New Approaches to Computation-Communication Co-scheduling in Real-time Cyber-Physical Systems

A Synopsis Report submitted in partial fulfillment of the requirements for the degree

of

Doctor of Philosophy
in
Computer Science and Engineering

by
Sanjit Kumar Roy

Under the guidance of
Dr. Arnab Sarkar and Dr. Chnadan Karfa

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

4^{th} December, 2020

Abstract

Cyber-Physical Systems (*CPSs*) like those in the automotive and avionic domains, smart grids, nuclear plants, etc., often consist of multiple control sub-systems running on distributed processing platforms. Most of these control systems are modeled as real-time *independent tasks* or *Precedence-constrained Task Graph* (*PTG*) depending on the nature of interactions between their functional components. These tasks typically read their input parameters via sensors. The sensed inputs are then transmitted as messages over communication channels to processing elements where corresponding control outputs are computed. The outputs in turn, are communicated to actuators as messages through communication channels. *This dissertation presents a few novel real-time task-message co-scheduling strategies for safety-critical CPSs consisting of various types of task and execution platform scenarios.*

The entire thesis work is composed of multiple contributions categorized into four phases, each of which is targeted towards a distinct task/platform scenario. The first phase delves with the design of co-scheduling strategies for independent periodic real-time tasks with associated input and output messages, on a bus-based homogeneous multiprocessor system. Although most scheduling approaches have traditionally been oriented towards homogeneous multiprocessors, continuous demands for higher performance and reliability along with better thermal and power efficiencies, have created an increasing trend towards distributed heterogeneous processor platforms. In the second phase, we have considered the problem of scheduling real-time systems modeled as *PTGs* on fully-connected heterogeneous systems. The tasks considered in both the first and second phases, may have multiple implementations designated as service-levels/quality-levels, with higher service-levels producing more accurate results and contributing to higher *rewards*/*Quality of Service* (*QoS*) for the system. In the third phase, we extend the problem of scheduling *PTGs* on fully-connected platforms, to *CPS* systems where the processors are connected through a limited number of bus based shared communication channels. While the third phase considers the problem of scheduling a single *PTG*, the final phase solves the problem of scheduling multiple independent periodic real-time *PTGs*. The works proposed in the third and fourth phases, endeavour towards the maximization of slack within the generated schedule, which can then be used to minimize energy dissipation in the system. The thesis proposes both optimal and heuristic solution approaches for all its phases. Practical applicability and efficacy of the presented schemes have been extensively evaluated through simulation-based experiments as well as real-world benchmarks.

1 Introduction

Today, *Cyber-Physical Systems* (*CPSs*) are becoming an important part of our daily lives. A *CPS* is composed of physical sub-systems together with computing and networking (cyber sub-systems) where embedded computers and networks monitor and control the physical processes. For example in a traditional aircraft, a pilot controls the aircraft using movable surfaces on the wings and tail, connected to the cockpit through mechanical and hydraulic sub-systems. On the other hand in a fly-by-wire aircraft, the control commands are electronically sent by a flight computer over a network to actuators at the wings and tail, making the aircraft much lighter than a traditional aircraft, resulting in better fuel efficiency.

Many *CPSs* are based on federated architectures where functional sub-components are associated with their own dedicated processing elements, many of which remain severely underutilized. Such underutilization may lead to the deployment of more resources than are actually necessary when the functionalities/applications are allowed to execute in an integrated fashion on a small consolidated number of processing elements. Thus, federated architectures may result in higher design costs compared to more integrated execution of applications on smaller consolidated platforms. However, consolidated architectures lead to significantly increased design complexity due to a higher degree of contention for shared resources (such as processing elements, buses, memories etc). Given a distributed platform consisting of a set of processing elements connected through communication channels, the successful execution of tasks and transmission of messages (while satisfying deadlines and other resource constraints), is essentially a real-time task-message co-scheduling problem. *This dissertation focuses towards co-scheduling strategies for real-time CPSs where functionalities may be represented as independent tasks, Precedence-constrained Task Graphs (PTGs) or even multiple independent applications each represented as a separate PTG. The targeted platforms may consist of homogeneous/heterogeneous processing elements which may either be fully interconnected or connected through shared possibly heterogeneous buses.*

Solution approaches to real-time scheduling problems can be broadly classified as *heuristic* and *optimal*. Heuristic schedule construction methodologies are typically based on the satisfaction of a set of sufficiency conditions and cannot take into consideration all necessary schedulability requirements. Consequently, such scheduling schemes become sub-optimal in nature with their results often deviating significantly from their optimal counterparts. On the other hand, optimal solution approaches take all necessary and sufficient conditions into consideration and have the potential to make a fundamental difference in time-critical systems with respect to performance, reliability, and other non-functional metrics like cost, power, space, etc. Optimal schedules can also act as benchmarks allowing accurate comparison and evaluation of heuristic solutions [1]. Several strategies including automata based synthesis, *CSP/SMP* based modeling, search based technique, etc. have been typically used to construct optimal scheduler for *CPSs*. *This thesis has focused towards the synthesis of optimal scheduler using a state-space search approach or CSP based modeling followed by solution generation using CPLEX [2] a standard industry grade constraint solver*

Though, optimal scheduling solutions may potentially deliver significantly better performance compared to sub-optimal heuristic solutions, finding optimal solutions may become prohibitively expensive for large problem sizes. Further, during design space exploration, multiple quick design iterations are needed and/or powerful server systems may not be available at the designer's disposal. In such cases the designer must resort to sub-optimal yet satisfactorily good polynomial time heuristic solution for the problem at hand. Many heuristic schedulers which are commonly based on variations of the well known *list scheduling strategy* [3–7] are found in literature, particularly for *CPS* applications represented as task graphs.

This dissertation presents a few novel real-time optimal/heuristic offline task-message co-scheduling strategies for safety-critical CPSs consisting of various types of task and execution platform scenarios. In real-time safety-critical *CPSs* where deadline misses may lead to catastrophic consequences, offline scheduling is often preferred as all timing requirements can be guaranteed before putting the system in operation, specially in cases where the task systems are persistent and do not vary dynamically at run time. Additionally, offline scheduling allows time and space complexities involved in solution space exploration to become independent of run-time scheduling overheads.

2 Related Work

Traditionally, scheduling of real-time *independent* tasks on multiprocessor systems has been based on either partitioned or global approaches [8–10]. With partitioning, the multiprocessor scheduling problem is transformed into uniprocessor scheduling problem, where a task is assigned to a designated processor and gets executed entirely on that processor. Well known optimal uniprocessor scheduling algorithms include *Rate-monotonic* (*RM*) (static priority) and *Earliest Deadline First* (*EDF*) (dynamic priority), proposed by Liu and Layland [11]. However, a major drawback of partitioning is that, upto half of the system capacity may remain unutilized in order to ensure timing constraints of a given task set [12]. Unlike the fully partitioned approaches, more global schedulers like *Pfair* [13], PD^2 [14], *ERFair* [15], *Boundary Fair* (*BF*) [16], *SA* [17], *LLREF* [18], *RUN* [19], *DP-Fair* [20] can achieve very high utilization of the system capacity by allowing migrations of tasks among processors. The *Proportional fair* (*Pfair*) scheduler proposed by Baruah et al. [13] is known to be the first optimal global real-time scheduler on multiprocessor systems for tasks with implicit deadlines. Based on *Pfair*, Anderson et al. [15] proposed a work-conserving multiprocessor scheduling algorithm called the *Early-Release fair* (*ERfair*) scheduler. However both the above schemes attempt to maintain proportional fairness at each time slot and incur unrestricted preemption/migration overheads due to this. Recently, Levin et al. [20] proposed a semi-partitioned approximate proportional fair optimal scheduler called *DP-Fair* with much lower and bounded context switching overheads.

The problem of scheduling *PTGs* on multiprocessor systems has also received the attention of researchers over many decades. Various optimal solution approaches, such as linear programming, Best-first search, and other exhaustive enumeration techniques including model-based formal synthesis mechanisms, have been proposed [21, 22]. Prasanna et al. [23] devised a control-theoretic optimal scheduling mechanism for task graphs executing on a homogeneous multiprocessor system. Later, they have extended their scheme to include communication overheads between task nodes [24]. Sarad et al. [1] developed a *mixed-integer linear programming* (*MILP*) based *PTG* scheduling strategy for platforms consisting of a set of fully connected *homogeneous* processing nodes. Liu et al. [25] considered the problem of task node assignment for *PTGs*, among heterogeneous clusters connected through communication links of various transmission capacities. Kanemitsu et al. [26] presented an *Integer Linear Programming* (*ILP*) based optimal task scheduling scheme for fully-connected heterogeneous distributed systems. Hsiu et al. [27] developed an optimal and approximation algorithm for the scheduling of *PTGs* on heterogeneous distributed platforms with shared buses. However, a drawback of this scheme is the simplistic assumption that mapping of task nodes to processing elements are known a priori.

Although optimal scheduling solutions may potentially deliver significantly better performance, it may be noted that computation of such optimal solutions may often become prohibitively expensive for large problem sizes. Therefore, research in this domain has also focused towards the design of low-overhead heuristics that provide quick and satisfactory schedules. Heuristic scheduling of *PTGs* on multiprocessor platforms have often been dealt with *list scheduling* based techniques. These scheduling strategies typically maintain an ordered priority list of all tasks in the *PTG* [3–7] and involves two phases, (i) *task prioritization*: for selecting the highest-priority ready task and (ii) *processor selection*: for selecting a suitable processor that minimizes execution time. Some examples of this class of techniques include the *Modified Critical Path* (*MCP*) [28], *Highest Level First* (*HLF*) [29], *Critical Path On a Processor* (*CPOP*) [3], *Heterogeneous Earliest Finish Time* (*HEFT*) [3], *Predict Earliest Finish Time* (*PEFT*) [4] and *Heterogeneous Selection Value* (*HSV*) [5] algorithms. They attempt to construct a static-schedule for the given *PTG* to minimize the overall schedule length while satisfying resource and precedence constraints.

3 Challenges

Developing efficient scheduling strategies for diverse real-time applications in today's safety-critical *CPSs* must meet several challenges. We now enumerate a few such important challenges and discuss them [21].

1. **Timing requirements:**
   Real-time systems are characterized by their operations not only being logically correct, but also on the time at which they are performed. The time before which a task should complete its execution for the safety of the system, is called its deadline. *Scheduling schemes for safety-critical real-time systems must be able to guarantee the timing requirements (i.e., deadlines) associated with various types of tasks that co-exist in the system.*

2. **Resource constraints:**
   Safety-critical systems are implemented on platforms consisting of a limited number of

resources. Providing a lot of redundant hardware is not always possible/feasible as the system's cost increases, and the system's performance may degrade in terms of power/energy dissipation, etc. For example, in cost-sensitive safety-critical systems like cars, a cost differential of even a hundred dollars can make a commercial difference [30–33]. In addition, over the years, the nature of the processing elements used in real-time systems is transformed from uniprocessor to homogeneous multiprocessor platforms to heterogeneous multiprocessor platforms to cater to higher computation demands while adhering to restrictions on power/energy dissipation. *Scheduling schemes for safety-critical real-time systems must be able to effectively utilize available resources of the underlying platform to satisfy the resource constraints associated with the real-time task set.*

3. **Energy Minimization:**
   Energy consumption in real-time systems has become an important issue with the increase in the number of processing elements. Effective energy management is important for battery-powered embedded systems, such as those deployed in autonomous mobile robots, wearable devices, industrial controllers, etc. Recharging or replacing batteries in such systems is not always practical or feasible. Hence, effective energy management can enhance the lifetime of the batteries resulting in higher performance and financial advantages. Even for systems directly connected to the power grid, reducing energy consumption provides significant monetary and environmental gains [34]. Scheduling schemes for real-time systems must optimize the energy consumption satisfying other constraints like timing, resource, precedence, etc.

4 Objectives

The principal aim of this dissertation has been to investigate the theoretical and practical aspects of co-scheduling strategies in safety-critical *CPSs*, keeping in view the challenges/hurdles discussed in the previous section. In particular, the objectives of this work may be summarized as follows:

1. Development of co-scheduling strategies for a set of independent periodic tasks executing on a bus-based homogeneous multiprocessor system, with the objective of maximizing system level *Quality of Service* (*QoS*).

2. Design and implementation of *QoS* adaptive scheduling mechanisms for real-time systems modeled as *PTGs*, on fully-connected heterogeneous multiprocessor system.

3. Development of optimal co-scheduling strategies for *PTGs* executing on a shared-bus based heterogeneous distributed platform.

4. Design of energy-aware processor-bus co-scheduling strategy for the heterogeneous distributed *CPS* platforms, as mentioned above.

5 Summary of work done

As part of this PhD research work, we have developed multiple scheduler design schemes for real-time *CPSs*. The entire thesis work is composed of multiple contributions categorized into four phases, each of which is targeted towards a distinct task/platform scenario.

1. **Task Scheduling on Homogeneous Distributed Systems**
   CPSs, including those in the automotive domain, are often designed by assigning to each task an appropriate criticality-based reward value that is acquired by the system on its successful execution. Additionally, each task may have multiple implementations designated as service-levels, with higher service-levels producing more accurate results and contributing to higher rewards for the system.

   This work proposes co-scheduling strategies for a set of independent periodic tasks executing on a bus-based homogeneous multiprocessor system, with the objective of maximizing system level *QoS*. Each service-level of any task has a distinct computation demand (serviced by one or more processors) and communication demand (serviced by a set of shared buses) with higher service-levels having higher resource demands. Successful execution of a task at a certain service-level is associated with a reward corresponding to that service-level and this reward is proportional to the task's relative importance/criticality. The objective of the task allocation mechanism is to maximize aggregate rewards such that both computation and communication resource demands of all tasks may be feasibly satisfied. The problem is posed as a *Multi-dimensional Multiple-Choice Knapsack formulation* (*MMCKP*) and present a *Dynamic Programming* (*DP*) solution (called *MMCKP-DP*) for the same. Although *DP* delivers optimal solutions, it suffers from significantly high overheads (in terms of running time and main memory consumption) which steeply increase as the number of tasks, service-levels, processors and buses in the system grows. Even for a system with a moderate task set consisting of 90 tasks, it takes approximately 1 hour 20 minutes and consumes a huge amount of main memory space (approximately 68 GB). Such large time and space overheads are often not affordable, especially when multiple quick design iterations are needed during design space exploration and/or powerful server systems are not available at the designer's disposal.

   Therefore, in addition to the optimal solution approach, we propose an efficient but low-overhead heuristic strategy called *ALOLA* (*Accurate Low Overhead Level Allocator*) which consumes drastically lower time and space complexities while generating good and acceptable solutions which do not significantly deviate from the optimal solutions. *ALOLA* is a greedy but balanced heuristic service-level allocation approach that proceeds level by level so that a high aggregate *QoS* may be acquired by the system at much lower complexity compared to the optimal *MMCKP-DP* strategy. The mechanism starts by storing all tasks in a max-heap (based on a key $cost_i$) and assigning base service-levels to all tasks. The algorithm then proceeds by repeatedly extracting the task at the root of the heap, incrementing its service-level by 1, updating its *cost* value and reheapifying it, until residual resources are completely exhausted, or all the tasks have been assigned their maximum possible service-levels.

Our simulation based experimental evaluation shows that even on moderately large systems consisting of 90 tasks with 5 service-levels each, 16 processors and 4 buses, while *MMCKP* incurs a run-time of more than 1 hour 20 minutes and approximately 68 GB main memory, *ALOLA* takes only about 196 μs (speedup of the order of 10^6 times) and less than 1 MB of memory. Moreover, while being fast, *ALOLA* is also efficient being able to control performance degradations to at most 13% compared to the optimal results produced by *MMCKP-DP*. Both the presented solution strategies (*MMCKP-DP* and *ALOLA*) assume *Deadline Partitioning Fair* (*DP-Fair*) [20], a well known optimal multiprocessor scheduler, as the underlying scheduling mechanism.

2. **PTG Scheduling on Heterogeneous Distributed Systems**
Continuous demands for higher performance and reliability within stringent resource budgets is driving a shift from homogeneous to heterogeneous processing platforms for the implementation of today's *CPSs*. These *CPSs* are typically represented as *PTGs* due to the complex interactions between their functional components and are often distributed in nature. This work considers the problem of scheduling a real-time system modeled as *PTG*, where tasks may have multiple implementations designated as service-levels, with higher service-levels producing more accurate results and contributing to higher *rewards/QoS* for the system. In this work, we propose *the design of ILP based optimal scheduling strategies as well as low-overhead heuristic schemes for scheduling a real-time PTG executing on a distributed platform consisting of a set of fully-connected heterogeneous processing elements.*

First, we develop an *ILP* based optimal solution strategy namely, *ILP-SATC* (*ILP - Service-level Allocation with Timed Constraints*), which follows an intuitive design flow and represents all specifications related to resource, timing and dependency, through a systematic set of constraints. However, its scalability is limited primarily due to the explicit manipulation of task mobilities between their earliest and latest start times. In order to improve scalability, a second strategy namely, *ILP-SANC* (*ILP - Service-level Allocation with Non-overlapping Constraints*) has been designed. *ILP-SANC* is based on the *non-overlapping approach* [1] which sets constraints and variables in such a way that no two tasks executing on the same processor overlap in time. Further, in *ILP-SANC* the total number of constraints required to compute a schedule for a *PTG* becomes independent of the number of processors in the platform, which helps to control complexity of the proposed scheme. For example, given a *PTG* with seven tasks, each having two service-levels and executes on a distributed system consisting of 2 heterogeneous processors, *ILP-SATC* generates 7834 constraints and takes $\sim$2 seconds to find the optimal schedule. On the other hand, *ILP-SANC* generates only 203 constraints and takes 0.06 seconds to find the same solutions.

Though *ILP-SANC* shows appreciable improvements in terms of scalability over the *ILP-SATC*, it still suffers from high computational overheads (in terms of running time) as the number of nodes in a *PTG* and/or the number of resources, increase. For example, given a *PTG* with ∼20 tasks, each having three service-levels and executes on a distributed system consisting of 8 heterogeneous processors, *ILP-SANC* takes ∼4 hours to find the optimal schedule. It may be noted that such large time overheads may often not be affordable, especially when multiple quick design iterations are needed during design space exploration. Therefore, two low-overhead heuristics (i) *G-SAQA* (*Global Slack Aware Quality-level Allocator*) and (ii) *T-SAQA* (*Total Slack Aware Quality-level Allocator*) are proposed. Both *G-SAQA* and *T-SAQA* internally make use of *PEFT* [4], a well known *PTG* scheduling algorithm on heterogeneous multiprocessor systems, to compute a baseline schedule which assumes all task nodes to be at their base service-levels. Since *PEFT* attempts to minimize schedule length, the resulting schedule length may be marked by unutilized slack time before deadline.

The *G-SAQA* algorithm starts by using *PEFT* to compute task-to-processor mappings as well as start and finish times of tasks, based on task execution times associated with their base service-levels. If length of the obtained *PEFT* schedule violates deadline, then the algorithm terminates as generation of a feasible schedule is not possible. Otherwise, the available *global slack* ($slack_g = Deadline - PEFT\ makespan$) is used to enhance the tasks' assigned service-levels in an endeavour to maximize achievable reward while retaining task-to-processor mappings as provided by *PEFT*. The enhancement of task service-levels happen in a service-level by service-level manner, starting with all tasks situated at their base service-levels. At each step, the most eligible task is selected (from the task set) for service-level upgradation by one. The selection of this task is based on a prioritization key which is the ratio between gain in rewards and increase in execution time to upgrade service-level from current to the next one.

Though *G-SAQA* follows an intuitive design flow, it only considers global slack ($= Deadline - PEFT\ makespan$) to upgrade service-levels of tasks in the *PTG*. However, a closer look at the *PEFT* schedule reveals that there exists gap within the scheduled nodes of the *PTG* which could be used along with the global slack to achieve better performance in terms of service-levels and delivered rewards compared to *G-SAQA*. It may also be possible to consolidate multiple small gaps within the *PEFT* schedule into larger consolidated slacks which may be used to further improve performance in terms of achieved rewards. Therefore, the *total slack* available with a task at any given time comprises of the global slack along with the maximum consolidated inter-node gap between the task and its successor on its assigned processor in the *PEFT* schedule. With the above insights on the total task-level slacks available in a *PTG*, it proposes another heuristic namely, *T-SAQA* with the objective of achieving better performance compared to *G-SAQA*. The basic structure of *T-SAQA* is same as that of the *G-SAQA* algorithm except the way it updates the start times of selected task node's (selected for service-level enhancement) descendants and slacks associated with the task nodes in the *PTG*. In particular, *G-SAQA* uniformly delays the start times of all descendant nodes of the selected task and reduces the global slack value by the same amount. In this regard, it may be emphasized that *T-SAQA* works with distinct total slack values associated with the task nodes in the *PTG*, instead of using a single global slack pool. By harnessing the total slacks available with individual task nodes, *T-SAQA* updates the start and finish times of only those descendant task nodes of the selected task, whose start times are impacted due to the service-level upgradation of the selected task. Our simulation based experimental results show that both the heuristic schemes (*G-SAQA* and *T-SAQA*) are about $\sim 10^6$ times faster on an average than the optimal strategy *ILP-SANC* when number of tasks in the *PTG* is ∼15, number of service-levels of each task is 3 and number of heterogeneous processors in the system is 8. It also shows that both *T-SAQA* and *G-SAQA* returns at most ∼30% and ∼45% less rewards than *ILP-SANC*, respectively. In all cases *T-SAQA* outperforms *G-SAQA* in terms of rewards maximization while *T-SAQA* has more running time than *G-SAQA*.

3. **PTG Scheduling on Heterogeneous Distributed Shared Bus Systems**
The *PTG* scheduling technique considered in the previous section assumed a fully connected heterogeneous platform. Assumption of a fully connected platform helps avoid the problem of resource contention, as is the case when the system is assumed to be associated with shared data transmission channels. However, it may be appreciated that shared bus networks form a very commonly used communication architecture in *CPSs* [35, 36]. Therefore, this work extend the problem of scheduling *PTGs* on fully-connected platforms, to *CPS* systems where the processors are connected through a limited number of bus based shared communication channels. In this work, we propose *the design of ILP based optimal scheduling strategies as well as low-overhead heuristic schemes for the scheduling of real-time PTGs executing on a distributed platform consisting of a set of heterogeneous processing elements interconnected by heterogeneous shared buses.*

We first develop an *Integer Linear Programming* based solution strategy namely, *ILP-ETR* (*ILP with Explicit Time Reduced*) to produce optimal schedules for real-time *PTGs* executing on a distributed heterogeneous platform. *ILP-ETR* follows a comprehensive design approach which represents all specifications related to resource, timing and dependency, through a systematic set of constraints. Although *ILP-ETR* follows an intuitive design flow, its scalability is limited primarily due to the explicit manipulation of task mobilities between their earliest and latest start times. In order to improve its scalability, we propose an improved *ILP* formulation namely, *ILP-NC* (*ILP with Non-overlapping Constraints*) based on the *non-overlapping approach* [1] which sets constraints and variables in such a way that no two tasks executing on the same processor overlap in time. Experimental results show that *ILP-ETR* takes ∼5 hours to compute the schedule of a *PTG* with ∼20 nodes executing on a system with 4 processor and 2 buses, and the deadline of the *PTG* is set to its optimal makespan. On the other hand, *ILP-NC* takes only ∼12 secs to compute schedule for the same. Again, *ILP-ETR* is unable to find optimal solution within 24 hours when the deadline of the *PTG* is increased by 25% of its optimal makespan. In this case, *ILP-NC* takes only ∼12 seconds to find the optimal solution.

In addition to the two optimal *ILP* based approaches, we have designed a fast and efficient heuristic strategy namely, *CC-TMS* (*Contention Cognizant Task and Message*

Scheduler) for the problem at hand. The *CC-TMS* is based on a *list scheduling* based heuristic approach to co-schedule task and message nodes in a real-time *PTG* executing on a distributed system consisting of a set of heterogeneous processors interconnected by heterogeneous shared buses. The algorithm assigns priority to all nodes in the *PTG* according to a parameter called, *upward rank* of each node. It then selects the highest priority task node and computes the task's *EFT* (*Earliest Finish Time*) values on each processor while temporarily allocating parent message nodes to suitable buses. The task is actually mapped on the processor where it has minimum *EFT*. Based on this selected task-to-processor mapping the parent message nodes of the task node are assigned to suitable buses. This process repeats until all task nodes are scheduled on processors. To evaluate the performance of the *CC-TMS* with respect to optimal solutions, we define a metric called *Makespan Ratio* as follows:

$$Makespan\ Ratio = \frac{Optimal\ Makespan}{Heuristic\ Makespan} \times 100 \tag{1}$$

Extensive simulation based experimental results show that *CC-TMS* achieves 97% and 58% (*Makespan Ratio*) in the best and worst case scenarios, respectively.

4. **Energy-aware Scheduling for Systems Consisting of Multiple PTG Applications**

   The works done in the second and third phases deal with the co-scheduling of a single task graph on heterogeneous distributed platform. In the current phase, we endeavour towards the design of heterogeneous processor-shared bus co-scheduling strategies for a given set of independent periodic applications, each of which is modelled as a *PTG*. In particular, we have developed an *ILP* based optimal and heuristic strategy for the mentioned system model, whose objective is to minimize system level dynamic energy dissipation. Obviously to achieve energy savings, the processors in the system are assumed to be *DVFS* (*Dynamic Voltage and Frequency Scaling*) enabled and thus, the operating frequencies of these processors can be dynamically reconfigured to a discrete set of alternative frequency-levels at run-time. However, the *ILP* based optimal scheme called *ILP-ES* (*ILP for Energy-aware Scheduling*) is associated with very high computational complexity and is not scalable even for small problem sizes. Therefore, we propose an efficient but low-overhead heuristic strategy called *SAFLA* (*Slack Aware Frequency Level Allocator*) which consumes drastically lower time and space complexities while generating good and acceptable solutions.

   The *SAFLA* algorithm starts by using an efficient co-scheduling algorithm *TMC* (*Task and Message Co-scheduling*) which actually extend the *CC-TMS* algorithm (discussed above) to schedule multiple periodic *PTGs* executing on a shared bus-based heterogeneous distributed platform. This schedule is generated assuming all processor to be running at their highest frequency for the entire duration of the schedule. *SAFLA* terminates with failure if the schedule returned by *TMC* violates deadline. Otherwise, the available slack associated with each task node is used to enhance the tasks' assigned frequency-levels in an endeavour to minimize energy dissipation while retaining task-to-processor/message-to-bus mappings as provided by *TMC*. Experimental results show that *SAFLA* is an effective scheduling scheme and delivered handsome energy savings in most practical scenarios.

6 Organization of the Thesis

The thesis is organized into eight chapters. A summary of the contents in each chapter is as follows:

- **Chapter 1**: *Introduction*
  This chapter is introductory, discussing the motivation of our work.

- **Chapter 2**: *Background on Real-time Systems*
  This chapter presents a background on real-time systems and various co-scheduling strategies of real-time tasks and messages on distributed multiprocessor platforms. In particular, we try to present the vocabulary needed to understand the following chapters.

- **Chapter 3**: *Task Scheduling on Homogeneous Distributed Systems*
  In the third chapter, we propose strategies for co-scheduling a set of independent periodic tasks with multiple service-levels, executing on a bus-based homogeneous multiprocessor system. The problem is posed as a *Multi-dimensional Multiple-Choice Knapsack formulation* (*MMCKP*) and present a *Dynamic Programming* (*DP*) solution (called *MMCKP-DP*) for the same. Although *DP* delivers optimal solutions, it suffers from significantly high overheads (in terms of running time and main memory consumption) which steeply increase as the number of tasks, service-levels, processors and buses in the system grows, and severely restricts the scalability of the strategy. Therefore, in addition to the optimal solution approach *MMCKP-DP*, we propose an efficient but low-overhead heuristic strategy called *ALOLA* (*Accurate Low Overhead Level Allocator*) which not only consumes drastically lower time and space complexities but also generate good and acceptable solutions, which do not significantly deviate from the optimal solutions.

- **Chapter 4**: *PTG Scheduling (Optimal) on Heterogeneous Distributed Systems*
  Research conducted in the fourth chapter deals with the optimal scheduling mechanism of a real-time system modeled as *PTG* executing on a fully connected distributed heterogeneous platform. Here, tasks may have multiple implementations designated as service-levels, with higher service-levels producing more accurate results and contributing to higher *rewards/QoS* for the system. To solve the problem, an *ILP* based optimal solution approach, namely, *ILP-SATC*, is proposed. Though the formulation of *ILP-SATC* follows an intuitive design flow, its scalability is limited primarily due to the explicit manipulation of task mobilities between their earliest and latest start times. In order to improve scalability, a second *ILP* based strategy, namely, *ILP-SANC*, has been designed. Instead of explicitly relying on task mobility based manipulations as *ILP-SATC*, *ILP-SANC* guarantees that the executions of no two tasks in the system overlap in time on

the same processor. This modification in the design approach allows the constraint set in *ILP-SANC* to be independent of the number of processors in a platform and the deadline of a given *PTG*.

- **Chapter 5**: *PTG Scheduling (Heuristic) on Heterogeneous Distributed Systems*
  Though *ILP-SANC* in (Chapter 4) shows appreciable improvements in terms of scalability over the *ILP-SATC*, it still suffers from high computational overheads (in terms of running time) as the number of nodes in a *PTG* and/or the number of resources, increase. Therefore in the fifth chapter, two low-overhead heuristic algorithms, namely, *G-SAQA* and *T-SAQA*, are proposed for the same problem as discussed in the previous (fourth) chapter. The base-line heuristic, *G-SAQA*, is faster but returns moderately good solutions. *T-SAQA* extends *G-SAQA* and deliver significantly better solution, albeit at the cost of slightly higher time complexity.

- **Chapter 6**: *PTG Scheduling on Heterogeneous Distributed Shared Bus Systems*
  In this chapter, we propose the design of *ILP* based optimal scheduling strategies as well as low-overhead heuristic schemes for the co-scheduling of real-time *PTGs* executing on a distributed platform consisting of a set of heterogeneous processing elements interconnected by heterogeneous shared buses. To solve the problem, two *ILP* based strategies namely, *ILP-ETR* and *ILP-NC* are proposed. Although, both the approaches produce optimal solutions, *ILP-NC* suffers significantly lower computational overheads compared to *ILP-ETR*. In addition to the optimal solution approaches, we propose a fast but effective heuristic strategy called *CC-TMS* which consumes much lower time and space complexities while producing satisfactorily good solutions.

- **Chapter 7**: *Energy-aware Scheduling for Systems Consisting of Multiple PTG Applications*
  Chapter 7 deals with the energy-aware co-scheduling of multiple periodic *PTGs* executing on a distributed platform consisting of heterogeneous processing elements and interconnected through a set of heterogeneous shared buses. An *ILP* based optimal scheduling strategy is proposed to minimize the overall system-level energy dissipation. Further, an efficient, low-overhead heuristic strategy called *SAFLA* has been proposed for the problem at hand.

- **Chapter 8**: *Conclusion and Future Work*
  The thesis concludes with this chapter. A comparative analysis on the algorithms presented in the different contributory chapters has been carried out. We discuss the possible extensions and future works that can be done in this area.

7 Disseminations out of this Work

Journal Papers

1. Sanjit Kumar Roy, Rajesh Devaraj, Arnab Sarkar, Sayani Sinha and Kankana Maji. "Contention-aware optimal scheduling of real-time precedence-constrained task graphs on heterogeneous distributed systems." *Elsevier Journal of Systems Architecture (JSA).* Volume 105, May 2020, 101706.

2. Sanjit Kumar Roy, Arnab Sarkar and Rahul Gangopadhyay. "Processor and Bus Co-scheduling Strategies for Real-time Tasks with Multiple Service-levels." *Springer Journal of Scheduling (JOSH)*, (Second round major revision received).

3. Sanjit Kumar Roy, Rajesh Devaraj and Arnab Sarkar. "SLAQA: Quality-level Aware Scheduling of Task Graphs on Heterogeneous Distributed Systems." *ACM Transactions on Embedded Computing Systems (ACM TECS)*, (First round review response submitted).

4. Sanjit Kumar Roy, Rajesh Devaraj and Arnab Sarkar. "Contention Cognizant Scheduling of Task Graphs on Shared Bus based Heterogeneous Platforms." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (IEEE TCAD)*, (First round review response submitted).

5. Sanjit Kumar Roy, Rajesh Devaraj and Arnab Sarkar. "Energy-aware Co-scheduling of Multiple Task Graphs on Shared Bus based Heterogeneous Platforms." (Manuscript under preparation).

Conference Papers

1. Sanjit Kumar Roy, Rajesh Devaraj, Arnab Sarkar, Sayani Sinha and Kankana Maji. "Optimal scheduling of precedence-constrained task graphs on heterogeneous distributed systems with shared buses." *IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC).* Pages 185-192, 2019.

2. Sanjit Kumar Roy, Rajesh Devaraj and Arnab Sarkar. "Optimal scheduling of PTGs with multiple service levels on heterogeneous distributed systems." *American Control Conference (ACC).* Pages 157-162, 2019.

References

[1] S. Venugopalan and O. Sinnen, "ILP formulations for optimal task scheduling with communication delays on parallel systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 1, pp. 142–151, 2015. [Pg.1], [Pg.3], [Pg.6], [Pg.8]

[2] "CPLEX Optimizer: https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer." [Online]. Available: https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer [Pg.1]

[3] H. Topcuoglu *et al.*, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems*, vol. 13, no. 3, pp. 260–274, 2002. [Pg.2], [Pg.3]

[4] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 682–694, 2014. [Pg.2], [Pg.3], [Pg.7]

[5] G. Xie, R. Li, and K. Li, "Heterogeneity-driven end-to-end synchronized scheduling for precedence constrained tasks and messages on networked embedded systems," *Journal of Parallel and Distributed Computing*, vol. 83, pp. 1–12, 2015. [Pg.2], [Pg.3]

[6] R. Bajaj and D. P. Agrawal, "Improving scheduling of tasks in a heterogeneous environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 2, pp. 107–118, 2004. [Pg.2], [Pg.3]

[7] S. Bansal, P. Kumar, and K. Singh, "Dealing with heterogeneity through limited duplication for scheduling precedence constrained task graphs," *Journal of Parallel and Distributed Computing*, vol. 65, no. 4, pp. 479–491, 2005. [Pg.2], [Pg.3]

[8] R. I. Davis and A. Burns, "A survey of hard real-time scheduling for multiprocessor systems," *ACM Computing Surveys (CSUR)*, vol. 43, no. 4, p. 35, 2011. [Pg.2]

[9] H. Baek, J. Lee, and I. Shin, "Multi-level contention-free policy for real-time multiprocessor scheduling," *Journal of Systems and Software*, vol. 137, pp. 36–49, 2018. [Pg.2]

[10] H. S. Chwa, H. Back, J. Lee, K.-M. Phan, and I. Shin, “Capturing urgency and parallelism using quasi-deadlines for real-time multiprocessor scheduling,” *Journal of Systems and Software*, vol. 101, pp. 15–29, 2015. [Pg.2]

[11] C. L. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard-real-time environment,” *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973. [Pg.2]

[12] B. Andersson and J. Jonsson, “The utilization bounds of partitioned and pfair static-priority scheduling on multiprocessors are 50%,” in *Real-Time Systems, 2003. Proceedings. 15th Euromicro Conference on.* IEEE, 2003, pp. 33–40. [Pg.2]

[13] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, “Proportionate progress: A notion of fairness in resource allocation,” *Algorithmica*, vol. 15, no. 6, pp. 600–625, 1996. [Pg.2]

[14] J. H. Anderson and A. Srinivasan, “Mixed Pfair/ERfair scheduling of asynchronous periodic tasks,” in *Real-Time Systems, 13th Euromicro Conference on, 2001.* IEEE, 2001, pp. 76–85. [Pg.2]

[15] ——, “Early-release fair scheduling,” in *Proceedings 12th Euromicro Conference on Real-Time Systems. Euromicro RTS 2000.* IEEE, 2000, pp. 35–43. [Pg.2]

[16] D. Zhu, D. Mossé, and R. Melhem, “Multiple-resource periodic scheduling problem: how much fairness is necessary?” in *RTSS 2003. 24th IEEE Real-Time Systems Symposium, 2003.* IEEE, 2003, pp. 142–151. [Pg.2]

[17] A. Khemka and R. Shyamasundar, “An optimal multiprocessor real-time scheduling algorithm,” *Journal of parallel and distributed computing*, vol. 43, no. 1, pp. 37–45, 1997. [Pg.2]

[18] H. Cho, B. Ravindran, and E. D. Jensen, “An optimal real-time scheduling algorithm for multiprocessors,” in *2006 27th IEEE International Real-Time Systems Symposium (RTSS’06).* IEEE, 2006, pp. 101–110. [Pg.2]

[19] P. Regnier, G. Lima, E. Massa, G. Levin, and S. Brandt, “Run: Optimal multiprocessor real-time scheduling via reduction to uniprocessor,” in *2011 IEEE 32nd Real-Time Systems Symposium.* IEEE, 2011, pp. 104–115. [Pg.2]

[20] G. Levin, S. Funk, C. Sadowski, I. Pye, and S. Brandt, “DP-FAIR: A simple model for understanding optimal multiprocessor scheduling,” in *Real-Time Systems (ECRTS), 2010 22nd Euromicro Conference on.* IEEE, 2010, pp. 3–13. [Pg.2], [Pg.6]

[21] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications.* Springer, 2011, vol. 24. [Pg.2], [Pg.3]

[22] X. Wang, Z. Li, and W. M. Wonham, “Dynamic multiple-period reconfiguration of real-time scheduling based on timed DES supervisory control,” *IEEE Transactions on Industrial Informatics*, vol. 12, no. 1, pp. 101–111, 2016. [Pg.2]

[23] G. Srinivasa Prasanna and B. R. Musicus, “Generalised multiprocessor scheduling using optimal control,” in *3rd annual symposium on Parallel algorithms and architectures.* ACM, 1991, pp. 216–228. [Pg.2]

[24] G. S. Prasanna and B. R. Musicus, “Generalized multiprocessor scheduling and applications to matrix computations,” *IEEE Transactions on Parallel and Distributed systems*, vol. 7, no. 6, pp. 650–664, 1996. [Pg.3]

[25] J. Liu, Q. Zhuge, S. Gu, J. Hu, G. Zhu, and E. H.-M. Sha, “Minimizing system cost with efficient task assignment on heterogeneous multicore processors considering time constraint,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2101–2113, 2014. [Pg.3]

[26] H. Kanemitsu, M. Hanada, and H. Nakazato, “Clustering-based task scheduling in a large number of heterogeneous processors,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 11, pp. 3144–3157, 2016. [Pg.3]

[27] P.-C. Hsiu, C.-K. Hsieh, D.-N. Lee, and T.-W. Kuo, “Multilayer bus optimization for real-time embedded systems,” *IEEE Transactions on Computers*, vol. 61, no. 11, pp. 1638–1650, 2012. [Pg.3]

[28] M.-Y. Wu and D. D. Gajski, “Hypertool: A programming aid for message-passing systems,” *IEEE transactions on parallel and distributed systems*, vol. 1, no. 3, pp. 330–343, 1990. [Pg.3]

[29] T. C. Hu, “Parallel sequencing and assembly line problems,” *Operations research*, vol. 9, no. 6, pp. 841–848, 1961. [Pg.3]

[30] C. Krishna, “Fault-tolerant scheduling in homogeneous real-time systems,” *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 48, 2014. [Pg.4]

[31] M. Chetto, “Optimal scheduling for real-time jobs in energy harvesting computing systems,” *IEEE Trans. Emerging Topics Comput.*, vol. 2, no. 2, pp. 122–133, 2014. [Online]. Available: https://doi.org/10.1109/TETC.2013.2296537 [Pg.4]

[32] G. Raravi, B. Andersson, V. Nélis, and K. Bletsas, “Task assignment algorithms for two-type heterogeneous multiprocessors,” *Real-Time Systems*, vol. 50, no. 1, pp. 87–141, 2014. [Pg.4]

[33] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou, and Y. Halgand, “A survey of approaches combining safety and security for industrial control systems,” *Reliability Engineering & System Safety*, vol. 139, pp. 156–178, 2015. [Pg.4]

[34] M. Bambagini, M. Marinoni, H. Aydin, and G. Buttazzo, "Energy-aware scheduling for real-time systems: A survey," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 15, no. 1, pp. 1–34, 2016. [Pg.4]

[35] S. Chakraborty, M. A. Al Faruque, W. Chang, D. Goswami, M. Wolf, and Q. Zhu, "Automotive cyber–physical systems: A tutorial introduction," *IEEE Design & Test*, vol. 33, no. 4, pp. 92–108, 2016. [Pg.8]

[36] T. Mitra, J. Teich, and L. Thiele, "Time-critical systems design: A survey," *IEEE Design & Test*, vol. 35, no. 2, pp. 8–26, 2018. [Pg.8]