# PR$^3$ : Power Efficient and Low Latency Baseband Processing for LTE Femtocells

Nishant Budhdev, Mun Choon Chan, Tulika Mitra
School of Computing, National University of Singapore
{nishant,chanmc,tulika}@comp.nus.edu.sg

***Abstract*—In order to provide greater network capacity, the use of small base stations such as Femtocells has increased to allow higher spectrum reuse. In these Femtocells, base station designers have started to explore the use of general purpose multi-core architectures to provide greater flexibility.**

**Multi-core architectures allow power-performance trade-off possibilities through techniques such as Dynamic Voltage Frequency Scaling (DVFS) and Power Gating. In this work, we propose a power management framework based on reinforcement learning called PR$^3$, which uses both DVFS and Power Gating. Our approach is unique as it introduces a feedback from the network scheduler and baseband processor to the Power Governor, so that information about both the network and computation workloads are included in the decision making.**

**Evaluation on a hardware platform (Odroid XU3) running PHY LTE uplink baseband processing benchmark, shows that PR$^3$ performs well in terms of both power and latency. It is able to save upto 50% power while maintaining low processing latency. PR$^3$ is also adaptive, making it effective over a wide range of traffic loads.**

## I. Introduction

Future generations of cellular networks will be driven by the need to support Internet access with shorter latency and larger data usage. The next generation of cellular networks, the $5^{th}$ Generation (5G), will also be expected to provide support for new use cases such as Internet-of-Thing (IoT) services. To support these requirements, cellular networks have embraced heterogeneity. Heterogeneous networks (HetNet) are composed of a combination of different base station types which can support multiple access technologies.

Early generations of cellular networks were primarily composed of Macro and Microcell base stations, which cover a large area, and are owned and maintained by the service providers. However recent years have seen an increase in the deployment of Femtocell base stations which have substantially shorter range. By 2012, Femtocells had already outnumbered traditional base station installations [1]. As of 2016, it is estimated that nearly 18 million Femtocells have been deployed [2]. This number is estimated to double by 2021 to 37 million. Today, over 90% of these Femtocells are installed in residential areas, and the initial hardware cost along with operating costs are borne by customers. This is a major limitation as each network protocol update requires customers to buy new hardware for accessing new and updated features. As a result, even though Femtocells promise higher network capacity and better service quality to customers, reducing the fixed and operating costs for Femtocells is necessary to accelerate the rate of adoption.

One way to improve flexibility and reduce hardware cost, is to use general purpose processors for baseband processing. Traditionally baseband processing on base stations is performed by an ASIC or DSP [3]. Using ASICs is expensive as the non-recurring engineering (NRE) costs, for a dedicated ASIC design, are exorbitant. On the other hand, DSP's require complex programming, making integration into Femtocells tedious. In contrast, today's general purpose processors, like the x86 family or the multi-core ARM Cortex-A series, can meet the processing demands while providing ease of programming. Furthermore, general-purpose processors are cheap and provide flexibility for future modifications. This flexibility also allows manufacturers to utilize off-the-shelf components to design Femtocells with different capabilities. Multiple recent works show that baseband processing can be performed using GPU [4] or general-purpose processors [5][6].

The major operating cost associated with Femtocells arises from its power consumption. Femtocells can consume up to 11W [7], of which 40% is attributed to the baseband processor. Furthermore, general-purpose processors today provide techniques, such as DVFS and Power Gating, for achieving different power-performance trade-off. The power-performance trade-off on these multi-core general-purpose processors is regulated by the Power Governor. Existing Power Governors use only processor utilization to determine the optimal trade-off. This is a major drawback for baseband processing as these governors do not take into account network statistics and as a result are unable to adapt to traffic load variations.

The main contribution of this paper is a new Power Governor algorithm based on Reinforcement Learning called PR$^3$. PR$^3$ takes into account: the expected network load provided by the LTE network scheduler, the current system state (e.g., queue length) and past performance information (e.g., recent average latency and power consumption). PR$^3$ combines these attributes to determine the appropriate action to save power while maintaining network QoS constraints such as latency and loss. In short, the three types of attributes: **P**roactive, **R**eactive and **R**etrospective respectively, along with **R**einforcement Learning form **PR**$^3$ .

We evaluate PR$^3$ on a real multi-core platform (Odroid XU3 board) running the PHY benchmark [8], an open-source LTE uplink baseband processing implementation. We show

that PR$^3$ produces good performance across a wide range of network traffic load for Femtocells. At low network load, PR$^3$ achieves average subframe processing latency well below 1ms while reducing the power consumption by 50% as compared to existing Power Governor algorithms. At higher network load, PR$^3$ achieves lower average subframe latency while reducing the power consumption by 20%.

The paper is organized as follows: we present the background in Section II, the framework is described in Section III and the proposed algorithm in Section IV. Evaluation results are presented in Section V. Finally, we present related work in Section VI and the conclusion in Section VII.

## II. Background

### A. Overview of LTE and Femtocells

In LTE, the bandwidth is divided into multiple subcarriers of 15kHz each. As there can be tens of thousands of subcarriers on large base stations, subcarriers are grouped into a Physical Resource Block (PRB) to reduce resource allocation complexity. PRBs are allocated to users every subframe for transmission. For LTE, each subframe is 1ms. Data transmitted within each subframe depends on the modulation scheme, the coding rate, and the number of PRBs allocated. Higher layer modulation schemes or better coding rates transmit larger amounts of data. For example, 64QAM transmits up to 3 times more data than QPSK within a subframe.

LTE base stations perform the following functions:

1. Network Scheduling: All communication between the base station and mobile devices is scheduled by the network scheduler. It allocates PRBs to users, based on Channel Quality (CQI), past throughput, etc.
2. Baseband Processing: The baseband converts analog signals into computer readable digital bits and vice versa.
3. RF Processing: The RF performs static processing on all analog signals received or transmitted.
4. Power Amplification: It is used to amplify signals for downlink communication.

Base stations are responsible for upto 60% of the power consumed by cellular networks [9]. This costs network operators over $10 billion per year in operating costs [10]. Base stations can be divided into different types based on their coverage and peak network load. Micro and Macrocell base stations, which cover areas ranging from 0.21 to 2.6 $km^2$, are owned and maintained by the service provider. The peak network speed supported lies in the range of 300-600 Mbps. These base stations can consume up to 1320W [7], of which 13-37% is attributed to the baseband processor.

On the other hand, Femtocells cover a much smaller area ranging from 250 to 1400 $m^2$. They are typically installed in residential areas and can support upto 64 active LTE connections at a given time [11][12]. Depending on the Femtocell architecture, the peak network load supported lies in the range of 50-70 Mbps. Femtocells consume up to 11W [7], of which 40% is due to baseband processing. Although Femtocells have significantly lower power consumption as compared to Macrocells, the number of Femtocells continues to increase at a rapid speed, soon making Femtocells the major power consuming component in a cellular network.

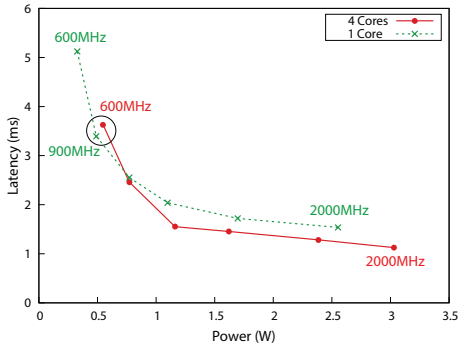



Fig. 1. Power vs Latency plot at different frequencies for 1 and 4 active cores.

### B. Baseband Processor

Historically, baseband processing has been implemented on base stations using ASICs or DSPs. These specialized implementations make hardware development expensive, time consuming and inflexible for introduction of new features. As each generation of cellular networks has seen multiple protocol changes, service providers are typically forced to use new hardware to support each change.

Alternatively, general-purpose processors provide easy migration to new and updated standards. Recent general-purpose multi-core architectures[13][14] show that Multicore System-on-a-chip (MCSoC) designs can be used for baseband processing, as they are capable of billion-operations-per second level throughput [15]. General purpose MCSoC reduce design costs while providing greater flexibility at the base station to introduce new functionality. Additionally, with the recent introduction of Network Function Virtualization (NFV), physical layer functions can be implemented on a general purpose MCSoC to reduce costs and improve flexibility.

### C. Reducing Consumption on Multi-Core Proessors

According to Desset et al. [16], power consumption of the baseband processor on Femtocells makes up 43% and 32% on the uplink and downlink respectively. Hence, it is important to manage power consumed by baseband processing to reduce operating costs.

Power consumption for general-purpose processors can be stated as:

$$P = cV^2f + VI$$

where $c$ is the internal capacitance, $V$ is the supply voltage, $f$ is the operating frequency and $I$ is the current flow through the processor. The first half of the equation, consisting of capacitance, voltage and frequency, constitutes the dynamic power. This is the power consumed by the processor while in operation. Dynamic power can be reduced by either decreasing the voltage and/or the frequency at runtime. This is known as Dynamic Voltage Frequency Scaling (DVFS). This decrease in voltage and frequency also corresponds to lower processing throughput. However, using DVFS causes the processor to be

unavailable for a short duration every time the frequency and voltage are changed. The second half of the equation known as static power; is the power consumed when the processor is idle. Static power can be reduced by putting the processor to sleep. This technique is called Power Gating [17]. Various architectures provide different sleep states with increasing power savings and wake-up latency. We present information on DVFS and Power Gating on baseband processors below.

*1) DVFS:* Figure 1 shows latency as a function of power consumed at different frequencies for baseband processing (using PHY benchmark [8]). Each point in the plot refers to a unique frequency level, starting from 600MHz to 2000MHz with intervals of 300MHz. We see that increasing the frequency, has reduced gains for baseband latency at higher frequencies. Conversely, at lower frequencies, power reduction is negligible as latency increases exponentially.

Furthermore, while power at 600 MHz using 4 cores is higher than at 900 MHz with 1 core, the baseband latency is also higher in the former case (circled in Figure 1). Consequently, selecting an optimal operating frequency on multi-core baseband processors is non-trivial.

*2) Power Gating:* Wake-up latency for Power Gating varies across different architectures ranging from 1-100$\mu$s for x86 architecture, to as high as 2800$\mu$s for ARM Cortex-A15 [18][19]. In the case of ARM, the cost involved is equivalent to the duration for nearly 3 LTE subframes. While it is desirable to use x86 architecture due to the low wake-up latency, x86 processors often consume significantly high power as compared to ARM processors. Hence for Femtocells, with total power consumption of only 11W, ARM processors are more suitable for baseband processing.

Sjalander et al. [8] use Power Gating to reduce power consumption of the baseband processor by estimating per subframe workload. The work has two major drawbacks: it is architecture specific and does not perform well under realistic network load. Additionally, the high cost for wake-up on ARM platforms, makes it sub-optimal to use Power Gating on a per subframe basis.

### D. Limitations of Existing Power Governors

Power Governor is a software library which can monitor and regulate power at fine granularity [20]. Linux provides four Power Governors(Table I), on general purpose platforms, to reduce dynamic power consumption using DVFS. Performance and Powersave are static governors wherein the frequency remains fixed the entire time. In contrast to these static options, Ondemand and Conservative power governors calculate the running frequency every sampling period. The operating frequency for these governors is based on the CPU utilization during the previous interval. The ability of dynamic governors to revise the operating frequency at runtime, allows them to save power for dynamic workloads while maintaining required performance levels. From Figure 1 we can conclude that the Powersave is not a realistic option for baseband processing, as it will have a significant impact on the network Quality of Service (QoS). Hence we omit Powersave from our evaluations.

TABLE I
LIST OF LINUX GOVERNORS. RANGE IS FOR ODROID XU-3. [21]

| Policy | Description | Range (MHz) |
|---|---|---|
| Performance | This policy sets the operating frequency to the maximum. | 2000 |
| Powersave | This policy sets the operating frequency to the minimum. | 200 |
| Ondemand | The operating system monitors CPU utilization for a fixed interval of time. At the end of the interval the operating frequency is set to optimize CPU utilization. | 200 - 2000 |
| Conservative | Similar to Ondemand in terms of calculation of optimal frequency. However, the frequency is increased/decreased by a fixed step size every interval. The default step size is 100MHz. | 200 - 2000 |

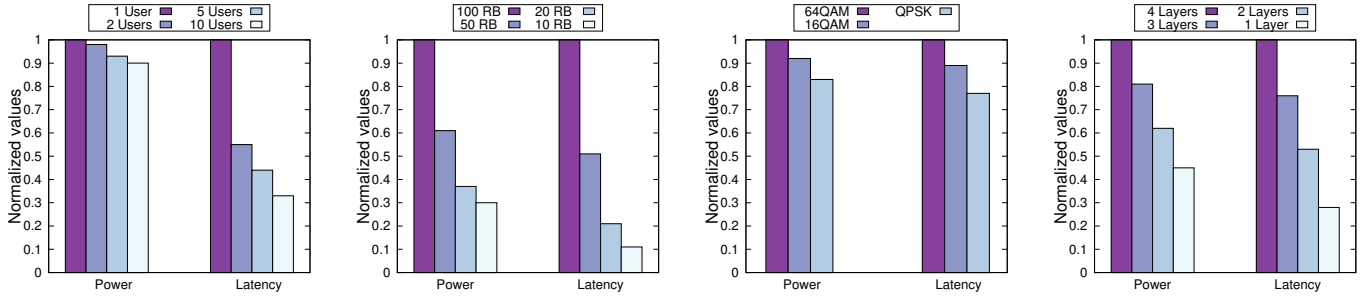
All management policies in Table I work in isolation. For example, these Power Governors have no feedback regarding subframe baseband processing latency and do no take into account the cost associated in changing the system state. Additionally, they do not take into account network load, queue length, subframes dropped, etc., and are therefore unable to adapt to traffic load variations in cellular networks. This absence of feedback has an adverse impact on baseband processing latency and power consumption. On the other hand, maintaining high operating frequency to lower average and maximum baseband processing latency can also cause the processor to overheat and shutdown. This affects the network QoS as all incoming subframes will be dropped while the processor is switched off and cooling down.

Power Governors for baseband processors are further limited by the fact that the workload varies significantly every millisecond. This implies that Power Governors should be simple and require minimal overhead. Furthermore even when limited future information is known, if the time taken by the Power Governor is large the information becomes stale and can in turn have an adverse impact on subframe latency and power consumption. It is important to note that the Power Governor is a complimentary component that does not require any modification to the LTE stack.

The key novelty of PR$^3$ is, it takes into account both the network traffic and the baseband processing load by introducing a feedback from the network scheduler and the baseband processor to the Power Governor to improve power-latency trade-off. Consequently PR$^3$ is able to reduce both the subframe latency and baseband power consumption.

## III. FRAMEWORK OVERVIEW

The proposed framework is applicable to both uplink and downlink baseband processing. However, for brevity, we will present the framework only for one direction (uplink) as the algorithm can be easily extended to the other direction (downlink). The uplink direction is chosen because we use LTE baseband processing benchmark (PHY) from Sjalander et al. [8] in our evaluation and the benchmark performs only uplink baseband processing.




(a) Number of users (b) Number of PRBs (c) Modulation scheme (d) Number of layers

Fig. 2. Normalized Power and Baseband processing Latency trends for different scenarios. All the experiments are run at the highest system frequency. In all cases 100 PRBs are transmitted per subframe, except for Figure(b) which shows the variation in number of PRBs.

The LTE base station front-end receiver (uplink) consists of multiple components. Operations such as radio receiver, receive filter and Fast Fourier Transfrom (FFT) are performed statically on all received data. These constitute the RF component in Figure 3. Once the FFT has been computed, the data is stored in a small buffer before baseband processing is performed by the Baseband Processor.

The proposed framework introduces a *feedback*, from the network scheduler and the baseband processor to the Power Governor, with information about the network and baseband processing workload. The Network Scheduler provides the expected network load for a short horizon to the Power Governor. This early information gives the Power Governor additional time to adapt the system configuration, namely the processor frequency and the number of active cores, to be able to meet the expected workload. The expected workload is characterized by 4 parameters: (1) number of users allocated within a subframe, (2) number of PRBs allocated to each user, (3) modulation scheme assigned to each user, and (4) number of transmission layers for each user.

Figure 2 shows the impact of these parameters on power and latency. As each user is processed in parallel, Figure 2(a) shows that the baseband processing is faster when multiple users transmit within a subframe. On the other hand, increasing the number of PRBs allocated to a user increases both power consumption and baseband processing latency (see Figure 2(b)). Figure 2(c) shows that although 64QAM can transmit up to 3 times more data than QPSK, the difference between them in terms of power consumption or baseband latency is quite small. Finally decreasing the number of layers for transmission produces larger reduction in baseband latency as compared to the reduction in power consumption (see Figure 2(d)).

### A. Power Governor Algorithm Overview

We use this framework and propose PR$^3$, a Power Governor algorithm, that uses DVFS in conjunction with Power Gating to reduce power consumption and maintain network QoS. PR$^3$ uses three types of attributes to find the optimal system configuration. The three attributes are proactive, reactive and retrospective. Figure 3 shows how these attributes are taken into account in the framework.

- **Proactive:** Network load estimates received from the network scheduler fall under the category of proactive attributes. These attributes leverage the fact that the LTE base station is responsible for scheduling network traffic. The horizon of the expected network load is based on the time taken to transmit the schedule, the subframe size, time the mobile terminal needs to process the control data received from the base station, interpret the schedule and process the information to send. This *lookahead period* (on the scale of a few milliseconds) provides enough slack time to tune the system parameters to provide the desired performance w.r.t. both power consumption and baseband processing latency. This is a unique feature over existing Power Governors that do not use this knowledge.
- **Reactive:** Reactive attributes refers to information about current network state. PR$^3$ uses the current buffer queue as a reactive attribute. This allows it to rectify situations when it reaches sub-optimal states based on proactive attributes, reducing the likelihood of going into overload or dropping subframes due to buffer overflow. Other attributes such as temperature can also be monitored actively to provide a policy which ensures the operating temperature remains below a threshold.
- **Retrospective:** Retrospective attributes refers to the feedback inputs to PR$^3$. The feedback provided by the recent average baseband processing latency ensures that if the system consistently misses the targeted baseband processing latency, then the feedback will force PR$^3$ to increase the processing capacity. Another retrospective attribute considered by PR$^3$ is the power consumed by the baseband processor. Sub-optimal actions, which increase power consumption significantly, can be identified and avoided in the future.

Existing Power Governors are retrospective in nature and look only at the processor utilization. PR$^3$ utilizes information on the expected load (proactive), current load (reactive) and past performance (retrospective). The ability to look at these attributes in an integrated manner is useful because it allows PR$^3$ to determine how to exploit DVFS and Power Gating which have very different overheads and non-trivial trade-offs.

### B. Design Considerations and Objectives

The framework takes the following constraints into account:

- No subframe should be dropped due to buffer overflow.

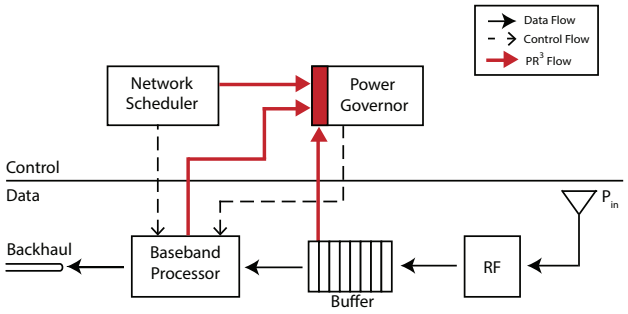



Fig. 3. Proposed Power Governor framework for PR$^3$.

- Frequency of switching system states should not affect Network QoS.
- Computationally inexpensive power management algorithm to reduce overhead.

The two objectives are:

1. Ensure the average subframe processing latency is less than or equal to the sub-frame duration (1ms).
2. Minimize power consumption.

The objectives are to met in the above given order. The first objective is important, as any governor which tries to reduce power consumption should not impact the network throughput and latency significantly. The second objective ensures power efficiency when latency requirements are met.

## IV. Power Governor Algorithm

Reinforcement Learning (RL) [22] based algorithms are used to find the optimal solution to sequential decision problems. The algorithms use an agent which performs actions with the objective to increase reward or reduce cost. The agent interacts with the environment at discrete intervals. At each such time step $t$, the agent receives an observation which includes the reward/cost based on the state $s$ of the environment. The agent then selects an action $a$ to perform based on the received information. The probability of selecting an action $a$ under state $s$ is called a policy $\pi(s,a)$.

The long-term reward after time $t$, is defined as $R_t = \sum_{t=0}^{\infty} \gamma^t r_{t+k}$. Here, $\gamma$ is called the discount factor which determines the importance of future rewards. Therefore, if $\gamma$ is 1, the future reward is as important as the immediate reward, whereas if $\gamma$ is 0, the agent does not care about future rewards, i.e., greedy approach. The expected $R_t$, starting from state $s$, taking action $a$ and following policy $\pi : Q^{\pi}(s,a) = E\{R_t | s_t = s, a_t = a\}$ is given as $Q^{\pi}(s,a)$. The optimal $Q$ value is defined as $Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$.

$$Q^*(s,a) = E\{r + \gamma \cdot \max_{a'} Q^*(s',a') | s_t = s, a_t = a\}$$
$$= \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma \cdot \max_{a'} Q^*(s',a')]$$

This equation is known as the Bellman optimality equation [23]. Here $P^a_{ss'}$ refers to the transition probability, which is defined as the probability of the environment to reach state $s'$, by taking action $a$ at state $s$. The reward associated with this transition is referred as $R^a_{ss'}$ in the equation. However, it is often impossible to calculate the value of $P^a_{ss'}$ and $R^a_{ss'}$, making it difficult to find the optimal solution. This could be because of the environmental setup or when the state space is enormous making it impractical to calculate these values.

Q-learning [24] solves this issue by converging to the Bellman optimal solution incrementally without requiring prior knowledge of either $P^a_{ss'}$ or $R^a_{ss'}$. The following equation describes the underlying principle of Q-learning:

$$Q(s,a) = Q(s,a) + \theta \cdot \{r + \gamma \cdot \max_{a'}[Q(s',a')] - Q(s,a)\}$$

In Q-learning, the expected reward for the state action pair $(s,a)$ is, $r + \gamma \cdot \max_{a'}[Q(s',a')]$. Q-learning then updates the Q values incrementally, with the state action pairs updated sequentially by using the difference $\{r + \gamma \cdot \max_{a'}[Q(s',a')] - Q(s,a)\}$. Repeating this procedure, Q-learning converges to the Bellman optimal solution.

The environment in the case of PR$^3$ consists of the base station. The state $s$ of the environment, corresponds to the network state. Whereas action $a$, refers to a possible system configuration for the baseband processor. The objective of PR$^3$'s agent is to minimize the cost function. We discuss each of these aspects in detail below.

The network state is defined as the combination of $N_t$ (expected network load) and $Q_t$ (queue length at the baseband processor). Expected network load consists of four parameters which determine the subframe load, as discussed in Section 3. With this configuration, the total number of unique network states is more than $10^{18}$. We reduce this state space through clustering. Based on the results from Figure 2, the network load is clustered into 101 discrete states. Since Queue length has a small number of discrete values, we do not modify it. Therefore $N_t \in \{0, 1, 2, ..., 100\}$ is the network load as determined by the LTE network scheduler (0% to 100%), and $Q_t \in \{0, 1, 2.., 10\}$ is the number of subframes in the queue.

The action space is the union of $F_t$ and $C_t$, which refers to the operating frequency and number of active cores respectively. $F_t \in \{2, 3, 4, ..., 20\}$ corresponds to the range of frequencies 200MHz to 2000MHz on the platform used in the evaluation. Since our platform has 4 cores, $C_t \in \{1, 2, 3, 4\}$ corresponds to the number of active cores. At each interval PR$^3$ selects an optimal action from this action space. For PR$^3$, each time step is equivalent to 10 subframes (i.e., 10ms in LTE). This is done to reduce the overhead of PR$^3$ as well as provide fine grain control to adapt to varying network load.

For our model, we use a cost function instead of a reward function. The cost function is defined as:

$$R(s_t, a_t) = \begin{cases} p_t + c_t + d_t & l_t < L \\ P_{max}(l_t/L) + c_t + d_t & L \leq l_t \end{cases} \quad (1)$$

Here, $p_t$ is the power consumption of the baseband processor at time $t$. $c_t$ corresponds to the cost for action $a_t$. If the next optimal system state is the same as the current system state then the cost is 0. Additionally, cost for switching frequencies is lower than that of increasing/decreasing the number of active cores because of the large overhead involved in the latter. The

cost for such actions are derived empirically for the given hardware. $d_t$ corresponds to the number of subframes dropped in the interval $(t-1, t]$. $P_{max}$ corresponds to the maximum power consumption of the hardware. $l_t$ refers to the average baseband processing latency for all subframe in $(t-1, t]$. $L$ is the desired baseband processing latency for the system.

When the average processing latency is lower than the threshold $L$, the cost function only includes the power consumption. On the other hand, when average baseband processing latency exceeds the desired latency, we multiply the constant $P$ to the ratio $l_t/L$ to ensure that the cost function increases significantly, discouraging the agent from moving into such states.

The linear nature of the cost function ensures the overhead for calculation is minimal. Moreover, using a piece-wise function helps model two distinct behaviors into our algorithm based on the latency threshold. Each behavior corresponds to a unique objective mentioned in Section III.

To accelerate the convergence of Q-learning, we use information from Figure 1 to reduce the action space. The action space is reduced to 33 states (200-1200MHz for 1 core, 900-1200MHz for 2 and 3 cores, 900-2000MHz for 4 cores). Furthermore, to reduce the number of states for exploration, we use the batch update method. For any action $a_0$ and $a_1$, we define $a_1 > a_0$ if action $a_1$ has equal number of active cores and higher frequency than in $a_0$. From Figure 1, we observe that, if the baseband processor exceeds the latency constraints at state action pair $(s, a_1)$, it will also exceed at $(s, a_0)$, $\forall a_0 < a_1$. Therefore, we do not explore such redundant states thereby reducing the amount of time required for training.

### A. Training

PR$^3$ is calibrated to have a greedy behavior. This implies that we do not need to keep track of the transition probability between different states as the most optimal action is based only on the cost associated with the current state and action. This gives us the freedom to use any trace during learning while keeping the final optimal policy independent of the traces used during the training phase.

Moreover, since it is impossible to cover all traces due to the large state space, we train on randomly generated subframe workloads. Each randomly generated workload is executed for 500ms during which the power consumption and subframe processing latency are measured. After 500ms another subframe is generated randomly and is executed for the next 500ms. We do this until we have explored each network load bucket (101 in total) once. For our training phase we use less than 450 unique subframes. Finally, during evaluation, we switch-off learning. Hence, the results are based entirely on offline learning of random subframes.

### B. Flexibility of PR$^3$

**Network Load Flexibility:** Network load varies among different base stations, and each base station also sees temporal variation throughout the day. Using online-learning PR$^3$ can find a policy that is optimal locally to a base station by learning over its unique workload.

**System Architecture Flexibility:** Given the wide variety of multi-core architectures available today, it is difficult and expensive to obtain an optimal policy for each architecture heuristically. By using power consumed by the baseband processor as an input, PR$^3$ can adapt to different multi-core architectures effortlessly. Different architectures often have different constraints for DVFS and Power Gating. PR$^3$ handles this by modifying the action cost $c_t$ within the cost function. Consequently, we can also modify the set of possible actions based on the system capabilities by restricting the action space.

## V. Evaluation

### A. Experimental Setup

*1) Hardware Platform:* All experiments are performed on Odroid XU3 from Hardkernel [25]. The SoC implements ARM big.LITTLE technology with a cluster of four ARM Cortex-A7 cores (LITTLE cores) and a cluster of four ARM Cortex-A15 cores (big cores). We run our experiments exclusively on the A15 cluster as the A7 cluster does not provide the required performance for 1ms subframes. Each cluster provides DVFS capability. However, cores within a cluster operate at the same frequency. The A15 cores can be clocked between 200MHz and 2000MHz with increments of 100MHz. The maximum transition latency for switching between frequency levels is 100$\mu$s. The input voltage for each frequency level is controlled by the hardware. We evaluate PR$^3$ against existing Power Governors: Conservative, Ondemand and Performance.

A brief description of the baseline Power Governors is given in Table I. Information regarding power consumption, core temperature and running frequency is obtained by sampling the sensors on the board at the rate of 10Hz. The ambient temperature during evaluation is 27°C.

*2) PHY Benchmark:* To simulate the uplink baseband processing, we run the PHY benchmark [8] provided by Sjalander et al., on the Odroid board. PHY is an open source implementation of LTE uplink baseband processing.

To meet the processing requirements, the benchmark implements parallelization and pipelining. The base station processes each user in parallel as the processing is independent for each user. Furthermore, each user's processing is parallelized. Matched filter, IFFT, windowing and FFT kernels within Channel Estimation can be parallelized on the basis of number of layers and number of antennas. Therefore, the maximum amount of parallelization for Channel Estimation is 16 (4 layers x 4 antennas). Similarly, Data Demodulation and Decoding, Antenna Combining and FFT can be processed in parallel by up to 24 tasks (6 symbols x 4 layers).

The benchmark is instrumented to obtain queue length, dropped subframes and per subframe processing latency. The subframe processing latency is calculated as the time between, the subframe is added to the buffer (after RF processing) to the time the subframe completes processing. To obtain the number of dropped subframes, we monitor the queue length at all times. The default PHY benchmark settings drop subframes, when the queue length exceeds 10 subframes. All incoming subframes are dropped until the queue length drops below 10.

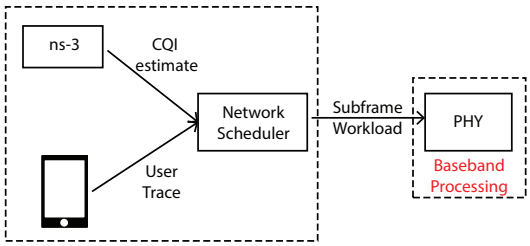



Fig. 4. Generating Subframe Workload from Network Traces.

The benchmark takes subframe allocation as an input. The allocation contains the number of users allocated within the subframe, number of PRBs allocated, number of transmission layers and modulation scheme for each user. We modify the benchmark to use custom network allocation as input. More information on generation of these input traces is given below.

### B. Experimental Parameters

*1) Configuration and Policies:* The basic assumption made in all our experiments is that the system does not drop any subframe under maximum load when operating at the highest frequency and maximum number of active cores. The results of the Performance Governor validate that the hardware resources are sufficient to support LTE bandwidth. Consequently our system consisting of four Cortex-A15 cores, can support an inter-subframe arrival time of *1ms*.

*2) Network Traces:* To generate traces for evaluation, we use ns-3 to generate Channel Quality Indicator (CQI) estimates for a large number of users. Each user is assigned a random position and velocity. We use 3 values for velocity: 0, 5 and 15 km per hour. To simulate fading behavior due to distance, we use the Log Normal Propagation Loss Model from ns-3. For simulating channel fading behavior independent of the distance, we use the Rayleigh fading model. The two models combine to produce a unique fading pattern for each user.

Contrarily, simulating realistic data usage traces is more difficult as simple TCP/UDP connections do not capture the complex interactions within the network. Consequently, we use a mobile device to capture real network traces.

The trace for background data is generated by tracking background network traffic on an Android phone running WhatsApp, Google Hangouts, Skype, Gmail and other Google native applications with no user interaction. We capture the packet traces for over 16 hours and use it to emulate network transmissions for a large number of users whose CQI estimates we produced using ns-3. The data flow is shown in Figure 4.

For generating traces for foreground data, we combine traffic from two sources. First, we upload multiple files and photos of different sizes. The size of files vary from 60KB to 30MB and the size of photos uploaded varies from 450KB to 750KB. Next, to simulate genuine user interactions, we allow random users to upload files of different sizes from 20KB to 1MB, at random intervals. These two workloads are combined with the background traffic to obtain the final network traces.

For network scheduling, we use the Proportional Fair algorithm [26] to combine the CQI and network traces described above to produce a per subframe network schedule. Proportional Fair tracks average user throughput and current CQI to determine the users for transmission within each subframe. Furthermore, we use the CQI values to map each user to the corresponding modulation schemes by using information provided in the E-UTRA 3GPP release [27].

TABLE II
SYSTEM AND NETWORK ATTRIBUTES FOR DIFFERENT TRACES. (NO SUBFRAME DROP)

| Trace (Load) | Power Governor | Power (W) | | Latency(ms) | | | Temp |
|---|---|---|---|---|---|---|---|
| | | Ave. | Max | Ave. | 90% | Max | (C) |
| Low | Performance | 2.73 | 3.18 | 0.31 | 0.62 | 1.76 | 55.3 |
| | Conservative | 1.59 | 2.79 | 0.44 | 1.03 | 4.33 | 52.2 |
| | Ondemand | 1.89 | 2.62 | 0.40 | 0.94 | 5.01 | 53.8 |
| | $PR^3$ | 0.84 | 1.92 | 0.49 | 0.88 | 3.37 | 49.7 |
| Medium | Performance | 2.97 | 4.28 | 0.58 | 1.45 | 3.01 | 65.4 |
| | Conservative | 1.81 | 3.29 | 0.85 | 2.05 | 9.88 | 56.1 |
| | Ondemand | 1.76 | 2.61 | 0.78 | 2.17 | 5.97 | 53.9 |
| | $PR^3$ | 1.48 | 2.77 | 0.74 | 1.77 | 3.36 | 54.1 |
| High | Performance | 3.17 | 4.73 | 0.91 | 2.33 | 5.23 | 65.3 |
| | Conservative | 2.11 | 3.90 | 1.17 | 2.71 | 13.4 | 55.5 |
| | Ondemand | 2.10 | 3.18 | 1.22 | 2.87 | 7.12 | 55.6 |
| | $PR^3$ | 1.68 | 3.76 | 1.04 | 2.36 | 5.12 | 53.8 |

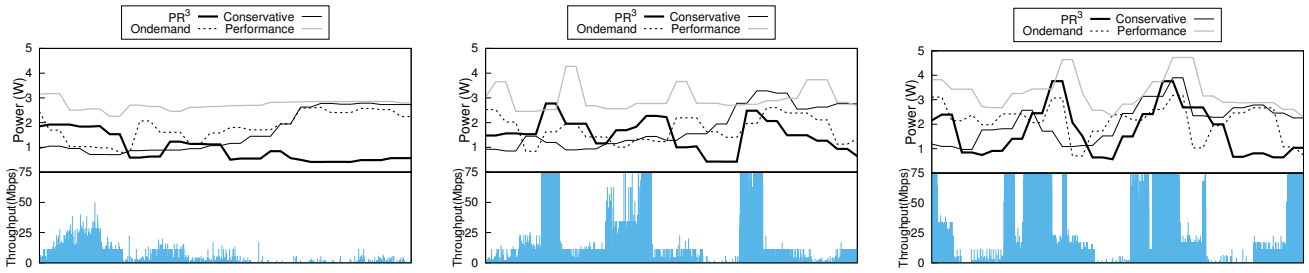
In the evaluation, we vary the number and types of active users to obtain traces with different load. We divide our traces into 3 broad categories. Low (load) traces have an average throughput of less than 10Mbps. Medium (load) traces have throughput between 10 to 20 Mbps. The medium trace in Figure 5 and Table II has a throughput of 10.5Mbps. The high (load) traces have throughput over 20Mbps. The peak throughput for the Medium and High trace is over 70 Mbps.

### C. Low Load

It has been observed that about 42% of the time (10hr per day) [7][28] the base station operates in a low load condition. We expect $PR^3$ to achieve good results during low periods because it can vary the system state to take advantage of these low load periods to provide sufficiently low latency ($< 1ms$) with minimal power consumption. To obtain low load traces, we use a trace consisting of background data from mobile devices combined with a few active users. The average traffic load is 2.48Mbps. Detailed results are given in Table II.

In general, for Performance, Conservative and Ondemand Power Governors, when the average power consumption is high, the subframe latency decreases. For example, Performance consumes the most power (2.73W) but has the lowest latency (0.31ms); while Conservative consumes 1.59W but has a higher average latency of 0.44ms. Interestingly, even though the Ondemand Governor has lower average latency (0.40ms), it has a higher maximum latency compared to the Conservative Governor, showing that it is unable to keep up in some cases when there is a sudden surge in traffic.

We observe that the $PR^3$ consumes much less power compared to all other baselines. On average, $PR^3$ saves 47%, 55% and 67% over Conservative, Ondemand and Performance respectively. The average latency of $PR^3$ is 0.49ms, well below 1ms. In fact, because of $PR^3$'s adaptability, the $90^{th}$




(a) Low (b) Medium (c) High

Fig. 5. Comparing power consumption of different Power Governors. The bottom half of graphs show the upload throughput for the corresponding traces.

percentile and maximum latency are 0.88ms and 3.37ms respectively; which are lower than the corresponding values for Conservative and Ondemand, while consuming lower power.

High baseband temperature can lead to processor shutdown. This is a major issue in places with hotter climates or non-air conditioned indoors. PR$^3$ mitigates this issue by having lower operating temperature (49.7°C) as compared to other governors. This difference becomes significant when compared with the Performance Governor (55.3°C).

### D. High Load

The average traffic rate for the high load trace is 20.84Mbps. When the traffic load is mostly high, Power Governors have less opportunities to save power. Instead, the main objective is to maintain low latency in order to satisfy network QoS.

The result here are similar to the low load case, except that with much higher load, the average latency for Performance, Conservative and Ondemand Power Governors increases to 0.91ms, 1.17ms and 1.22ms respectively. The results for maximum latencies are also much larger, reaching 13.4ms in the case of Conservative Governor.

Again, PR$^3$ outperforms other governors in power savings. The power savings are 20%, 20% and 46% over Conservative, Ondemand and Performance respectively. Even the average latency for PR$^3$ (1.04ms) is significantly lower than Conservative (1.17ms) and Ondemand (1.22ms). Interestingly, the maximum latency of the Performance Governor is higher than PR$^3$ even though it consumes 89% more power.

The power consumption and throughput, for the low, medium and high load traces, for various Power Governors are shown in Figure 5. The traffic load for medium and high traces changes significantly over time. Compared to the Conservative and Ondemand Governors, PR$^3$ tracks changes in traffic load much better. When traffic load is high, PR$^3$ switches to a high performance state quicker as compared to Ondemand or Conservative, to maintain low latency. Furthermore when the traffic load is low, it switches to a low power state to conserve energy. These results clearly demonstrate the utility of PR$^3$ in terms of its ability to quickly and effectively adapt to changes in network load.

### E. Effectiveness of Proactive and Reactive Attributes

In this section, we look at impact of Proactive and Reactive attributes on the performance. The results for different variants of PR$^3$ running on a low load trace with an average of 7.2Mbps are given in Table III. Note that when the horizon is 0 subframe (case I and II), the Proactive component is disabled. In these cases the agent chooses an optimal action on the basis of the current subframe only.

TABLE III
SYSTEM AND NETWORK ATTRIBUTES FOR DIFFERENT HORIZONS.

| Case | Horizon (subframes) | Power(W) | | Latency(ms) | |
|---|---|---|---|---|---|
| | | Average | Max. | Average | Max. |
| I | 0 (No Proactive and No Reactive) | 1.015 | 2.775 | 0.80 | 29.70 |
| II | 0 (No Proactive) | 1.173 | 2.857 | 0.65 | 6.72 |
| III | 1 | 1.234 | 2.891 | 0.61 | 6.33 |
| IV | 2 | 1.245 | 2.845 | 0.61 | 5.35 |
| V | 5 (No Reactive) | 1.179 | 2.894 | 0.63 | 4.58 |
| VI | 5 (PR$^3$) | 1.282 | 3.018 | 0.59 | 3.65 |
| VII | 10 | 1.288 | 3.092 | 0.59 | 3.37 |

TABLE IV
SYSTEM AND NETWORK ATTRIBUTES FOR OFFLINE AND ONLINE LEARNING FOR A SIMULATED 24HOUR TRACE.

| Learning type | Power Consumption(W) | | Latency(ms) | |
|---|---|---|---|---|
| | Average | Max. | Average | Max. |
| Offline (PR$^3$) | 1.534 | 3.986 | 0.81 | 4.66 |
| Online | 1.571 | 4.132 | 0.76 | 4.47 |

We consider the impact of removing either the Proactive or Reactive component versus the default PR$^3$ configuration (case VI). Without Proactive (Case II), the average power reduces from 1.282W to 1.173W, whereas without Reactive (Case V), the average power is 1.179W. Hence, the impact on average power consumption is small. The major impact is on the maximum latency which increases from 3.65ms to 6.72ms without Proactive (Case II), and 4.58ms without Reactive in Case V. Hence, the impact of Proactive attributes, which determines the ability to look ahead, is more significant as compared to the reactive attributes.

Finally, Table III also shows that the horizon of 5 subframes is sufficient. Even if the lookahead can be increased to 10 subframes (Case VII), the incremental improvement is small. The most common transmission scheme for Uplink schedules 5ms in advance [27]. Other cases of transmission involve scheduling subframes even earlier, providing even more information to PR$^3$ to select an optimal state.

### F. Online vs. Offline Learning

In this section, we look at the effectiveness of offline learning from randomly generated samples, used in our work,

versus the impact of incorporating online learning. To evaluate the two, we generate a sample 24 hour trace based on the average data traffic profile for a base station [7]. In the case of online training, the Q value pairs are initialized with value equal to the offline learning case. Over the duration of the trace the online learning updates these Q values pairs thereby revises its optimal policy over time.

The results in Table IV show that even when online learning is enabled, the results remain almost the same. Online learning increases the power consumption slightly (2.4%) while reducing the average latency by a small amount (6.7%).

## VI. Related Work

Peng et al. [29] discuss in brief different challenges facing current Femtocell deployments. While they provide an extensive list of network issues, they overlook the cost associated with Femtocells. Others such as Lin et al. [30] propose a distributed Network Management algorithm called RRS, which improves fairness while reducing outage probability for Femtocells. Works such as Sciancalepore et al. [31] have used online Reinforcement Learning to predict traffic patterns. These forecasts are in turn used to fulfill Service Level Agreements (SLA) for 5G networks.

Mishra et al. [32] use a two-tier feedback based control theoretic approach for managing power consumption for different islands with time varying workloads. Other approaches like the one proposed by Xiaorui et. al. [33] use optimal control theory to control power consumption and temperature through DVFS. These approaches require significant overhead affecting network QoS for the baseband processor.

To simulate LTE downlink baseband processing, the WiBench [34] open source kernel suite can be used. The benchmark provides signal processing kernels which can be combined together to obtain LTE downlink processing setup. However it can only process single-user subframes which prohibits simulating realistic traces for LTE.

## VII. Conclusion

In this paper, we first highlight the limitations of existing Power Governors. As these governors only consider processor utilization and do not take into account network load in the decision making process, they are unable to adapt to traffic load variations in the cellular networks. The main contribution of our work is a Power Governor algorithm called $PR^3$ that uses information from both the network scheduler and baseband processor. By integrating this information, $PR^3$ adapts to dynamic workloads. We evaluate the algorithm on the Odroid XU3 board running the PHY benchmark. Evaluation shows that $PR^3$, is able to provide good performance across a wide range of network traffic load. It reduces power consumed under all scenarios and also provides faster baseband processing under medium and high load. $PR^3$ also reduces the worst case baseband latency over existing dynamic Power Governors.

## VIII. Acknowledgement

This work has been supported by National Research Foundation, Prime Minister's Office, Singapore under its Industry-IHL Partnership Grant and Huawei International Pte. Ltd. NRF2015-IIP003.

## References

[1] I. T. . Media, "Small-cell market status report," 2012.
[2] Small Cell Forum, "Market Status Report," 2017.
[3] P. Brauer, M. Lundqvist, and A. Mällo, "Improving latency in a signal processing system on the epiphany architecture," in *IEEE Conference on Parallel, Distributed, and Network-Based Processing*, 2016.
[4] Q. Zheng, Y. Chen, R. Dreslinski, C. Chakrabarti, A. Anastasopoulos, S. Mahlke, and T. Mudge, "Architecting an lte base station with graphics processing units," in *IEEE Signal Processing Systems*, 2013.
[5] M. Srinivasan, C. S. R. Murthy, and A. Balasubramanian, "Modular performance analysis of multicore soc-based small cell lte base station," in *IEEE Conference on Very Large Scale Integration*, 2015.
[6] O. Brini and M. Boukadoum, "Real-time cpu-gpu demodulator for the lte physical layer," in *Symposium on Circuits Systems*, 2016.
[7] G. Auer, V. Giannini, I. Gódor, P. Skillermark, , M. Olsson, M. A. Imran, D. Sabella, M. J. Gonzalez, C. Desset, and O. Blume, "Cellular energy efficiency evaluation framework," in *IEEE VTC*, 2011.
[8] M. Själander, S. A. McKee, P. Brauer, D. Engdal, and A. Vajda, "An lte uplink receiver phy benchmark and subframe-based power management," in *IEEE (ISPASS)*, 2012.
[9] Vodafone Group, "Vodafone Sustainability Report," 2015.
[10] E. Oh, B. Krishnamachari, X. Liu, and Z. Niu, "Toward dynamic energy-efficient operation of cellular network infrastructure," *IEEE Communications Magazine*, 2011.
[11] 2017, https://networks.nokia.com/products/ femtocells.
[12] "Cisco universal small cell 8718 and 8818 data sheet," 2016.
[13] Http://www.tilera.com/.
[14] Http://www.kalrayinc.com/.
[15] T. Instruments, "Lte emerges as early leader in 4g technologies," 2009.
[16] C. Desset, B. Debaillie, V. Giannini, A. Fehske, G. Auer, H. Holtkamp, W. Wajda, D. Sabella, F. Richter, M. J. Gonzalez *et al.*, "Flexible power modeling of lte base stations," in *IEEE WCNC*, 2012.
[17] M. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. Vijaykumar, "Gated-v dd: a circuit technique to reduce leakage in deep-submicron cache memories," in *Symposium on Low power Electronics and Design*, 2000.
[18] R. Schöne, D. Molka, and M. Werner, "Wake-up latencies for processor idle states on current x86 processors," *Springer Computer Science-Research and Development*, 2015.
[19] ARM, "Arm idle states binding description."
[20] Intel, https://software.intel.com/en-us/articles/intel-power-governor.
[21] Https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt.
[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.
[23] R. Bellman, *Dynamic programming*. Courier Corporation, 2013.
[24] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, 1992.
[25] "Odroid-xu3,," http://www.hardkernel.com/main/products/prdt_info.php?g_code=g140448267127.
[26] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, and A. Viterbi, "CDMA/HDR: A bandwidth-efficient high-speed wireless data service for nomadic users," *IEEE Communications Magazine*, 2000.
[27] 3GPP, "TS 36.213 Rel. 9," no. Release 9, 2010.
[28] Sandvine, "Global report phenomena," 2016.
[29] C. Peng, Y. Li, Z. Li, J. Zhao, and J. Xu, "Understanding and diagnosing real-world femtocell performance problems," in *IEEE INFOCOM*, 2016.
[30] M. Lin, N. Bartolini, and T. La Porta, "Power adjustment and scheduling in ofdma femtocell networks," in *IEEE INFOCOM*, 2016.
[31] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5g network slicing resource utilization," *IEEE INFOCOM*, 2017.
[32] A. K. Mishra, S. Srikantaiah, M. Kandemir, and C. R. Das, "Cpm in cmps: Coordinated power management in chip-multiprocessors," in *ACM High Performance Computing, Networking, Storage and Analysis*, 2010.
[33] X. Wang, K. Ma, and Y. Wang, "Adaptive power control with online model estimation for chip multiprocessors," *IEEE Transactions on parallel and Distributed Systems*, 2011.
[34] Q. Zheng, Y. Chen, R. Dreslinski, C. Chakrabarti, A. Anastasopoulos, S. Mahlke, and T. Mudge, "Wibench: An open source kernel suite for benchmarking wireless systems," in *IEEE Symposium on Workload Characterization*, 2013.