Journal of Systems Architecture 60 (2014) 509–518

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

Resolving priority inversions in composable conveyor systems

Shivakumar Sastry [a,*], Aniruddha Gokhale [b]

[a] Department of Electrical and Computer Engineering, University of Akron, Akron, OH 44325-3904, USA
[b] Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN 37325, USA

ARTICLE INFO

Article history:
Received 17 September 2013
Received in revised form 4 February 2014
Accepted 25 February 2014
Available online 7 March 2014

Keywords:
Real-time systems
Priority inversion
Composable conveyors
Networked systems

ABSTRACT

The well known problem of priority inversions that occurs in classical real-time systems also manifests in decentralized cyber-physical systems. Using a specific example of composable conveyor systems, we show how priority inversions hinder the transport of entities through the conveyor systems. We present a novel adaptation of the classical priority inheritance protocol for resolving these cyber-physical priority inversions. While the approach resolves cyber-physical priority inversions, the structure and constraints of the conveyor systems cause the jitter associated with the end-to-end latency of the highest priority parts to increase. Further, these constraints also limit the applicability of the classical priority ceiling protocol in this class of cyber-physical systems. Simulation results demonstrate the correctness and reasonable communication overhead of the approach.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

Composable, or reconfigurable, conveyor systems (CCS) are representative cyber-physical systems that capture the spatio-temporal interactions that can occur in a variety of emerging automation systems, particularly in the area of material handling. These cyber-physical systems are flexibile, easy to use, and can be dynamically reconfigured to assure real-time Quality of Service (QoS) in operational theaters such as warehouses, manufacturing lines, package sorting facilities (e.g., FedEx and UPS), or front-line logistics for future military deployments. CCS are composed using basic building blocks that have pre-defined behaviors. These systems are interesting to study because tasks, which involve the end-to-end transport of an entity in the system, evolve simultaneously both in time and space. Since these systems are well-structured, the desired behaviors and the unintended consequences that result from spatio-temporal interactions between tasks can be studied in a systematic manner. The insights gained from such a study can be applied to a large-class of cyber-physical systems.

Realizing CCS in practice is challenging. The intertwined dynamics of the cyber-physical elements, e.g., the logic embedded in individual micro-controllers of CCS units, the wireless transceivers and protocols for messaging and coordination between the micro-controllers in physically adjacent units, and spatio-temporal evolution of the entities along end-to-end paths in the conveyor system present formidable challenges for addressing several design and operational issues. For example, a CCS designer may want to understand if a particular topology can yield a desired QoS without actually having to deploy and test the system; the designer may want to understand the robustness or resilience of the topology with respect to one or more failures. Our recent work [1,2] addressed some of these questions. We developed intuitive abstractions and analysis tools to enable CCS designers to experiment with different layouts and analyze the QoS that could be achieved from the topology. In this paper, we address the issue of priority inversions that can occur in CCS because of the spatio-temporal interactions between tasks.

Consider a new conveyor system that must be designed to sort packages based on their service category. For example, "next morning delivery," "next afternoon delivery," or "ground delivery" are typical categories that are commonly used. This requirement imposes a notion of "priority" on the packages that are handled by the sorting system. Packages with different priorities arrive via Inputs (sources) and these packages move along paths to some Output. The paths of the conveyor system are formed by a sequence of physically adjacent units called Segments. Each Segment unit comprises a belt that can move the entity from one end to the other. One or more Segments in the system can also be incident with Turn units. The Turn units can merge multiple upstream paths to a single downstream path; alternatively, a Turn unit can also fork a single upstream path to multiple downstream paths. To improve the utilization of the units and resilience of the topology, it is necessary for many of these Input–Output paths to overlap. One consequence of such overlaps is that when two paths

* Corresponding author.
E-mail addresses: ssastry@uakron.edu (S. Sastry), a.gokhale@vanderbilt.edu (A. Gokhale).

http://dx.doi.org/10.1016/j.sysarc.2014.02.003
1383-7621/© 2014 Elsevier B.V. All rights reserved.

merge, low priority packages that are moving along one path can block higher priority packages that need to use the same path for an unbounded duration of time – thus resulting in a classical priority inversion [3]. Although the cyber-physical priority inversion problem may not cause significant disruptions in a package sorting facility, it will be a significant problem for assembly plants where parts much reach their designated positions in a timely manner.

The priority inversion described above is a cyber-physical phenomenon for the following reasons. As will be explained more precisely in Section 2.2, the cyber-physical priority inversion occurs because of both the physical topology of the conveyor system and the cyber decisions that impact the flow of entities on the system. In fact, the cyber-physical priority inversions are caused by an unfortunate consequence of the temporal sequence of priorities of the arriving entities, the physical location of the Inputs where the entities arrive on the system, the physical topology of the conveyor system, and the temporal sequence of routing decisions made at the different Turn units of the system. Such cyber-physical priority inversions are further exacerbated when the system is dynamically reconfigured because small changes to the structure can significantly impact the QoS that can be achieved using the conveyor system.

Classical techniques for resolving cyber-level priority inversions in centralized (cyber-only) real-time systems are well-known. The priority inheritance protocol (PIP) and priority ceiling protocol (PCP) in [3] and the stack resource policy in [4] are excellent solutions. In this paper we show that PIP can be effectively adapted to resolve the cyber-physical priority inversions in CCS; on the other hand, PCP cannot be readily adapted to address the problem.

The rest of the paper is organized as follows: Section 2 presents the problem statement more formally and surveys related work. We discuss the adaptation of PIP in Section 3. We show why PCP cannot be readily adapted in Section 4. Our results and discussion are in Section 5 and we conclude in Section 6.

2. Problem statement and related work

We now illustrate more formally how the cyber-physical priority inversion problem occurs and briefly describe the related work. To better understand the problem statement, we first provide the model of composable conveyors we assume in this work.

2.1. Model of composable conveyor systems

The conveyor systems we consider move entities from inputs ($\mathcal{I}$) to outputs ($\mathcal{O}$). These systems are composed using two kinds of units – *Segments* ($\mathcal{S}$) and *Turns* ($\mathcal{X}$) – that have fixed behaviors [5]. Each unit is autonomously regulated by a local microcontroller that interacts with adjacent units over wireless links to coordinate the transfer of entities. A Segment moves an entity over a fixed distance, in one of two assigned directions. Input and Output units are Segments that can move entities only in one direction. A Turn has four ports that can be configured to either bring in entities or remove entities. We assume that units can handle only one entity at a time. When two or more entities simultaneously arrive at the input ports of a Turn, only one entity is accepted by the Turn. We assume that a Turn will not accept an entity only when it is accepting a different entity with a higher priority. The entity that is not accepted for transfer must wait on the unit until it is accepted for transfer; such a wait will propagate to further upstream units because each unit can only hold a fixed number of entities.

We assume that Segments and Turns have a fixed direction that remains unchanged and there are no failures. The underlying directed graph of the system is acyclic. All units in the system can handle at most one entity; however, a unit can simultaneously *transfer-in* and *transfer-out* an entity. There are adequate buffers at the inputs to hold entities that are not yet admitted to the system. Recall, that multiple paths along which entities move overlap at one or more conveyor units. A physical entity that is already on a unit cannot be "preempted." In addition, the sequence of entities on adjacent Segments of the conveyor system cannot be physically reordered. These three reasons collectively cause priority inversions to occur as we explain below. Such inversions are inevitable in systems where resources must be preferentially allocated to tasks. It is, however, important to ensure that such inversion does not occur for an unbounded duration of time.

A conveyor system can be represented as a directed graph $G = (U, E)$. The nodes of $G, u_i \in U$ represent the units, *i.e.*, Segments, Turns, Inputs, and Outputs. An edge $(u_i, u_j) \in E$ represents the relation that an entity can move from unit u_i to unit u_j. Entities that arrive via input $I_k \in \mathcal{I}$ are delivered to a specific output $O_j \in \mathcal{O}$ along a path

$$P(I_k, O_j) =< u_1 = I_k, u_2, \ldots, u_n = O_j >$$

where $u_i \in U$. Such paths can either be pre-computed when the system cannot be reconfigured, or discovered and maintained when the system is reconfigurable using standard shortest path algorithms [6]. Since the paths merge at Turns, some of the paths may overlap and share common units.

2.2. Cyber-physical priority inversion scenarios in composable conveyor systems

Consider the conveyor system shown in Fig. 1. Here, three entities, namely τ_1^a, τ_2^b, and τ_3^c, have arrived via input I_1, I_2 and I_3, respectively. Suppose Segments S_1 and S_2 simultaneously send a request to Turn X_2 to transfer an entity. Assuming that the entities arriving via lower numbered inputs have higher priorities, i.e., $prio(I_2) > prio(I_3)$, X_2 must accept τ_2^b. Because S_1 cannot accept a new entity until its current entity is transferred out, the higher priority entity, τ_1^a, on X_1 is blocked by the lower priority entity, τ_3^c, on S_1. This blocking is an expected consequence of using priorities in the system. However, since the number of entities that can arrive via input I_2 is not bounded, priority inversion (i.e., blocking for unbounded time) can occur.

Using the definition of priority inversion in [3], we can infer that a cyber-physical priority inversion occurs whenever a high priority entity stream is intercepted by a low priority entity stream and this latter stream is intercepted by a medium priority entity stream. Fig. 2 illustrates one such scenario.

At unit $X_m, m > 3$, the low priority entity stream from I_m intercepts the entity stream from I_1. Further downstream at X_3, the entity stream from I_3 intercepts the entity stream from I_m. In this example, the streams from I_m and I_1 are merged after X_m. Since $prio(I_3) > prio(I_m)$, entities from I_m can be blocked by entities from I_3 at unit X_3. When the units along the path $P(I_m, X_3)$ each have

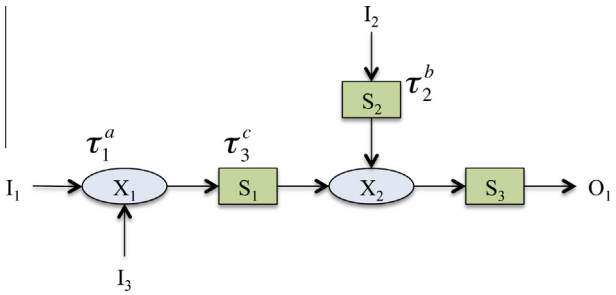

Fig. 1. When multiple entity streams, each with a different priority, share common conveyor units, priority inversion can occur.

Download English Version:

https://daneshyari.com/en/article/457621

Download Persian Version:

https://daneshyari.com/article/457621

Daneshyari.com