Cross layer control for bounded shared state inconsistency in wireless IoT devices

Tärneberg, William; Karaca, Mehmet; Robertsson, Anders; Kihl, Maria

Published in:
Conference on Decision and Control

DOI:
10.1109/CDC.2017.8264240

2017

Document Version:
Publisher's PDF, also known as Version of record

Cross layer control for bounded shared state inconsistency in wireless IoT devices

William Tärneberg[1], Mehmet Karaca[1], Anders Robertsson[2], Maria Kihl[1]
[1]Department of Electrical and Information Technology, Lund University, Sweden
[2]Department of Automatic Control, Lund University, Sweden

***Abstract*— The devices that constitute the Internet of Things are evolving to include more than just enabling sensing and actuation over a wireless interface. In a contemporary scenario, these devices perform tasks and contribute to an aggregate information flow, in a distributed manner. In the wake of this evolution, new distributed Internet of Things frameworks have emerged. These frameworks maintain a distributed shared state in a distributed hash table. An Internet of Things system's ability to make decisions autonomously and distributively depend on the level of consistency of the shared state. As wireless resources are scarce, the amount of deferred state information in each device is unknown when the system is highly utilised. In this paper, we have developed a controller with the objective to achieve a bounded time-average of deferred state information in each device while maintaining system stability. The controller is derived using Lyapunov drift optimisation with penalty. The resulting controller can successfully bound the shared state consistency level within a narrow margin, maintain system stability, and balance the traffic flow trade-off more successfully than comparable and conventionally used methods.**

I. Introduction

As the Internet of Things (IoT) matures and enters new fields of operation, the notion of independent wireless sensors and actuators is becoming obsolete. Contemporary IoT applications are more focused on data generation and distributed decision making than simply connecting mundane household things for the sake of programmability and remote actuation. Forthcoming IoT applications and services will instead operate across multiple devices to, for example, accomplish a one-time task, operate a continuous process, or collect data. These applications will be composed and provisioned dynamically at runtime in a distributed manner to meet real-time demands. Additionally, IoT devices and their capabilities are discovered and managed in a manner reminiscent of micro services from the cloud computing domain, see Figure 1.

Emerging platforms such as Ericsson's Calvin [1], will proposedly enable and orchestrate these types of IoT applications and systems of IoT devices. To do so, these platforms rely on a shared distributed state realised using a Distributed Hash Table (DHT) [2]. The DHT contains the capabilities and state of the system's devices and hosted applications. The information shared through the DHT allow devices to autonomously broker resources. This approach removes the need for centralised control and introduces some degree of platform fault tolerance. This also allow services and server-less functions to seamlessly migrate between IoT devices and cloud resources. Nevertheless, all autonomous

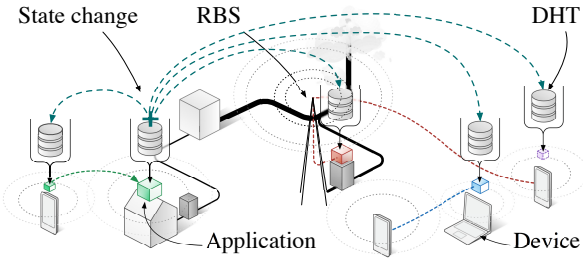

Fig. 1. Forthcoming IoT application paradigm.

distributed management decisions in such a system rely on a consistent DHT. Although a DHT is inherently fault tolerant, fine grained and mission critical decisions require near complete knowledge of the state's consistency. The expected consistency of the DHT is thus linked to the operational efficiency of the system and its applications.

An IoT device, being it a thermometer or a multi-purpose robot, quintessentially connects to the world beyond, wirelessly. Devices that have sparse and intermittent power supplies or are entirely self-reliant must be frugal with how they use the wireless medium. Emerging 5th Generation wireless systems (5G) concepts are embracing these challenges faced by IoT devices [3]. Proposed 5G standards consider device limitations and intermittency, multiple medium access methods [4], Device-to-Device (D2D) communication, and new radio-access technologies [5]. The propositions of 5G are more about IoT and enabling mission critical applications than higher throughput.

In IoT devices, as applications and internal processes serve requests they generate both egress application data and updates bound for the DHTs. From now on we refer to the former as *application traffic* and the latter as *state traffic*. Although significant to the consistency of the system as a whole, the state traffic is not as latency sensitive as the application traffic.

Application traffic and state traffic contend for the device's wireless resources. The amount of state traffic versus application traffic produced by a service request is stochastic, as is the rate of service requests. Consequently, the state traffic will intrude on the application traffic. The trade-off between application traffic and state traffic is non-trivial. For example, naïvely prioritising the application traffic or capping the state traffic can disrupt the state traffic entirely. Doing so will ensure neither a favourable trade-off between the two traffic flows nor that the system will be stable. As the traffic and the wireless link connecting the device is stochastic and vary

greatly over time, no absolute bounds on either traffic flow can be trivially set that accommodates both traffic flows. It is evident that maintaining a consistent DHT comes at a significant performance cost for the applications that the system is intended for. We therefore need to accept some level of DHT inconsistency. However, uncertainty in the DHT's inconsistency has a detrimental effect on the quality of the system's distributed decision-making.

In this paper, we propose a Cross Layer Controller that bounds DHT inconsistency uncertainty by bounding the time-average amount of deferred state traffic using a by using a virtual queue technique [6]. The proposed method is expressed as a jointly-formulated flow control and scheduling decisions derived using stochastic Lyapunov drift optimisation techniques found in [6], [7]. The proposed controller is thus able to accommodate the stochastic nature of the two traffic flows and ensure a bounded amount of deferred state information while maintaining queue and system stability. More precisely, the proposed controller schedules the two traffic flows so that the expected deferred state traffic is bounded at a predefined level without explicitly violating the application traffic. Intuitively, the deferred state traffic represents the amount of state information that would be lost if the device fails, or the level inconsistency of the of distributed state. This level can be dynamically adjusted to reflect each devices' individual probability of failure, or a system-wide tolerated level of distributed state inconsistency. The resulting flow control decision is implemented and operated independently in each device while the scheduling decision is implemented centrally in the wireless infrastructure.

The authors of [8] stabilise a network with finite queues. This is not adequate for the problem presented in this paper as it needs to accommodate a variable target consistency while adequately accommodating the application traffic. In doing so one must be able to temporarily violate the target consistency to maintain a high throughput. Lyapunov drift analysis has been used extensively when analysing the performance and stability of mesh networks, such as [9]. Although related in terms of the characteristics of the devices, these works do not share the same objectives nor do they take into account the consistency of a shared data source, such as a DHT. The work [10] taxamonise methods for achieving consistent DHT-base routing algorithms in mesh networks. Although such methods are related they cannot be applied at the scale and rate of change of IoT systems. Additionally, in our previous work [11] we explored a similar trade-off but without explicit consistency objectives using Model Predictive Control (MPC).

Through simulation, we show that the proposed controller is able to bound the time-average shared state inconsistency within a narrow margin of the desired level while keeping the system stable. It is also shown that the controller is also able to do so while accommodating changes in load and wireless channel conditions. Additionally, the proposed controller is also able to more successfully balance the system's two traffic flows than comparable and conventionally used methods.

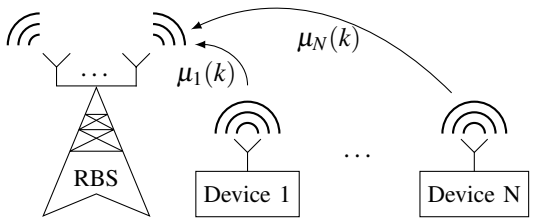

Fig. 2. Infrastructure model.

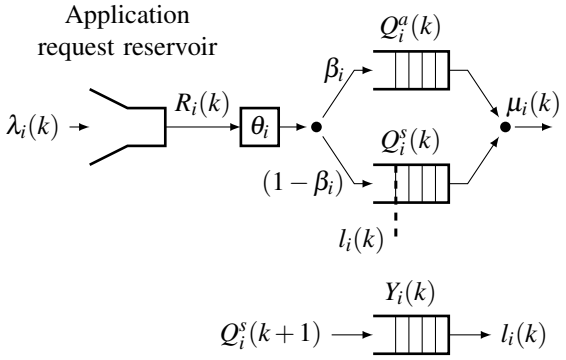

Fig. 3. Device model with virtual queue $Y_i(t)$.

II. System model

In this section, we define our system model for a device and the system's infrastructure. As depicted in Figure 1 the target system consists of N devices and a Radio Base Station (RBS). To accommodate the traffic flows' individual objectives, each device operates two egress traffic queues, one for each traffic class; application traffic Q_i^a and state traffic Q_i^s. The state queue has a variable time-average limit $l_i(k)$. The limit $l_i(k)$, constitutes the bound on the defered amount of state data. Both egress queues share the same sink, the wireless interface. All devices communicate over the wireless interface provided by the RBS. Each device has a time varying channel capacity of $\mu_i(k)$ in the time interval $[k,k+1]$. $\mu_i(k)$ has a particular probability distribution. Access to the medium is scheduled by the RBS at each time instance. For each device i, the capacity $\mu_i(k)$ can be shared between the application and state queues. Let $\mu_i^a(k)$ and $\mu_i^s(k)$ be the capacity allocated to the application and state queues for device i at time period k, respectively and $\mu_i(k) = \mu_i^a(k) + \mu_i^s(k)$. The targeted system as a whole is depicted in Figure 2. An overview of the device model is illustrated in Figure 3.

Each device is subject to ingress application requests at the rate of $\lambda_i(k)$. The application requests are deposited in a FIFO reservoir and processed at a controllable rate $R_i(k)$. As a request leaves the reservoir, the device commits to processing the request, which subsequently results in a change to the DHT. Specifically, $R_i(k)$ is the number of requests committed at time k, and θ_i is the mean of the size of each request. β_i is the fraction of $R_i(k)$ that is application traffic while $(1-\beta_i)$ of $R_i(k)$ is state traffic $\beta_i, \theta_i, \lambda_i(k)$ are constant for a device, but can be different among the devices, and are unique for each application i.

III. Cross Layer Controller

In this section, we detail the premise of the target problem and formalise the proposed Cross Layer Controller. From here on, we refer to the controller as simply the controller. The controller is formulated using the Lyapunov drift-plus-penalty Theorem presented in [6]. In the resulting formulation, the controller has two control decisions; flow control and user (IoT device) scheduling. The two decisions are jointly formulated but act independently.

Contrary to the predictive approach in [11], this approach acts only on the system's current state in each time instance. This allows the controller to accommodate the stochastic nature of the system's queuing dynamics. In addition, the proposed controller can handle stochastic wireless channel conditions. The time-average horizon objective is achieved by acting on the relative changes in the system's queue sizes, i.e., the Lyapunov drift.

A. Objective

The objective of the controller is to maximise the utility of the application queue while ensuring its stability, i.e., $E[Q_i^a(k)] < \infty$ and bound the time-average occupancy of the state queue, i.e., $E[Q_i^s(k)] \leq l_i$. In a practical sense, the utility achieved by an IoT device is defined as the amount of requests (average $R_i(k)$) that is to be handled by that IoT device. The achievable utility depends on the designed flow controller, scheduling decisions, and the network capacity. We define a strictly increasing utility function for the application queue i, $\forall i$ as a function of output rate of the application request reservoir $R_i(k)$, as follows: let r_i be the time-average utility of the application queue i,

$$r_i = \lim_{k\to\infty} \frac{1}{k} \sum_{\tau=0}^{k-1} E[\beta_i R_i(\tau)] \tag{1}$$

The objective of the controller is formally expressed as,

$$\max \quad \sum_{i=1}^{N} g(r_i) \tag{2}$$

$$s.t. \quad E[Q_i^s(k)] \leq l_i \qquad \forall i \tag{3}$$

$$E[Q_i^a(k)] < \infty \qquad \forall i \tag{4}$$

We note that imposing the constraints in Equation (3) make the solution of our optimisation problem complicated and the problem is actually a complex Markov Decision Problem (MDP). It is well-known that MDP suffers from the curse of dimensionality since the number of queue state vectors grows geometrically with the number of IoT devices. Therefore, it is not feasible to find a suitable algorithm in practice. In this paper, instead of finding an optimal solution to the problem Equations (2) to (4), we develop a much simpler sub-optimal controller which is based on Lyapunov drift technique [6].

B. Queuing dynamics

Let us first define the system's queue dynamics. The application and state queue's dynamics of IoT device i are as follows,

$$Q_i^a(k+1) = \max\{Q_i^a(k) - \mu_i^a(k), 0\} + R_i(k)\theta_i\beta_i$$
$$Q_i^s(k+1) = \max\{Q_i^s(k) - \mu_i^s(k), 0\} + R_i(k)\theta_i(1-\beta_i)$$

To satisfy the constraint in 3 we define a virtual queue (see Figure 2) with the following dynamics,

$$\begin{aligned} Y_i(k+1) &= \max\{Y_i(k) - l_i(k), 0\} + Q_i^s(k+1) \\ &= \max\{Y_i(k) - l_i(k), 0\} + \max\{Q_i^s(k) - \mu_i^s(k), 0\} \\ &\quad + R_i(k)\theta_i(1-\beta_i) \end{aligned} \tag{5}$$

The virtual queue $Y_i(k)$ for application i in Equation (5) accumulates that application's amount of deferred state data in $Q_i^s(k)$ and deducts its bound on the deferred amount of state data $l_i(k)$. Consequently, as long as the virtual queue $Y_i(k)$ is stabilised then the constraint in Equation (3) will be satisfied. This can be achieved by using Lyapunov Optimisation Drift, as we show next.

C. Lyapunov drift

In this paper we employ a quadratic Lyapunov function expressed as,

$$L(k) = \frac{1}{2} \sum_{i=1}^{N} Q_i^a(k)^2 + Y_i(k)^2$$

This Lyapunov function is the scalar measure of the total of both actual and virtual queues in the network. The Lyapunov drift is given by,

$$\Delta L(k) = E[L(k+1) - L(k) \mid \bar{Q}] \tag{6}$$

Where $\bar{Q} = \{(Q_1^a(k), Y_1(k)), \ldots, (Q_N^a(k), Y_N(k))\}$. It is a well-established result [6] that minimising the following function,

$$\min \Delta L(k) - \sum_{i=1}^{N} VE[g(R_i(k))] \tag{7}$$

minimises Equation (2). Clearly, it enables us to have a simple multi-objective optimisation problem where the first term (the Lyapunov drift) satisfies the constraints in Equations (3) and (4), whereas the second term ($\sum_{i=1}^{N} VE[g(R_i(k))]$) maximises the total network utility as expressed in Equation (2). The drift-plus-penalty theorem in [6] does not require strict convexity assumptions. However, it ensures that the expected of the optimisation problem's primals converge to a solution that is within a factor of optimality, with bounded queue sizes. Furthermore, V is a system parameter which allows us to make a trade-off between the achieved utility and the average queue backlog in the queues, as applied in [6], [7]. Next, we design a controller which minimises Equation (7) at each period k.

D. Controller design

Since we do not have an explicit close-form expression for Equation (7), in order to formulate a controller that minimises Equation (7) in discrete time, we will first have to find the bound for Equation (6). We proceed with the following in mind: $\max\{x,0\}^2 \leq x^2$ and $\max\{x,0\} \leq x^2$ to express,

$$
\begin{aligned}
Q_i^a(k+1)^2 \leq & Q_i^a(k)^2 + \mu_i^a(k)^2 + (R_i(k)\beta_i\theta_i)^2 \\
& - 2Q_i^a(k)(\mu_i^a(k) - R_i(k)\beta_i\theta_i), \\
Y_i(k+1)^2 \leq & (Y_i(k) - l_i(k))^2 + Q_i^s(k)^2 + \mu_i^s(k)^2 \\
& + 2Y_i(k)(Q_i^s(k) + R_i(k)(1-\beta_i)\theta_i) \\
& - 2Q_i^s(k)(\mu_i^s(k) - R_i(k)(1-\beta_i)\theta_i) \\
& + (R_i(k)(1-\beta_i)\theta_i))^2
\end{aligned}
$$

We note that the network capacity at a time period k is always upper-bounded due to practical limitations (e.g., transmit power). That is to say, $\mu_i^a(k) \leq \mu_{\max}$, $\mu_i^s(k) \leq \mu_{\max}$ for all i and k. Also, the amount of requests that can be admitted is upper-bounded, i.e., $R_i(k) \leq R_{\max}$, for all i and k. $\mu_{\max}$ and $R_{\max}$ are constants. The choice of $R_i^{\max}$ is independent of the stability of the system. However, a choice of $R_i^{\max}$ less than the allocated aggregate mean channel capacity results in a strictly underutilised system. Then we have,

$$
\begin{aligned}
& Q_i^a(k+1)^2 - Q_i^a(k)^2 \\
& \qquad \leq B^a - 2Q_i^a(k)(\mu_i^a(k) - R_i^a(k)\beta_i\theta_i)
\end{aligned} \tag{8}
$$

where $B^a = \mu_{\max}^2 + (R_{\max}\beta_i\theta_i)^2$.Similarly, we find a bound for the virtual state queue $Y_i(k)$ as follows,

$$
\begin{aligned}
Y_i(k+1)^2 - Y_i(k)^2 \leq & \; l_i(k) - 2Y_i(k)l_i + Q_i^s(k)^2 \\
& + 2Y_i(k)(Q_i^s(k) + R_i(k)(1-\beta_i)\theta_i) \\
& + (R_i(k)(1-\beta_i)\theta_1)^2 + \mu_i(k)^2 \\
& - 2Q_i^s(k)[\mu_i(k) - R_i(k)(1-\beta_i)\theta_i] \\
\leq & \; B^Y + Q_i^s(k)^2 \\
& - 2Y_i(k)(l_i - Q_i^s(k) - R_i(k)(1-\beta_i)\theta_i) \\
& - 2Q_i^s(k)(\mu_i^s(k) - R_i(k)(1-\beta_i)\theta_i)
\end{aligned} \tag{9}
$$

Where $B^Y = l_{\max}^2 + \mu_{\max}^2 + (R_{\max}(1-\beta_i))^2$ and $l_{\max}(k) = \max\{l_i(k)\}$. By using Equations (8) and (9), Equation (7) can now be expressed in its entirety,

$$
\begin{aligned}
& \Delta L(k) - \sum_{i=1}^{N} VE[g(R_i(k))] \\
& \leq \sum_{i=1}^{N} B^a - 2E[Q_i^a(k) \mid \bar{Q}](\mu_i^a(k) - R_i^a(k)\beta_i\theta_i) \\
& \quad + B^Y - 2E[Y_i(k)(l_i(k) - Q_i^s(k) - R_i(k)(1-\beta_i)\theta_i) \mid \bar{Q}] \\
& \quad - 2E[Q_i^s(k) \mid \bar{Q}](\mu_i^s(k) - R_i(k)(1-\beta_i)\theta_i) \\
& \quad + E[Q_i^s(k)^2 \mid \bar{Q}] - VE[g(R_i(k))]
\end{aligned} \tag{10}
$$

Minimising the right hand side of the inequality in Equation (10) ensures queue stability and maximises our objective in Equation (2). Next, we provide our controller, which jointly minimises the right hand side of Equation (10) at each k.

1) Flow control: The flow control decision's objectives is to regulate the flow $R_i(k)$ from the application request reservoir. We proceed by collecting the expressions from Equation (10) that include $R_i(k)$ which is given as,

$$
\begin{aligned}
& \sum_{i=1}^{N} E[2Y_i(k)R_i(k)(1-\beta_i)\theta_i + 2Q_i^s(k)R_i(k)(1-\beta_i)\theta_i \\
& \quad + 2Q_i^a(k)R_i(k)\beta_i\theta_i - Vg(R_i(k))]
\end{aligned}
$$

Minimising the above expression for each device i at each k will minimise the right-hand side of Equation (10). Each device i solves the following problem, which will determine the flow decision for that device at time k,

$$
\begin{aligned}
\arg\min_{R_i} \quad & R_i[Q_i^s(k)(1-\beta_i)\theta_i + Q_i^a(k)\beta_i\theta_i \\
& + Y_i(k)(1-\beta_i)\theta_i] - V_i g(R_i) \\
s.t. \quad & R_i \leq R_i^{\max}
\end{aligned}
$$

The choice of $R_i^{\max}$ is independent of the stability of the system. V is a parameter that determines the balance between the controller's desire to back pressure requests and maximise its utility through R_i. Any positive V will result in a stable system.

Furthermore, the flow control is at its most effective when it keeps up with the system's rate of change, $r_{\text{flow}}(k) = \min\{\lambda_i(k), \mu_i^a(k)\mu_i^s(k)/(\mu_i^a(k) + \mu_i^s(k))\}$. However, a lower rate does not violate the stability criterion.

2) Scheduling: The other control decision in Equation (10) is the scheduling decision (i.e., determination of $\mu_i^a(k)$ and $\mu_i^s(k)$). The scheduling decision is centralised to the RBS. The idea is that $\mu_i^a(k)$ and $\mu_i^s(k)$ are determined in a way that the right hand-side of Equation (10) is minimised. Grouping and maximising in terms of $\mu_i^a(k)$ and $\mu_i^s(k)$ from Equation (10) yields,

$$
\arg\max_{i} \quad \max\{Q_i^a(k)\mu_i^a(k), Q_i^s(k)\mu_i^s(k)\} \tag{11}
$$

This essentially implies that the largest queue amongst all the devices will receive the channel capacity at that time k.

E. Parameter estimation

The distribution of the parameters θ_i and β_i are unknown to the system at runtime. The mean of the parameters are estimated using stochastic gradient descent, $\theta_i^{k+1} \leftarrow \theta_i^k - \gamma_i \nabla F(\theta_i^k)$, where $\gamma_i = \gamma \quad \forall i$.

IV. Evaluation

In this section, we motivate and taxonomise the evaluation scenarios for the proposed Cross Layer Controller. To evaluate the controller, a simulator was developed in Python using SimPy [12] for queue representations and the simulation core. CVXPY [13] is used for solving convex optimisation problems. For the sake of reproducibility, the simulator and the simulation below have been made available at [14].

The simulations presented below are designed to reveal how the proposed controller can maintain the maximum time-average deferred state information, converge, and handle variations in inputs.

A. Comparison policies

To evaluate the proposed controller we introduce three contrasting scheduling policies.

Round Robin (RR) scheduling In RR scheduling, the queues are scheduled in a sequential order regardless of

their occupancy, i.e., this approach does not enforce an explicit scheduling objective. RR is a commonly used scheduling approach.

Prioritised scheduling In prioritised scheduling, the queues in the system are partitioned into different prioritisation sets. The set of queues in a prioritisation level are never scheduled unless the queues in the higher prioritisation levels are empty. In this scenario the $Q_i^a(k)$ is prioritised over $Q_i^s(k)$ to ensure minimum application latency.

Priority Threshold (PT) scheduling In PT scheduling the queues in the system are partitioned into two sets. One of the sets has an occupancy threshold. A queue in that set is only scheduled if its occupancy exceeds the threshold. In the simulations below, the set of $Q_i^s(k)$ are assigned a threshold of $l_i(k)$.

B. Metrics

The metrics below are used to evaluate the proposed controller.

Utilisation ρ The utility level of a system of queues is expressed as $\rho = \lambda/\mu$. In the simulations below we observe two utilisation factors; end-to-end utilisation for device i $\rho_i^{\text{e2e}}(k)$ and the resulting level of utilisation due to back pressure from the flow control decision $\rho_i^{\text{flow}}(k)$. We also define $\bar{\rho}^{\text{e2e}}(k)$ as the time designed time-average utilization of a device as determined by the load and system parameters in Table I. Furthermore, $\hat{\rho}^{\text{e2e}}(k)$ and $\hat{\rho}^{\text{flow}}(k)$ are the instantaneous time-averages for system and post-controller utilisation levels, respectively, over all N devices. If the system is stable, $E[\rho_i^{\text{flow}}] \leq 1$. Consequently, $\bar{\rho}^{\text{e2e}}(k) \geq 1$ is a system loaded over 100%.

Utility The utility of the system is proportional to R.

Trade-off The trade-off between the competing traffic flows is at the core of the problem addressed in this paper. We formally express this trade-off as,

$$\alpha(k) = \begin{cases} \frac{Q_i^s(k) - l_i(k)}{l_i(k)} & \text{if} \quad \rho_i^{\text{e2e}}(k) \geq 1 \\ |\Delta Q_i^s(k) - \Delta Q_i^a(k)| & \text{if} \quad \rho_i^{\text{e2e}}(k) < 1 \end{cases}$$

Where Δ is the forward difference of $f(x)$, $\Delta f(x) = f(x+1) - f(x)$. The first component premiers stability and a bounded $Q_i^s(k)$ when the system is overutilised. The second component premiers that $Q_i^a(k)$ and $Q_i^s(k)$ change at a similar rate when the system is underutilised. A low value is desired.

We also define $\hat{\lambda}(k), \hat{\mu}(k), \hat{Q}^a(k), \hat{Q}^s(k)$, and $\hat{R}(k)$ as the instantaneous mean for those values over all N devices.

C. System parameter values

The simulation parameters used to evaluate the proposed controller are presented in Table I. For this scenario we assume a Poisson arrival process and exponential service times. In the simulation scenario below, the total system utilisation $\bar{\rho}^{\text{e2e}}(k)$ is intermittently over 100%, $\bar{\rho}^{\text{e2e}}(k) > 1$, see Figure 4. Furthermore, for the policies detailed in Section IV-A, $R_i(k) = \lambda_i(k) \quad \forall i$.

Parameter	Value
Sim. duration	500
N	10
$\lambda_i(k)$	Poisson process, $exp(\lambda = 0.24) \quad \forall i$, see Figure 4
$\mu_i(k)$	See Figure 4
r_{flow}	0.01
R_i^{Max}	0.05 $\quad \forall i$
θ_i	$\mathcal{N}(\mu = 3, \sigma^2 = 3/2) \quad \forall i$, sampled for each arrival
β_i	$\mathcal{U}(0.35, 0.65)$, sampled once
$l_i(k)$	See Figure 4
γ_i	0.01 $\quad \forall i$
V	16
$g_i(R)$	$g_i(R) = \ln \beta R_i(k)$

TABLE I

SIMULATION PARAMETER VALUES OF THE PROPOSED CONTROLLER.

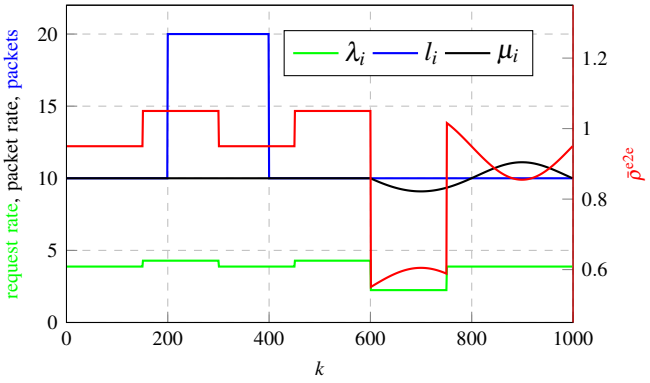

Fig. 4. System parameters and designed utilisation.

D. Input values

When evaluating the proposed Cross Layer Controller, we are particularly interested in the instances when the system is overutilised. We therefore balance the system inputs to achieve a desired level of system utilisation as presented by $\bar{\rho}^{\text{e2e}}(k)$ in Figure 4. This is achieved by either varying $\lambda_i(k)$ or $\mu_i(k)$. The system utilisation depicted in Figure 4 has three different load scenarios recurring over five time periods. The three load scenarios are; strictly underutilised, strictly overutilised, and transient. They are distributed over five periods, as demarked in Figure 4.

After the epoch of intermittent overutilisation $t > 600$, system load enters the transient epoch where the designed utilisation drops to 55%, $\bar{\rho}^{\text{e2e}}(k) = 0.55$. Up until this point the change in $\bar{\rho}^{\text{e2e}}(k)$ was attributed change in $\lambda_i(k)$. After this point $t > 600$, $\lambda_i(k)$ is kept constant and $\mu_i(k)$ is varied as a sinusoid, see Figure 4. From now on we refer to the intermittent overutilisation and transit epochs as epoch 1 and epoch 2, respectively.

V. RESULTS

In this section we present the results from the simulations detailed in Section IV. Note that the time scale in the simulations is unit-less. We also note that the estimations for σ_i and β_i have converged after 100 time units.

A. Expected deferred state traffic

The design objective of the controller is $E[Q_i^s] \leq l_i$ for each device i. Throughout epoch 1, as the system is overutilised, the objective of the controller is to keep $E[Q_i^s] = l_i$. Figure 5 shows that the controller is able to maintain a time-average within the 95th percentile $< 1\%$ of $l_i(k)$.

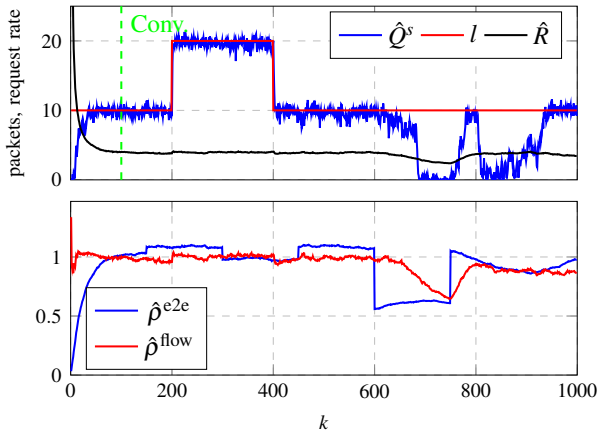

Fig. 5. Mean state queue occupancy $\hat{Q}^s$.

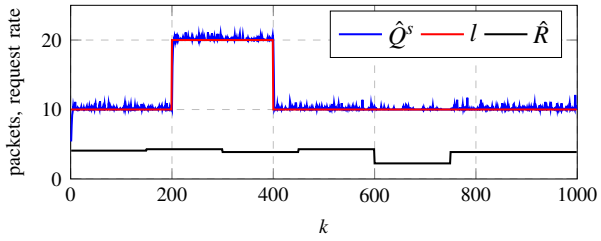

Fig. 6. $\hat{Q}^s(k)$ and $l_i(k)$ for PT scheduling.

$l_i(k)$ is increased to 20 at $k = 200$ and then decreased back to 10 at $k = 400$. We note that $\hat{Q}^s(k)$ tracks the target $l_i(k)$ practically immediately. The quick adjustment is achieved by momentarily sacrificing instantaneous stability $\hat{\rho}^{\text{flow}}(k) > 1$. This is practically instantly compensated for by a proportional reduction in $R_i(k)$. The expected stability of the system is thus maintained.

At the beginning of epoch 2, $\hat{\rho}^{\text{flow}}(k)$ lags $\hat{\rho}^{\text{e2e}}(k)$ and $\bar{\rho}^{\text{e2e}}(k)$ as the controller accommodates accumulated deferred application requests. Between $k = 600$ and just before $k = 680$ the controller gradually balances the application traffic and state traffic to when $\hat{\rho}^{\text{flow}}(k) < 1$. After this point there is no contention between the two traffic flows and $\hat{Q}^s$ drops to just above 0, see Figure 5.

In epoch 2, $\lambda_i(k)$ and $l_i(k)$ are kept constant and $\mu_i(k)$ is oscillated sinusoidally around $\pm 10\%$ of $\bar{\rho}^{\text{e2e}}(k) = 1$. The scheduling component of the controller will be first to act on the change in $\mu_i(k)$. Consequently, there is small lag in $\hat{R}(k)$ from the change in $\hat{\rho}^{\text{e2e}}(k)$, see Figure 5. However, this lag has evidently no practical impact on the controller's ability to track $l_i(k)$ in relation to $\hat{\rho}^{\text{e2e}}(k)$.

Furthermore, PT scheduling has the objective $E[Q_i^s] = l_i$. As Figure 6 shows, PT scheduling is able to meet this objective while the system is overutilised. However, contrary to the proposed controller, as the system moves into phase 2, $\hat{Q}^s(k)$ does not diminish. This is because none of the state queues will be served as long as there are any non-empty application queues $\sum_{i=1}^{N} Q_i^a(k) > 0$ or $Q_i^s(k) \geq l_i \forall i$ in the system. This is an undesirable trade-off for the targeted system as we do wish to reduce the amount of deferred state information when possible.

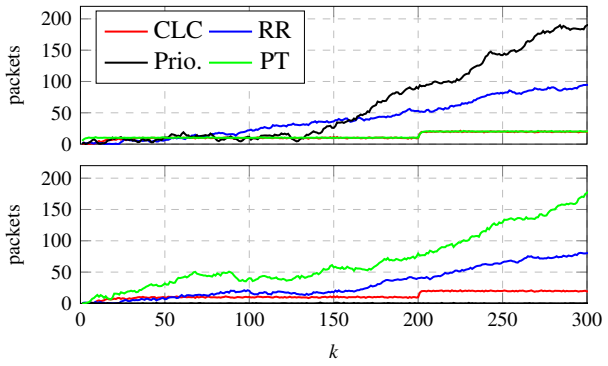

Fig. 7. Upper; $\hat{Q}^s(k)$, Lower; $\hat{Q}^a(k)$.

B. Stability and system utility

Another objective of the controller is to maintain a stable time-average system of queues, namely $E[Q_i^s] < +\infty$ and $E[Q_i^a] < +\infty$ for each device i. The system's stability is revealed by observing the growth of the queues $\hat{Q^s}(k)$ and $\hat{Q^a}(k)$ as well as the system utilisation after back pressure is applied, $\hat{\rho}^{\text{flow}}(k)$. As illustrated by Figure 7, both conditions: $E[Q_i^s] < +\infty$ and $E[Q_i^a] < +\infty$ are met using the proposed controller.

In Figure 5, note that $\hat{\rho}^{\text{flow}}(k)$ persist a level of around 1 even though $\hat{\rho}^{\text{e2e}}(k) \geq 1$. This is because the controlled system administers deferred application requests that have been back pressured. This is as designed and means that the system of N devices, as well as all application queues and service queues are stable.

Furthermore, the back pressure effect can be directly observed through $\hat{R}(k)$ in Figure 5. Note the inverse relationship between $\hat{R}(k)$ and $\hat{\rho}^{\text{flow}}(k)$ in Figure 7.

Neither of the methods; RR, PT, nor Prio. have stability as an objective. This is made evident by their growing queue sizes while $\hat{\rho}^{\text{e2e}}(k) \geq 1$ in Figure 7. In the case of RR scheduling, as it does not discriminate between two traffic flows, they grow with a similar gradient. Prioritised scheduling on the other hand prioritizes Q^a and thus lets Q^s grow uncontrollably. PT scheduling inadvertently maintains a stable Q^s at the expense of Q^a stability.

Because the proposed controller practices back pressure, it will analogously achieve a lower utility than the other scheduling methods as R_i is in this case suppressed. RR, PT, and Prio, will on the other hand persist $R_i(k) = \min\{R_{\max}, \lambda_i(k)\}$. This difference is contrasted in Figures 5 and 6 for the controller and the other methods respectively. The difference is small, but operating with a utility beyond the stability point has a detrimental effect on the latency experienced by each serviced application request.

C. Choice of V

As discussed in [6], [7], although any positive V will result in a stable system, the choice of V does have an impact on the controller's ability to meet its other objectives. A large V will promote the utility function, while a small V will promote back pressure. This is reflected in the immediate value of R at each k. However, all values of V will achieve the same time-average R, see Figure 8.

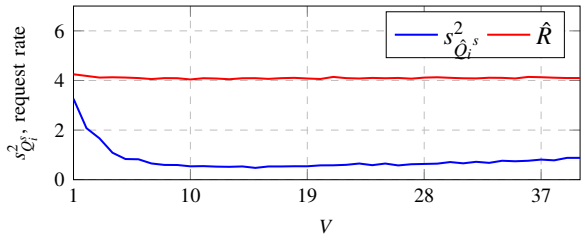

Fig. 8. Variance of $Q_i^s(k)$, $s^2_{Q_i^s}$ and $R_i(k)$ as a function of V.

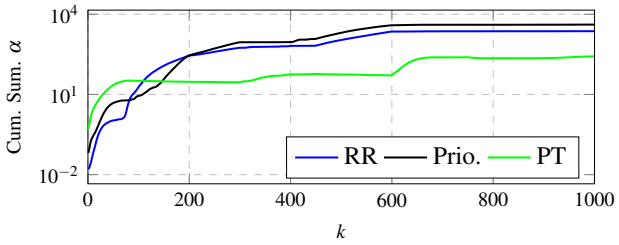

Fig. 9. Cumulative difference in trade-off to the controller.

Furthermore, as illustrated by Figure 8, the choice of V has an impact on the sample variance of Q_i^s, $s^2_{Q_i^s}$. Although the time-average converges at $V > 10$, it is desirable to minimise $s^2_{Q_i^s}(k)$ provided it is not at the expense of system utility $R_i(k)$. We can also note that R is not affected by V. As illustrated in Figure 8, $s^2_{Q_i^s}(k)$ is convex to V. For this particular system, minimum $s^2_{Q_i^s}(k)$ is at $V = 16$.

D. Quantifying the trade-off

We have now established that the controller successfully bounds the size of $Q_i^s(k)$ to $l_i(k)$ when the system is overutilised $\bar{\rho}^{\text{e2e}}(k) \geq 1$ and that the controller is able to stabilise the system. The trade-off metric formulated in Section IV-B attempts to summarise the performance of the controller in comparison to the other scheduling methods presented in Section IV-A over time.

In terms of the metric Section IV-B, the controller performs strictly better than the comparison scheduling methods. Therefore, for increased contrast, Figure 9 shows the cumulative trade-off for each method subtracted by the cumulative trade-off for the controller. PT scheduling tracks the proposed controller well but fails to balance the two flows as $\hat{\rho}^{\text{flow}}(k) < 1$ in the underutilised epoch. Prioritised scheduling allocates, on average, no resources to Q^s which therefore always maintains a high occupancy. RR scheduling indiscriminately schedules Q^a and Q^s and thus achieves a middle-of-the-road value during overutilisation.

Also worth noting is that

VI. Conclusions

In this paper, we have presented a Cross Layer Controller for achieving both a predictable maximum time average shared state inconsistency and system stability. The proposed controller and system objectives were formulated using Lyapunov Drift optimisation with penalty. The resulting Cross Layer Controller was verified through simulation. The simulator revealed that the proposed Cross Layer Controller can track the desired level of shared state inconsistency within a narrow margin. It was also shown that the Cross Layer Controller achieves system stability and can more successfully balance the system's two traffic flows than comparable and conventionally used methods. Additionally, the proposed Cross Layer Controller can accommodate both momentary stochasticity in the queues and rapid changes in set points l_i while maintaining the desired time average shared state inconsistency.

Possible extensions of this work include a down-link scheduling policy for scheduling entire application processes across the system's IoT devices. To capture the energy constrained nature of IoT devices, such a system controller can include a maximum time-average transmission energy for each device.

Acknowledgements

This work is funded in part by the Swedish Research Council (VR) under contract number C0590801 for the project Cloud Control. Maria Kihl, Anders Robertsson, and William Tärneberg are members of the Lund Center for Control of Complex Engineering Systems (LCCC) funded by the Swedish Research Council (VR) and the Excellence Center Linköping - Lund in Information Technology (ELLIIT). Maria Kihl is part of the Walleberg Autonomous Systems and Software Program (WASP).

References

[1] P. Persson and O. Angelsmark, "Calvin–merging cloud and IoT," *Procedia Computer Science*, vol. 52, pp. 210–217, 2015.

[2] F. Paganelli and D. Parlanti, "A DHT-based discovery service for the Internet of Things," *Journal of Computer Networks and Communications*, vol. 2012, 2012.

[3] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid, "Internet of things in the 5G era: Enablers, architecture, and business models," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 510–527, 2016.

[4] T. Shuminoski and T. Janevski, "Lyapunov optimization framework for 5G mobile nodes with multi-homing," *IEEE Communications Letters*, vol. 20, no. 5, pp. 1026–1029, 2016.

[5] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive MIMO for next generation wireless systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 186–195, 2014.

[6] L. Georgiadis, M. J. Neely, L. Tassiulas *et al.*, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends® in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

[7] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 16, no. 2, pp. 396–409, 2008.

[8] L. B. Le, E. Modiano, and N. B. Shroff, "Optimal control of wireless networks with finite buffers," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 4, pp. 1316–1329, 2012.

[9] J. Ryu, L. Ying, and S. Shakkottai, "Back-pressure routing for intermittently connected networks," in *2010 Proceedings IEEE INFOCOM*, March 2010, pp. 1–5.

[10] G. Fersi, W. Louati, and M. B. Jemaa, "Distributed hash table-based routing and data management in wireless sensor networks: a survey," *Wireless networks*, vol. 19, no. 2, pp. 219–236, 2013.

[11] J. Dürango, W. Tärneberg, L. Tomás, J. Tordsson, M. Kihl, and M. Maggio, "A control theoretical approach to non-intrusive geo-replication for cloud services," in *IEEE 55th Conference on Decision and Control (CDC) 2016*. IEEE, 2016, pp. 1649–1656.

[12] Simpy. [Online]. Available: https://pypi.python.org/pypi/simpy

[13] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[14] CLC IoT simulator. Lund University. [Online]. Available: https://gitlab.com/eit-wit/scheduling_simulator