability distribution models the scenario in which disturbances occur in the generated frames.
Table 3. State space size evolution.

| Number of Cameras | Model I | Model II |
|-------------------|---------|----------|
| 2                 | 123     | 309      |
| 3                 | 157     | 5537     |
| 4                 | 183     | 9297     |
| 5                 | 213     | 54,561   |
| 6                 | 247     | N/A      |
| 7                 | 277     | N/A      |
| 8                 | 307     | N/A      |
| 9                 | 337     | N/A      |
| 10                | 363     | N/A      |

5.1. Classical model checking results

Table 3 shows the evolution of the size of camera state-space. The camera and manager modules are modelled in minimal number of states while capturing the relevant behaviour of the physical camera network. The system also imposes an upper limit on the number of frames transmitted to the manager to ensure convergence of property verification. Without the limit, the state space grows infinitely and makes model checking impossible due to its infinite size.

Model I without probabilities grows at a lower rate compared to the model with probabilities, Model II. PRISM is unable to build the model of the camera network with probabilities due to memory limitations for Model II with the number of cameras larger than 5.

5.1.1. Deterministic model results

The subsection analyses the impact of controller gain $k_i$ on the performance of the camera model without disturbances. Figure 4 shows the effect of $k_i$ on the percentage of frames dropped by the manager due to bandwidth restriction violations. The plot on the left shows the percentage when controller gain $k_i$ is 5 and the right plot when $k_i$ is 25, for an increasing number of cameras. There is a very low percentage of dropped frames when the number of cameras in the network is 2, 4, 5 and 10 due to floating point calculations leading to quicker convergence of bandwidth calculations. The frames are dropped mostly during network initialisation and initial regulation of camera quality. This is transient.

Figure 4 Percentage of Dropped Frames for Model I

---

G. NAYAK SEETANADI ET AL.
phase of the camera network as the total amount of available bandwidth is lower than the ideal bandwidth required for each camera. This causes quality $q_p$ regulation in camera frame sizes. If the current frame size, $s_{p,w}$ is greater than the amount of bandwidth $B_{p,w}$ allocated by the manager, then the frame is dropped. Larger $k_i$ leads to faster bandwidth adaptation and convergence, leading to fewer dropped frames.

Figure 4: Percentage of Dropped Frames for Model I

5.1.2. Probabilistic model results

Adding probabilities to the model increases the size of its state-space as shown in Table 3. This increases the complexity of property verification due to the large state-space. PRISM is unable to build models with large number of cameras due to memory issues resulting in a reduced number of cameras to 5. Figure 5 shows the percentage of dropped frames for two values of controller gain $k_i$. Having a low value of $k_i$, causes a frame drop greater than 75% while using a gain of higher value reduces the dropped frames. It is also seen that the amount of dropped frames is higher for Model II compared to Model I. This is due to the presence of dynamics in the model giving a more complex analysis of the system. Using a probabilistic model in combination with model checking enables choosing controller parameters using more detailed information. There is also faster convergence for a particular number of cameras as seen in the previous experiment.

Figure 5: Percentage of Dropped Frames for Model II

5.2. Statistical model checking results

Statistical model checking allows verification of an increased number of camera modules with disturbances using model II. Compared to classical model checking, statistical model checking is able to build state-space of the camera network with 20 cameras instead of 5. Figure 6 shows the effect of threshold, $\tau_{thr}$ on the percentage of dropped frames by the cameras. The percentage of dropped

Figure 5 Percentage of Dropped Frames for Model II
frames increases for higher values of $\tau_{thr}$ and lower values of $k_i$ as the camera adaptation is slower and the manager does not intervene for small disturbances. Higher values of $k_i$ lead to reduced number of dropped frames for all thresholds due to faster camera adaptations. Increasing $\tau_{thr}$ does not increase the number of dropped frames as the cameras regulate their bandwidth utilisation quicker. There is faster convergence for values 4, 5, 10 and 20 due to divisibility and model building convergence. We do not apply SMC to the linear camera model, Model I, as the state-space of the model is smaller in size and classical model checking provides more robust bounds on property verification.

Figure 6.: Percentage of Dropped Frames for Model II using SMC

6. Experimental results

This section describes the experimental results obtained using the camera test-bed. First, we provide context for the experiment by describing results from our previous work [8]. Next we discuss costs obtained for varying triggering thresholds $\tau_{thr}$ with experiments conducted to evaluate the number of dropped frames and manager interventions.

6.1. Experimental setup

The camera network is composed of multiple cameras and a network manager connected through local Ethernet. The multiple units of the network are implemented on a PC with Intel Core i7-4790 8 core CPU with 32 GB RAM. The current implementation of the network consists of three off-the-shelf Logitech C270 cameras.

The experiment has the following setup:
- Proportional Gain: $k_{p,1} = k_{p,2} = 10$
- Integral Gain: $k_{i,1} = k_{i,2} = 0.5$
- Minimum Quality: $q_{1,\min} = q_{2,\min} = 15$
- Maximum Quality: $q_{1,\max} = q_{2,\max} = 85$
- Camera Importance: $\lambda_1 = 0.7, \lambda_2 = 0.3$
- Scaling Factor: $\varepsilon = 0.4$
• Allocation Period: $\pi_{\text{alloc}} = 30\text{ ms}$
• Total Network Bandwidth: $\mathcal{H} = 4\text{ Mbps}$.

The total network bandwidth is constrained to ensure camera adaptation.

### 6.2. Results

Previous work [20] describes the results for camera adaptation with time-triggered manager. Similarly [8], investigated the implementation of an event-triggered manager. This paper builds on the previous experimental results by examining the operation of the event-triggered manager for a larger number of threshold values.

Recall that there are two control strategies:

• The network manager, that decides the amount of bandwidth to be allocated to each of the cameras

• The camera, that adapts its quality to fit the frame to the allocated bandwidth

[20] described the camera adaptation using control theory and also showed the advantage of co-operation between the network manager and the cameras work to ensure optimal network utilisation [8]. Improves on the work by showing a need for triggering the manager only when required and the non-trivial decision involving the optimal selection of the triggering threshold, $\tau_{\text{thr}}$.

### 6.3. Time triggered manager

Figure 7.: Results with Equal Distribution, and adaptation with [10] (left plots) and PI controller (right plots) [20]

In this experiment, the manager allocates an equal amount of bandwidth to the two cameras in the network to evaluate camera adaptation. The network manager allocates bandwidth to the cameras periodically with game-theoretic bandwidth allocation.

Figure 7 shows the comparison between two adaptation schemes, one from [10] and the adaptive PI controller from previous work [20]. Only camera 1 is
present initially up to time 40 s, then camera 2 joins the network. Camera 2 leaves the network at around time 80 s.

The left plot shows the adaptation strategy from [10] with a frame size estimation model to fit image sizes to the allocated bandwidth. The right plot shows the camera with a tuned PI controller. The parameters of the camera controller are tuned empirically according to standard control practices [21]. The PI controller is more efficient at using the bandwidth allocated by the network manager compared to the model from [10]. The combination of time-triggered bandwidth allocation and camera adaptation using PI controller is efficient in bandwidth allocation and its utilisation by the cameras.

### 6.4. Event-triggered manager

**Figure 8**: Event-triggered activation with $\tau_{\text{thr}} \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and the optimal strategy synthesised by PRISM: bandwidth allocation ($b_1$ and $b_2$) and image quality ($q_1$ and $q_2$) [8].

The event-triggered manager is designed to be more efficient compared to the time-triggered manager. It is only triggered when the absolute error in the cameras exceeds $\tau_{\text{thr}}$.

**Figure 8** shows the event-triggered manager triggered at different thresholds. Both cameras are initialised in the network simultaneously at time 0.

![Event-triggered manager diagram](image-url)

**Figure 8** Event-triggered activation with thr 2 $\{0.1; 0.2; 0.3; 0.4; 0.5\}$ and the optimal strategy synthesised by PRISM: bandwidth allocation ($b_1$ and $b_2$) and image quality ($q_1$ and $q_2$) [1].
Camera 1 captures a scene that contains a large number of artefacts and thus requires a larger allocation of the bandwidth. Camera 2 captures a simple scene with fewer movements and artefacts requiring a comparatively smaller bandwidth allocation. The manager allocates a larger amount of bandwidth to camera 1 and lower amount of bandwidth to camera 2 as seen in the figure. Thus, selecting the value of $\tau_{thr}$ becomes a crucial design choice to optimise performance of the camera surveillance network.

### 6.5. Choice of triggering threshold, $\tau_{thr}$

The cost function chosen in the paper is defined as shown in Equation 6.5, where $\text{man\_int}$ is the number of manager interventions and $\text{drop1}$, $\text{drop2}$ and $\text{drop3}$ are the number of dropped frames of camera 1, 2 and 3, respectively.

$$cost = 1 \cdot \text{man\_int} + 10 \cdot \sum (\text{drop1} + \text{drop2} + \text{drop3})$$

Figure 9 shows the cost for the camera network with various triggering thresholds $\tau_{thr}$. The plot shows the results of 1-h-long experiments performed 10 times independently. The cost is large for both low and high values of $\tau_{thr}$. For low values of $\tau_{thr}$, the cost is high due to the continuous reconfiguration of the network accumulating costs for both manager interventions and dropped frames.

Lower values of $\tau_{thr}$ lead to frequent network reconfigurations by the network manager. Small disturbances in the camera scenes invoke manager interventions and lead to frame drops during network reconfigurations. Higher values of $\tau_{thr}$ increase costs and variance. There are fewer network reconfigurations even during large disturbances in scenes captured by the cameras. This places more emphasis on the cameras to regulate their bandwidth usage, causing large number of frame drops. For $\tau_{thr} = [0.3, 0.6]$, the cost is lower compared to other values of $\tau_{thr}$. These values indicate a good trade-off between high costs due to frequent interventions and high costs due to dropped frames.

Figure 9: Total cost box plot for 10 1-h-long executions traces

![Figure 9 Total cost box plot for 10 1-hour long executions traces](image-url)
Figure 10 shows the trend of number of manager interventions for increasing values of $\tau_{thr}$. It shows a 5 point smoothed average of the number of manager interventions. The number of manager interventions reduces as $\tau_{thr}$ increases as there are fewer network re-configurations and the manager is not invoked for small disturbances in the camera images. Higher values of $\tau_{thr}$ lead to fewer manager interventions with no network reconfigurations even during large changes in camera scenes.

Figure 11 shows a 5-point average of the number of dropped frames versus threshold $\tau_{thr}$. The number of dropped frames increases for higher values of $\tau_{thr}$ due to lower number of manager interventions placing emphasis on quicker camera adaptations. These results show that the tuning of the cost function is non-trivial and can be varied by choosing the appropriate weights depending upon the behaviour of the camera network.

7. Related work

Video transmission using self-adaptive cameras have been investigated in scenarios where images are transmitted over the internet or local area networks
with focus on image compression [22–27]. Research on video transmission has led to transmission protocols such as RTP, RTSP, SIP and improvements upon them which measure key network parameters, such as bandwidth usage, packet loss and round-trip delays. These help to cope with network load conditions, controlling the load submitted to the network [28] or using traffic prioritisation [29]. Research into image compression led to standards to reduce bandwidth utilisation like MJPEG, JPEG2000, MPEG-4, H.264, MPEG-H and H.265 that remove redundant information in sequence of frames.

Some research domains impose limitations on the acceptable delays in the system. For example, augmented reality [30], industrial supervised multimedia control [24], multimedia embedded systems [31], automated inspection [32] and vehicle navigation [33]. Video surveillance is similar. Image compression is frequently preferred to video compression. This is due to lower latency incurred and low memory and computing requirements during image compression compared to video compression. The variability that occurs during compression complicates fitting the compressed image to the instantaneous network conditions. This has motivated substantial research into adaptive techniques [24,25,31]. These works focus mainly on the adaptation of streams to what the network provides. Not on the scheduling on the network. A way of achieving this network reservation is through the use of channels. This is addressed in [10] which addressed the problem using adaptive network channels using a global network manager that tracks camera bandwidth usage. Using control theory for quality adaptation used in the paper is similar to [25]. Our work adds to the camera adaptation by implementing a network bandwidth distribution strategy. Also explored in this paper is the use of classical and statistical model checking to verify desirable properties.

Although better performance can be obtained using domain knowledge to optimise bandwidth allocation [26], this paper assumes no such knowledge. Also investigated is the usage of cameras that transmit images using wireless networks [27] which highlights the need for adaptation. In this case it increases the amount of processing at the node. Compared to the work, this paper also provides formal guarantees on the system obtained via model checking.

Model checking has been applied to analyse complex system in recent times. [34,35] show the application of model checking to smart grids using different model checking tools. Model checking has also been used for finding errors in cloud applications [36].

This paper uses PRISM [9] for the camera network analysis. Using the support available for probabilistic models and statistical model checking, PRISM has been used for verification of a wide-variety of systems. For e.g. randomised distributed algorithms [37], software with probabilistic transitions [38], designs for applications using nanotechnology [39], communication protocols [40], self-adaptive software [41], security [42] and anonymous protocols [43].
PMC has been applied to verify properties of a video streaming system in [6]. In this work, high-fidelity simulations have been abstracted into probabilistic higher-level models. PMC has also been used in verification of safety and timeliness properties in air traffic control systems [7]. Contrary to all the mentioned papers, this paper uses PMC to verify a class of properties that is typically related with control-theoretical guarantees.

8. Conclusion

Model checking has shown great promise in verification of complex systems. Model checking is more exhaustive and robust verification approach compared to traditional methods such as software testing. The drawback of model checking is the state space explosion problem and translation of relevant properties of the system into formal language. The work in this paper addresses both of these concerns.

We applied model checking to a camera surveillance network with two independent strategies for bandwidth adaptation. We built two models of the adaptive cameras in the network, a linear model from [8] that did not consider disturbances during image encoding, and a second model that incorporated disturbances using probabilistic transitions in the model. Application of classical model checking techniques allowed for more informed choice of control parameters and triggering thresholds for the camera models, with and without disturbances. The camera model consisting of probabilistic disturbances led to state-space explosion due to memory issues. We showed that using SMC, the state-space explosion can be mitigated by performing property verification of complex models at the expense of relaxed bounds. Although the obtained bounds are not as robust as classical model checking, we obtained useful information on system performance and parameter selection. Finally, we evaluated the effect of different triggering thresholds on the number of manager interventions and dropped frames.

Acknowledgments

The authors are members of the LCCC Linnaeus Center and the ELLIIT Strategic Research Area at Lund University. This work was supported by the Swedish Research Council through the project “Feedback Computing”, VR 621-2014-6256.

Disclosure statement

No potential conflict of interest was reported by the authors.
**Funding**

The authors are members of LCCC Linnaeus Center and the ELLIT Excellence Center at Lund University. This work was supported by the Swedish Research Council (VR) project “Feedback Computing” [VR 621-2014-6256].

**ORCID**

Gautham Nayak Seetanadi [http://orcid.org/0000-0001-6491-4344](http://orcid.org/0000-0001-6491-4344)

**References**

[1] Heo J, Abdelzaher T Adaptguard: guarding adaptive systems from instability. In: 6th ACM International Conference on Autonomic Computing, New York, NY, USA; 2009.

[2] Maggio M, Bini E, Chasparis G, et al. A game-theoretic resource manager for rt applications. In: 25th Euromicro Conference on Real-Time Systems, Los Alamitos, CA, USA; 2013.

[3] Edpalm V, Martins A, Maggio M, et al. H.264 video frame size estimation. Department of Automatic Control, Lund Institute of Technology, Lund University; 2018. Technical Reports TFRT-7654.

[4] Edpalm V, Martins A, Årzén KE, et al. Camera networks dimensioning and scheduling with quasi worst-case transmission time. In: 30th Euromicro Conference on Real-Time Systems (ECRTS 2018); (Leibniz International Proceedings in Informatics (LIPIcs); Vol. 106); Dagstuhl, Germany; 2018. p. 17: 1–17:22.

[5] Ding W, Liu B. Rate control of MPEG video coding and recording by rate-quantization modeling. IEEE Transactions on Circuits and Systems for Video Technology, 1996;6 (1):12–20.

[6] Nagaoka T, Ito A, Okano K, et al. QoS analysis of real-time distributed systems based on hybrid analysis of probabilistic model checking technique and simulation. Transactions on Information and Systems. 2011;E94.D(5).

[7] Hanh T, Van Hung D Verification of an air-traffic control system with probabilistic real-time model-checking. UNU-IIST United Nations University International Institute for Software Technology; 2007.

[8] Seetanadi GN, Camara J, Almeida L, et al. Event-driven bandwidth allocation with formal guarantees for camera networks. In: RTSS, Real-Time Systems Symposium, Paris, France; 2017.

[9] Kwiatkowska M, Norman G, Parker D. PRISM 4.0: verification of probabilistic real-time systems. In: Proc. 23rd International Conference on Computer Aided Verification (CAV’11); (LNCS; Vol. 6806). Berlin, Heidelberg: Springer; 2011. p. 585–591.

[10] Silvestre-Blanes J, Almeida L, Marau R, et al. Online QoS management for multimedia real-time transmission in industrial networks. IEEE Trans Ind Electron. 2011;58(3):1061–1071.

[11] Holzmann G. Spin model checker, the: primer and reference manual. 1st ed. Addison-Wesley Professional; 2003.

[12] Larsen KG, Pettersson P, Yi W Model-checking for real-time systems. In: Proc. of Fundamentals of Computation Theory; Aug, Berlin, Heidelberg; 1995. p. 62–88; (Lecture Notes in Computer Science; 965).
[13] Andova S, Hermanns H, Katoen JP. Discrete-time rewards model-checked. In: 1st International Workshop Formal Modeling and Analysis of Timed Systems, Marseille, France; 2003.
[14] Baier C, Katoen JP. Principles of model checking. The MIT Press; 2008.
[15] Knuth DE, DE K, AC Y. The complexity of nonuniform random number generation. 1976.
[16] Valmari A. The state explosion problem. Springer, Berlin, Heidelberg; 1998. p. 429–528.
[17] David A, Larsen KG, Legay A, et al. Uppaal SMC tutorial. International Journal on Software Tools for Technology Transfer. 2015 Aug;17(4):397–415.
[18] Cimatti A. Industrial applications of model checking. Berlin, Heidelberg: Springer Berlin Heidelberg; 2001.
[19] David A, Larsen KG, Legay A, et al. Time for statistical model checking of real-time systems. In: Computer Aided Verification; Berlin, Heidelberg. Springer Berlin Heidelberg; 2011. p. 349–355.
[20] Seetanadi GN, Oliveira L, Almeida L, et al. Game-theoretic network bandwidth distribution for self-adaptive cameras. In: 15th International Workshop on Real-Time Networks, Dubrovnik, Croatia; 2017.
[21] Åström KJ, Hägglund T. PID controllers: theory, design, and tuning. 2nd ed. NC: Instrument Society of America, Research Triangle Park; 1995.
[22] Vandalore B, Feng W, Jain R, et al. A survey of application layer techniques for adaptive streaming of multimedia. Real-Time Imaging. 2001;7(3):221–235.
[23] Communication A. White paper: digital video compression: review of the methodologies and standards to use for video transmission and storage. 2004.
[24] Rinner B, Wolf W An introduction to distributed smart cameras. Proceedings of the IEEE, 2008;96(10):1565–1575.
[25] Wang X, Chen M, Huang HM, et al. Control-based adaptive middleware for real-time image transmission over bandwidth-constrained networks. IEEE Trans Parallel Distrib Syst. 2008;19(6):779–793.
[26] Toka L, Lajtha A, Hosszu É, et al. A resource-aware and time-critical IoT framework. In: IEEE International Conference on Computer Communications INFOCOM, Atlanta, GA; May; 2017.
[27] Zhang T, Chowdhery A, Bahl PV, et al. The design and implementation of a wireless video surveillance system. In: Proceedings of the 21st Annual International Conference on Mobile Computing and Networking; New York, NY, USA. ACM; 2015. p. 426–438; MobiCom ’15. Available from: doi: 10.1145/2789168.2790123.
[28] Veeraraghavan V, Weber S. Fundamental tradeoffs in distributed algorithms for rate adaptive multimedia streams. Comput Netw. 2008;52(6):1238–1251.
[29] Cao D, Nguyen T, Nguyen L Improving the video transmission quality over IP network. In: 5th International Conference on Ubiquitous and Future Networks, Da Nang, Vietnam; 2013.
[30] Razavi R, Fleury M, Ghanbari M. Low-delay video control in a personal area network for augmented reality. IET Image Process. 2008;2(3):150.
[31] Ramos N, Panigrahi D, Dey S. Dynamic adaptation policies to improve quality of service of real-time multimedia applications in IEEE 802.11e WLAN networks. Wireless Networks. 2007;13(4):511–535.
[32] Kumar A. Computer-vision-based fabric defect detection: a survey. IEEE Trans Ind Electron. 2008;55(1):348–363.
[33] De Lima D, Victorino A, LeBlanc DJ. A hybrid controller for vision-based navigation of autonomous vehicles in urban environments. IEEE Trans Intell Transp Syst. 2016;17 (8):772–781.
[34] Hamman ST, Hopkinson KM, Fadul JE. A model checking approach to testing the reliability of smart grid protection systems. IEEE Trans Power Delivery. 2017 Dec;32(6):2408–2415.

[35] Yüksel E, Zhu H, Nielson HR, et al. Modelling and analysis of smart grid: a stochastic model checking case study. In: 2012 Sixth International Symposium on Theoretical Aspects of Software Engineering, Beijing, China; July; 2012. p. 25–32.

[36] Tian T, Zhang Y, Zhou Q, et al. Modelx: using model checking to find design errors of cloud applications. In: 2016 IEEE International Conference on Computer and Information Technology (CIT), Nadi, Fiji; Dec; 2016. p. 607–610.

[37] Kwiatkowska MZ, Norman G, Segala R. Automated verification of a randomized distributed consensus protocol using cadence smv and prism. In: Proceedings of the 13th International Conference on Computer Aided Verification; London, UK, UK; 2001. p. 194–206; CAV ’01.

[38] Kattenbelt M, Kwiatkowska M, Norman G, et al. Abstraction refinement for probabilistic software. In: Verification, Model Checking, and Abstract Interpretation; Berlin, Heidelberg; 2009. p. 182–197.

[39] Norman G, Parker D, Kwiatkowska M, et al. Evaluating the reliability of NAND multiplexing with prism. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2005 Oct;24(10):1629–1637.

[40] Duflot M, Kwiatkowska M, Norman G, et al. A formal analysis of Bluetooth device discovery. International Journal on Software Tools for Technology Transfer. 2006 Nov;8(6):621–632.

[41] Calinescu R, Ghezzi C, Kwiatkowska M, et al. Self-adaptive software needs quantitative verification at runtime. Commun ACM. 2012 Sep;55(9):69–77.

[42] Basagiannis S, Katsaros P, Pombortsis A, et al. Probabilistic model checking for the quantification of dos security threats. Comput Sec. 2009;28(6):450–465.

[43] Shmatikov V. Probabilistic analysis of anonymity. In: Proceedings 15th IEEE Computer Security Foundations Workshop, Cape Breton, Nova Scotia, Canada. CSFW-15; June; 2002. p. 119–128.