Journal of Systems Architecture 000 (2016) 1–16

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Reservation based protocol for resolving priority inversions in composable conveyor systems

Abdelrhman Mahamadi*, Mukesh Chippa, Shivakumar Sastry

Complex Engineered System Lab, ECE Department at the University of Akron, Ohio, USA

ARTICLE INFO

Article history:
Received 10 April 2016
Revised 3 September 2016
Accepted 21 November 2016
Available online xxx

Keywords:
Priority inversion
Networked embedded systems
Advanced manufacturing

ABSTRACT

We present a reservation based protocol for resolving priority inversions in Composable Conveyor Systems. These systems represent a class of networked multi-processor systems that are used to physically transport entities from inputs to outputs. The absence of shared memory and the interaction between the cyber and physical subsystems present many challenges such as priority inversion. We view the end-to-end transport of an entity as a task and discuss how these systems admit a rich set of task models. Like in other real-time systems, a priority inversion is said to occur when a high-priority task is blocked by a lower-priority task for an unbounded duration of time. We present an approach to compute the average waiting time for entities, establish properties of the proposed protocol and present simulation results that demonstrate the efficacy of the proposed protocol. In the future, this protocol can be extended to other task models and a larger class of decentralized systems for advanced manufacturing.

1. Introduction

Recent advances in computing and communication technologies offer new platforms for the control and regulation of manufacturing systems [1,4]. Despite the rapid adoption of early digital communications and distributed systems technologies into automation and manufacturing systems, concerns such as the large installed base, cost and safety have created an inertia for change [31]. Over the last few years, wireless communications technologies [13] and new decentralized platforms for regulating automation systems have been explored [4]. The integration of fine-grained cyber capabilities at multiple levels of traditional automation systems offers new advantages such as flexible reconfiguration to address emerging market demands, energy efficiency, and scalable performance. This decentralized, reconfigurable architecture when coupled with the tight integration between cyber and physical activities exposes several fundamental issues that must be revisited and addressed to ensure safe and robust operational environments.

Composable Conveyor Systems (CCS) are well-structured systems that represent a class of networked embedded systems [28]. These systems are asynchronous, decentralized, systems that do not have shared memory. The nodes can interact over wireless links and by adjusting the transmission power in each node, one can achieve spatial separation between subsets of the nodes. In order to synchronize state or time, the nodes can execute multihop protocols over the wireless links. CCS can also be viewed as a networked system that moves physical entities from inputs to outputs. In this view, the interactions between the nodes occur at multiple time scales – for example, the physical movement of entities requires several seconds whereas the communications require several milliseconds; the topology of the system may be reconfigured once in a few hours whereas faults may occur once in a few days. The main objective of CCS is to transport physical entities from inputs to outputs. The physical entities present several constraints - for example, once a sequence of entities are on the conveyor units, they cannot be rearranged. These systems are attractive because they capture many of the spatio-temporal interactions that occur in a variety of material handling applications; at the same time, CCS units have simple behaviors that are pre-programmed. The topology is flexible and the system can be composed or reconfigured in response to demand.

Each unit is regulated by an autonomous microcontroller that interacts with its neighboring units over low-power wireless links. These conveyor systems are composed using two kinds of units called *Segments* and *Turns*. These systems can be dynamically reconfigured to assure real-time Quality of Service (QoS), i.e., *end-to-end latency, jitter* and *throughput*, in operational theaters such as warehouses, manufacturing lines, package sorting facilities, or front line logistics for future military deployments. Tasks in CCS, which involve the end-to-end transport of an entity in the system, evolve simultaneously both in time and space. The insights gained from

* Corresponding author.
E-mail address: abodi6767@hotmail.com (A. Mahamadi).

http://dx.doi.org/10.1016/j.sysarc.2016.11.008
1383-7621/

such a study can be applied to a larger class of cyber-physical systems [3,28].

The notion of priority arises in CCS because of externally imposed deadline requirements. For example, in a package sorting application, packages may have service requirements such as "next morning delivery", "two-day delivery" or "ground delivery" [28]. Entities with such requirements enter the CCS from some inputs and must be moved to some output along paths, i.e., a sequence of physically adjacent conveying units. To improve connectivity, utilization, and resilience of the CCS, the topology is designed to ensure that such paths overlap. One consequence of such overlaps is that when two paths merge at a unit, lower-priority entities that are moving along one path can block higher-priority entities on other intersecting paths.

As is well-known in the real-time systems literature, lower-priority tasks may in fact be serviced before higher-priority tasks for a *bounded* duration of time whenever there is a need to share common resources. However, lower-priority tasks must not be able to block higher-priority tasks for an *unbounded duration of time*. This is the classical priority inversion problem [29]; in centralized, real-time systems, this problem has been solved using classical protocols such as the Priority Inheritance Protocol (PIP). Certain problems that remained unresolved when using PIP were solved using the Priority Ceiling Protocol (PCP) and the Stack Resource Protocol (SRP) [10]. When priority inversions occur in CCS, it fundamentally affects how the entities with specific priorities are handled by the system. A higher-priority entities might suffer unbounded delays as we will demonstrate in Section 3. When the CCS topology and the arrival distribution of the entities are known in advance, it may be possible to schedule the activities carefully to minimize priority inversions. However, when the topology can change dynamically and/or the arrival distributions are not known in advance, we need dynamic online protocols to avoid and mitigate the effects of priority inversions. Since the units of CCS are modular and autonomous, such a protocol must be decentralized, adaptable and scalable.

In our prior work [28], we presented how priority inversions manifest in CCS. The end-to-end transfer of an entity through the CCS was viewed as a task and we showed that when all the entities that enter the CCS via the same input have the same priority, PIP could be effectively adapted to resolve priority inversions. We also presented simulation results to demonstrate the impact of priority inversions and established the need for new protocols to fully resolve priority inversions in CCS. In this paper, we extend this prior work along several dimensions. We show that when entities with different priorities enter the CCS from the same Input, the priorities of all entities waiting to be serviced at a turn can become elevated to the highest priority among the entities that have been admitted to the CCS, when PIP is used. Further, as is already well-known for centralized real-time systems, we shown that when PIP is used in CCS, both deadlocks and chained blocking occur. However, the classical solutions for addressing these issues in centralized systems – PCP and SRP, cannot be used in CCS. This is because the ceiling priorities that are computed in both these protocols cause the priorities of all the entities in the CCS to be elevated to the highest priority. Thus, to effectively resolve priority inversions in CCS, we propose a new Reservation Based Protocol (RBP) that does not saturate the priorities of the entities and avoids both deadlocks and chained blocking. Finally, we present an approach to compute the average end-to-end waiting time (AEWT) for entities in the CCS in when using RBP. The new simulation results demonstrate the efficacy of RBP and the validity of AEWT.

The remainder of this paper is organized as follows. After reviewing the background and related work in resolving priority inversions in Section 2, we describe the CCS precisely and discuss two task models in Section 3. In Section 4 we describe the classical resource sharing protocols, their adaptations to CCS, and present the new reservation based protocol in Section 5.3. We analyze these protocols and determine the average expected waiting time for entities from a particular input with a given priority in Section 6. We present simulation results in Section 7 and our conclusions in Section 8.

2. Related work

The priority inversion problem has been studied extensively in areas such as real-time systems, multiprocessor systems and material handling systems.

2.1. Priority inversion in real-time systems

Classical protocols for resolving priority inversions are the Priority Inheritance Protocol (PIP), the Priority Ceiling Protocol (PCP) [29] and the Stack Resource Protocol (SRP) [6]. In PIP, priority inversions are resolved by temporarily elevating the priorities of the tasks that hold a resource by inheriting the (higher) priority of a task that is requesting access to the same resource. In PCP and SRP, the highest ceiling among all possible tasks that can use a resource is used to determine whether or not to accept a task into the system; once accepted, a task is guaranteed to complete execution with no further blocks. These protocols were designed for centralized real-time systems.

Priority inversions in a multistage packet switching network were resolved using PIP [33]. This network was used to route message packets from N inputs to N outputs along fixed paths that each contain $\log_2(N)$ routers. Each router had two in-ports, two out-ports, and capacity to hold a single packet. Before sending packets, each router that had a packet sent a request to transfer the packet to the next router along its path. Every router accepted only the highest priority packet from its in-ports. The restricted topology of the network limits the applicability of this technique in CCS. Flow control techniques [11] are not directly applicable because the routes over which entities move are not always known in advance and there are no buffers in the CCS to store and forward entities.

A reservation based approach was proposed to resolve priority inversions in a centralized real-time system with shared resources [5]. Here, a task requested a reservation and waited for the resource to be granted before using the resource. This mechanism involved two messages: ($hold_j r(t)$) and ($next_j r(t)$). The *hold* message indicated the time for which the resource would be held after gaining access and the *next* message indicated the minimum time before the resource would be requested next. They proposed two reservation policies: first, a requesting task was granted access to resource r if $r(t)$ was less than or equal to the start of the next reservation, i.e., $next_k r(t)$, of all higher-priority tasks; in the second policy, a request was granted only when $hold_j r(t)$ was less than or equal to the next reservation for all higher-priority tasks, for any resource (i.e., not just r) in the system. In order to adopt these protocols for CCS, it is necessary to accurately estimate the time at which an entity will arrive at a downstream turn along its path. Such an estimate is difficult because the priorities of the entities arriving at all other inputs are not known; further, statistical methods used to estimate future times must be updated when there is a change in the topology of the CCS.

Another resource reservation protocol that shared resources over the Internet was proposed in [36]. Here, the sender started by sending a discovery message to the receiver. The receiver sent reservation requests to all the switches in that route with a specified bandwidth. If the reservations were confirmed by the switches, the route was confirmed to the sender and remained active until the sender terminated the reservation. This idea of

Please cite this article as: A. Mahamadi et al., Reservation based protocol for resolving priority inversions in composable conveyor systems, Journal of Systems Architecture (2016), http://dx.doi.org/10.1016/j.sysarc.2016.11.008