Keywords: Cyber-physical systems; Resource allocation; Mutual exclusion;

Abstract: Distributed computing problems such as mutual exclusion have been studied extensively for traditional distributed systems. In traditional systems, a strict layered approach is taken wherein a set of users (application processes) $U_1, \ldots, U_n$ is layered on top of a mutual exclusion algorithm with processes $P_1, \ldots, P_n$. User $U_i$ interacts with process $P_i$ to request access to resources which are modeled as tokens, and users rely entirely on mutual exclusion algorithm to regulate access to the resources. In a cyber-physical system, users (physical entities) may themselves possess capabilities such as sensing, observing and mobility using which they may also attempt to locate physical resources such as wheelchairs. Thus, a mutual exclusion algorithm in a cyber-physical system must contend with the behavior of users. This paper proposes a graph-based model for cyber-physical systems which is used to describe mutual exclusion algorithm as well as user behavior. Based on this model, we present several solutions for the mutual exclusion problem. We have also conducted an extensive simulation study of our algorithms using OMNeT++ discrete event simulation system.

1 INTRODUCTION

Cyber-physical systems are often formed as a result of existing physical systems being instrumented with cyber-infrastructure with the intention of aiding tasks which are being accomplished by traditional (perhaps manual) techniques. For example, consider a health care facility where users (e.g., hospital staff) share resources (e.g., wheelchairs, IV pumps) located in different parts of the facility. This facility may be using established traditional techniques to locate and share resources (e.g., depositing free resources at a central or a set of known locations). However, for more efficient operation, the resources can be instrumented with sensing devices to track their location and usage, and the information made available to potential users. (Wieland et al., 2007) describes a similar smart factory environment where context data (location and usage) regarding tools, machines, transport carts, and spare parts is made available via RFID tags to aid in locating the nearest tools/machines available to do a task. Similar systems have been discussed in various contexts such as locating empty spaces in parking lots (Chinrungrueng et al., 2007), room reservation in buildings (Conner et al., 2004) and smart building operations (Liu et al., 2010).

One central issue in many application scenarios such as discussed above is the use of resources in an exclusive manner. In this paper, we study the problem of mutual exclusion in cyber-physical systems where users need exclusive access to physical resources. In a traditional distributed system (TDS), a mutual exclusion algorithm is typically modeled as a set of processes $P_1, \ldots, P_n$, where $P_i$ executes on node $V_i$, and a strict layered structure is used wherein user $U_i$ interacts with $P_i$ to gain access to a resource. The access of the resources in a TDS is completely regulated by the mutual exclusion algorithm.

In a cyber-physical system (CPS), the cyber-infrastructure may have been superimposed on an existing physical system (Figure 1). In such a case, a distributed mutual exclusion algorithm executing in the cyber-infrastructure must operate in the context of existing techniques being used to locate resources in the physical system (such combined use of cyber and physical techniques may indeed be more efficient...
as shown later by our experiments). In such cases, a distributed algorithm may have to contend with direct interactions between the users and the resources. This introduces several aspects in the context of the mutual exclusion problem which are not addressed in a TDS. First, the users may not be passive entities—that is, in addition to requesting the cyber-infrastructure to locate resources, the users may actively look for resources on their own. For example, in Figure 1, in addition to asking \( P_1 \) to locate a resource, if \( U_1 \) observes that resource \( R_1 \) is available, it may acquire \( R_1 \) without waiting for a response from \( P_1 \), something \( U_1 \) would have done if it was a purely physical system. Second, in a TDS, the state of a resource is controlled by mutual exclusion algorithm. In a CPS, however, the users may independently observe and change the state of the resources. This, for instance, may cause scenarios wherein a user, say \( P_1 \) in Figure 1, may start using \( R_1 \) even though mutual exclusion algorithm may think that it is free and may reserve it for another user (as there may not be a way to "lock" a physical resource). Third, physical resources may be mobile so that they may be acquired at one location and released at another (e.g., a wheelchair may be freed at a different location). This is different from the view taken in a TDS where a resource (e.g., abstracted as a token) is released by the user at the same node where it was acquired.

We find that the aspects in a CPS discussed above can have a significant impact on the design of mutual exclusion algorithms. In (Cheriton and Skeen, 1983), interactions between user entities were attributed to "hidden channels" or channels external to the system and possible solutions were proposed which involve the user entities providing additional information such as timestamps. In cyber-physical systems, however, user entities may be unable participate in such timestamping algorithms as they represent actual physical objects. The event model proposed in (Tan et al., 2009) captures interactions between the physical system and cyber-infrastructure and defines events based on temporal and spatial attributes, and is a step in providing a theoretical basis to develop CPS distributed algorithms. Several other problems such as distributed algorithms for creating globals states in intelligent construction sites (Rajamani and Julien, 2010), event ordering (Romer, 2003; Kaveti et al., 2009) and termination detection (Bapat and Arora, 2008; Kurian et al., 2009) have been studied for CPSs. Although existing research discussed above has addressed some aspects of interactions between cyber-infrastructure and the users, the problem in the context of mutual exclusion has not been addressed. Depending on the interactions between the cyber-infrastructure and the users, there is a range of possible solutions for the mutual exclusion problem in a CPS. On one end, the users may ignore the cyber-infrastructure and address the problem on their own by physically locating the resources (by actions of walking and observing). At the other extreme, one can follow the TDS approach wherein the users ask the cyber-infrastructure to locate resources, and use them only as directed by the cyber-infrastructure. In between these extremes, we can have an array of solutions depending on the cooperation between the users and the cyber-infrastructure. The contribution of this paper is three-fold:

- We propose a model which views a CPS as a triple \( (\text{CyS}, \text{PhyS}, \text{Int}) \), where \( \text{CyS} \) models the cyber-infrastructure superimposed on physical system modeled as \( \text{PhyS} \), and \( \text{Int} \) captures the interactions between them. We call \( \text{CyS} \) and \( \text{PhyS} \) cyber-subsystem and physical-subsystem respectively.
- We propose a set of algorithms based on the proposed model for the mutual exclusion problem which accommodate different behaviors of the users. Each algorithm has two components, one describing the behavior of the users in \( \text{PhyS} \) and the other describing the mutual exclusion algorithm (or cyber algorithm) in \( \text{CyS} \). Each combination of user behavior and cyber algorithm yields a different CPS algorithm.
- We have conducted an extensive simulation study of proposed algorithms using OMNeT++ (Varga, 2001) which simulates both user behavior and cyber algorithm. We have studied the impact of various factors such as the observation capabilities of the users, frequency of sensing, and behavior of the users on the time to acquire a resource.

This paper is organized as follows. Section 2 discusses a model of TDSs and Section 3 presents the extension for a CPS. Section 4 discusses solutions to the mutual exclusion problem for CPSs. Section 5 discusses simulation and results, and Section 6 concludes the paper.

2 Traditional Distributed Systems

A traditional distributed system (TDS) is modeled as a graph \( G_C = (CE, E) \), where \( CE \) is a set of cyber entities (computing platforms) and \( E \) is a set of edges (or communication links) \( E_{ij} \) between two cyber entities \( V_i \) and \( V_j \) (Chandy and Lamport, 1985). Each \( V \in CE \) has a set of processes, denoted by \( V, \text{processes} \in CP \), running on it. Processes executing on cyber entities

...
communicate via the communication links to interact with each other.

A mutual exclusion algorithm for a TDS typically models a physical resource (e.g., a printer) as an abstract object and provides users with an interface with functions to request, acquire and release a resource, and ensures that at most one user is granted access to a resource at a time. Mutual exclusion algorithms have been studied extensively (Dijkstra, 1965; Dijkstra, 1971; Lynch, 1980; Reif and Spirakis, 1982) for both shared memory and message passing systems. In the more general $k$-mutual exclusion problem, at most $k$ processes are allowed to be in critical section at the same time. Two main approaches have been studied to address the distributed $k$-mutual exclusion problem. In the token based approach (Bulgannawar and Vaidya, 1995; Makki et al., 1992; Srimani and Reddy, 1992; Walter et al., 2001), a process is allowed to enter the critical section only after the process has acquired a token. In the permission based approach (Raymond, 1989), a process must request and be explicitly granted permission to enter the critical section from a specific subset of processes. While there has been significant research in distributed mutual exclusion algorithms, aspects specific to CPS such as users interacting with another and using additional mechanisms external to the algorithm to acquire and release resources, have not been studied.

In (Walter et al., 2001), a token based $k$-mutual exclusion algorithm, which is referred to as the KRL algorithm, was proposed for wireless ad hoc networks. As one of our solutions is based on the KRL algorithm, we discuss this algorithm in more detail in the following. In the KRL algorithm, each node $i$ maintains a data structure $\text{height}_i$, which is a three-tuple $(h_1, h_2, i)$. Edges are directed from higher height nodes to lower height nodes based on lexicographic ordering. For example, if $\text{height}_0 = (2, 3, 0)$ and $\text{height}_1 = (2, 2, 1)$, then $\text{height}_0 > \text{height}_1$ and the edge will be directed from node 0 to node 1. KRL algorithm maintains $n$ nodes and $k$ tokens, where $k < n$. For all nodes $i$, $\text{height}_i$ is initialized so that the directed edges form a directed acyclic graph (DAG) such that every node has a directed path to some token holder and every token holder node $i$ has at least one neighbor $n$ such that $\text{height}_n > \text{height}_i$. When a user at node $i$ wants to enter the critical section, it makes a request which is enqueued by $P_i$ in its local queue $Q_i$. When $P_i$ receives a request from neighbor $P_j$ and $\text{height}_j > \text{height}_i$, $P_i$ enqueues the request in $Q_i$. If $P_i$ is a non-token holding node and $Q_i$ is empty when the request is received, $P_i$ sends a request to its neighbor with the lowest height. Hence, requests propagate via lower height nodes to the token holders. If $P_i$ has (or receives) a token, it dequeues the first request from $Q_i$. If this request is from its own application process, $P_i$ gives permission to its application process; else, it sends the token to its neighboring node whose request it just dequeued.

3 Cyber-Physical Systems

We model a CPS as a triple $(\text{CyS}, \text{PhyS}, \text{Int})$. $\text{CyS}$ is defined in the same way as in a TDS. $\text{PhyS}$ is defined as a pair $(P_E, G_P)$, where $P_E$ is the set of physical entities and $G_P$ is a graph $(PA, RE)$, where $PA$ is a set of physical areas and $RE$ is the set of reachability edges. An edge $R_{ij} \in RE$ represents the fact that a physical entity can move directly from area $PhA_i$ to area $PhA_j$. A reachability edge $R_{ij}$ between $PhA_i$ and $PhA_j$ is analogous to a communication link $E_{ij}$ between two cyber entities $V_i$ and $V_j$. Figure 2 shows the graph $G_P$ for the example in Figure 1. For example in Figure 1, since there is a doorway connecting $PhA_1$ and $PhA_2$, there is a reachability edge between them. We further partition $P_E$ into two sets, $AE$ and $RS$, where $AE$ is a set of active entities and $RS$ is a set of resources. Active entities are the users of the system and can perform actions on their own (e.g., hospital staff) and may use the resources (e.g., wheelchairs) in the set $RS$.

We model each area $PhA \in P_E$ by two abstract variables, $PhA.ae$ and $PhA.rs$, which denote the set of active entities and set of resources respectively currently located in area $PhA$. Similarly, we model the state of a resource $r \in RS$ by a variable $r.state$, which is either free or busy. Figure 2 shows these values for the current state in Figure 1. We assume that these abstract variables are updated automatically based on the actions of active entities. For instance, if $U \in PhA.ae$, and $U$ moves out of the area $PhA$, then the state variable $PhA.ae$ is automatically updated so that $U$ is removed from $PhA.ae$. Similarly, when $U$ enters a new area, the corresponding state variable is updated to include $U$. We assume a similar update happens for $PhA.rs$ when resources are moved between areas.

3.1 Interaction between entities

This section defines the possible actions by active entities and interactions between the cyber- and physical-subsystems:

- **Sensing:** If a cyber entity $V \in CE$ has the capability of sensing the presence of physical entities in an area $PhA$, then we model this by allowing processes in $V.processes$ to read $PhA.ae$ and $PhA.rs$. Furthermore, we also allow these processes to read $r.state$ for each resource $r \in PhA.rs$. The read operations return the
value from the most recent sensing activity which may be different from the current values of these abstract variables.

- **Actions performed by an active entity:** We use the following actions to describe the behavior of an active entity $U \in AE$:
  - **Move(PhA)** is an action which represents $U$ moving from its current physical area to another physical area $PhA \in PA$, and is possible only if there exists a reachability edge from its current physical area to $PhA$.
  - **Observe()** is an action which represents $U$ observing physical objects within an observation radius ($O_R$). If $O_R$ of $U$ located in $PhA \in PA$ is 1, then $U$ can observe the status of $PhA$, i.e., $U$ can read $PhA.ae$ and $PhA.rs$, and $r.state$ for each resource $r \in PhA.rs$. In general, if $O_R$ of $U$ is $r$, then $U$ can observe the status of all areas reachable via at most $r$ hops in $G_P$. For implementation purposes, $Observe()$ returns the set of resources which $U$ can observe depending on its $O_R$.
  - While the Move and Observe actions can help $U$ locate resource on its own, it can also interact with the cyber algorithm. We assume that $U$ can use the action Send_request() to request the cyber-subsystem for a resource. When the cyber-subsystem has located the resource, $U$ uses Receive(path) to receive a path from the cyber-subsystem. Note that although the cyber-subsystem uses the edges in $G_C$ to communicate and locate resources, the path delivered to the user is a path in the graph $G_P$ from the current location of $U$ to a location of the resource.
  - Once a free resource has been identified, either by $U$ itself or by the cyber algorithm, $U$ must travel to the location of the resource to acquire it. The action Acquire(rs) represents the attempt by $U$ to physically acquire a resource $rs$ which results in $rs.state$ being set to busy. Since active entities and the cyber algorithm may operate independently, there could be situations where several active entities may attempt to acquire the same resource (e.g., if several people attempt to grab a wheelchair simultaneously, only one of them will be successful). To model this, we assume that the Acquire action is successful only if $rs.state$ is free, and that a successful Acquire will automatically change $rs.state$ to busy. Thus, Acquire(rs) $:< rs.state = free \rightarrow rs.state = busy >$ by $U$ can be viewed as an atomic action which returns true if $U$ successfully acquires $rs$; else it returns false.
  - **Release(rs)** is used to physically release a resource $rs$ which results in $rs.state$ being set to free, and is successful only if $rs.state$ is busy. Thus, Release(rs) $:< rs.state = busy \rightarrow rs.state = free >$ is an atomic action.

4 Mutual Exclusion in a CPS

In this section, we present mutual exclusion algorithms for a CPS. Each algorithm has two components: (a) the behavior of active entities describing their efforts to locate resources and (b) a cyber algorithm.

4.1 Behavior of active entities

Behavior describes the steps followed by an active entity to locate a resource with the help of Observe and Move actions. These actions can be described as an algorithm. We consider the following possible behaviors:

- **Behavior $B_0$:** In this behavior, $U$ searches for a resource without any help from the cyber-subsystem. At each step, if $U$ observes a free resource, it will attempt to acquire it. If unsuccessful (note that another active entity may attempt to acquire the same resource at the same time), then it picks a random adjacent area and moves to that area via the connecting reachability edge.

- **Behavior $B_1$:** This behavior represents the other extreme wherein $U$ sends a request message to the cyber-subsystem and waits for a response; then it follows the path received in the message. In this case, it will always successfully acquire a resource in the target area as the access is regulated solely by the cyber-subsystem.

- **Behavior $B_2$:** In this behavior, $U$ sends a request message to the cyber-subsystem and waits for a response. Subsequently, it follows the path received in the mes-
sage. However, as it moves, it also observes each intermediate PhA for a free resource; if available, it will attempt to acquire it.

**Behavior B₂:** In B₂, after delivering a path to a resource R₁ to U, the cyber-subsystem may find that another resource R₂ has been released subsequently which may be closer to U. B₃ is a variation of B₂ wherein U can dynamically accept updated paths from the cyber-subsystem while it is moving, and follows these shorter paths.

**Algorithm 1 Behavior 0**

L1: \(rs \leftarrow \text{observe}();\)
   if \((rs \neq \text{empty})\)
      for each \(r \in rs\)
         if \((r\text{.state} = \text{free})\) go to L2;
      select an unvisited neighbor \(n\) from graph \(G_P\);
      move(\(n\));
      go to L1;
L2: \(val \leftarrow \text{acquire}(r);\)
   if \((val = \text{false})\) go to L1;

**Algorithm 2 Behavior 1**

L1: \(\text{send_request}();\)
   \(path \leftarrow \text{receive_path}();\)
   while (NOT end of \(path\))
      \(p_1 \leftarrow \text{next_hop in } path;\)
      move(\(p_1\));
      \(rs \leftarrow \text{observe}();\)
      if \((rs \neq \text{empty})\)
         for each \(r \in rs\)
            if \((r\text{.state} = \text{free})\) then \(val \leftarrow \text{acquire}(r);\)
   /*receive updated path from cyber-subsystem.
   It is non-blocking receive*/
   \(new_path \leftarrow \text{receive_path}();\)
   if \((new_path \neq \text{empty})\)
      \(path \leftarrow new_path;\)
   if \((val = \text{false})\) then go to L1;

**Algorithm 3 Behavior 2**

L1: \(\text{send_request}();\)
   \(path \leftarrow \text{receive_path}();\)
   \(val \leftarrow \text{false};\)
   while (not end of \(path\))
      \(p_1 \leftarrow \text{next_hop in } path;\)
      move(\(p_1\));
      \(rs \leftarrow \text{observe}();\)
      if \((rs \neq \text{empty})\)
         for each \(r \in rs\)
            if \((r\text{.state} = \text{free})\) \(val \leftarrow \text{acquire}(r);\)
   if \((val = \text{false})\) then go to L1;

We have identified some possible behaviors of active entities above. Clearly, variations of these behaviors (including more complex ones which, for instance, involve active entities cooperating to avoid conflicts) can be defined in our proposed model. We have identified and studied one such cooperative behavior and has shown that it can significantly reduce the time to acquire a resource.

### 4.2 Cyber algorithms

To accommodate the different behaviors (B₁, B₂, and B₃), we have developed both centralized and distributed cyber algorithms. The algorithms assume that each \(V_i \in CE\) runs exactly one process denoted by \(P_i\). Each physical area PhA is sensed by exactly one cyber entity \(V_i\) and sensing ranges of cyber entities do not overlap. This eliminates possibilities of two cyber entities sensing the same resource and a physical resource not being sensed by any cyber entity.

#### 4.2.1 Centralized algorithm

For the centralized algorithm, we assume that there is a central server, and that all sites have routing information to forward packets to and from the server. In this algorithm, the server attempts to maintain up-to-date view of the entire system. For this purpose, it maintains a data structure called CPSView which comprises of \(G_P, G_C\), and the states of all areas. This information is updated by messages from processes as the state of the system changes. For example, in Figure 1, if \(U_1\) moves from PhA₁ to PhA₂, a lose message is sent by
When the server finds a free resource, say in area which is closer or some other active entity may have (but with resource marked as busy). This modified for that resource.

PU resources. For example, if entity moves. This edge reversal results in a new sink at from area). In the initial DAG which consists of two sink nodes moves. An important aspect is how the existing DAG algorithm proceeds, the DAG is modified as the token

KRL the algorithm. The algorithm initializes the system so that the directed edges form a DAG. As the algorithm proceeds, the DAG is modified as the token moves. An important aspect is how the existing DAG is re-used for subsequent searches. Figure 3(a) shows the initial DAG which consists of two sink nodes P1 and P2 with resources R1 and R2 respectively (this scenario assumes one cyber entity in each physical area). In the KRL algorithm, when entity U1 requests a resource and is granted R1, the token is passed along from P1 to P2, and the edges are reversed as the token moves. This edge reversal results in a new sink at P3 (but with resource marked as busy). This modified DAG is then subsequently used by others for locating resources. For example, if entity U2 in PPhA7 wants a resource and P2 picks P4 as its next hop (which is possible in the KRL algorithm), then the request will be forwarded using the existing edges to P3. In the context of a CPS, the following scenarios in the KRL algorithm must be addressed. First, the location of the token becomes delinked from the location of the resource. In the scenario above, the resource is still in PPhA1 even though the token has moved to P3. Furthermore, the user may acquire the resource at PPhA1 and release it at another location with the token still residing at P3. Since the time to acquire a resource also includes the time it takes for the user to move to the location of the resource, we must minimize this distance as well. Second, when the token reaches P3, it is marked as busy. If a user at P3 requests a resource and this request reaches P3, it must wait until the token is free even though another free resource (R2) exists. This second problem was somewhat alleviated by the variation proposed in (Walter et al., 2001).

In this paper, we have explored two strategies to address the issues discussed above. The first strategy, termed as KRL C P S, is a variation of the KRL algorithm wherein we do not perform edge reversal when the token moves. Rather, to keep the root of the tree linked to the location of the resource, we perform edge reversal only when the resource moves. Thus, in the scenario in Figure 3(a), the first tree will remain rooted at P1. Only when U1 moves to PPhA1 and then moves the resource to another location, edge reversals will occur.

The second strategy, termed Shortest Path Resource Allocation (SPRA), disregards the existing path information and creates paths on a on-demand basis. SPRA maintains a set of trees called SPTrees. Each SPTree is rooted at P1 where V1 senses a free resource. Each P1 maintains two variables, ptrR and height, attri is a tuple (ptrR, height). Initially, for all Pi, attr = (NULL, ∞). Each P1 also maintains a set NbrAttri which contains the most recent attr elements received from the neighboring nodes.

Figure 3(b) shows an initial setup showing two SPTrees rooted at P1 and P2. When a process makes a request, the request is forwarded via the parent pointers to the tree root. For example, when U1 located in PPhA3 makes a request, P3 will propagate the request to P1. On receiving this request, P1 sets R1.state to locked and sends confirmation back to P3 via intermediate child pointers. When U1 receives the confirmation, it has to move along the path to reach PPhA1. SPTrees are created and maintained as follows. As soon as Pi senses a free resource, it sets attr to (SELF, 0). Whenever attr changes, Pi broadcasts the new value to its neighbors. If Vj is neighbor of Vi, on receiving attr, P1 executes Algorithm 5 which first updates NbrAttrj, and then updates attrj if the value received from Pi
Algorithm 5 SPTree management

update attr_j in Nbr_Attr_j
if (ptr_j = SELF or ptrR_j = P_j)
   /* V_j either senses at least one free resource 
or P_j points to P_j itself. */
   exit();
else
   min_height ← min(height_k), attr_k ∈ Nbr_Attr_j;
   if (height_j ≤ min_height + 1)
      /*height_j is already minimum=*/
      exit();
   else
      attr_j ← (P_j, height_k + 1);
      broadcast attr_j;

provides a lower cost path to a free resource. When a resource, say R1 in PhA3 in Figure 3(b) is locked, P_i changes attr_j to (NULL, ∞), and sends this value to its children (which are propagated further). Hence, all the nodes in the tree will set their attr to (NULL, ∞) (shown in Figure 3(c)). Subsequently, these nodes connect to trees on an on-demand basis. For example, when P_j receives a request from U_2, P_j will attempt to rediscover a resource. It initiates a breadth first search. The resulting SPTrees are shown in Figure 3(d). Figure 3(e) shows SPTrees after U_1 moves R1 to PhA3 and releases it there.

5 SIMULATION AND RESULTS

We used the OMNeT++ Discrete Event Simulation System (Varga, 2001) to simulate the algorithms. There are many extensions available to simulate specific types of systems using OMNeT++. MiXiM (Wessel et al., 2009) is one such extension of OMNeT++ to simulate wireless and mobile networks and provides detailed models of the lower layers of the protocol stack. Our simulation is built on top of MiXiM. Each node in a simulation has two layers, application layer (appl) and network layer (net). Figure 4 shows the architecture of a node in the simulation. The behavior of each entity is coded in appl. For example, the appl at each cyber entity implements the cyber-subsystem algorithm being simulated. Each node has a mobility component (mob) which is connected to its appl. For example, in behavior B_1 (Algorithm 2), the move(p) action is implemented by the appl sending a move message to its mob component. When an active entity U_1 moves a resource R_1, appl of both U_1 and R_1 send move messages to their respective mob components at the same time. We have also simulated the sensing activities by having the appl layers of the resources and active entities send periodic messages announcing their presence. This interaction and movement of active entities and resources can be visualized on the screen during run time.

In the following discussion, active entity and resource will be referred to as person and wheelchair respectively. We use AT to represent the Acquire Time, which is the time elapsed from when a request is made and a wheelchair is acquired, NM represents the total number of messages generated in the network per request, and NH represents the number of physical areas a person needs to move to get a wheelchair. AT, NM and NH are averaged for 100 requests per person. For the experiments, we fixed the time it takes for a person to move from one area to an adjacent one to 3 seconds, and assumed that each person uses a wheelchair for a random amount of time between 20 and 30 seconds. Furthermore, we assume that cyber entities sense the status of the physical area it is located in every 100ms, and the default Q_R is 1. We use the 5-tuple <M, K, B_i, N_P, N_W> to represent a system configuration having N_P persons and N_W wheelchairs located in a grid of size M * K (or G_M,K) of physical areas and all N_P persons following behavior B_i. V_x,y represent the cyber entity located in row x and column y of the grid.

• Comparison of KRL-CPS and SPR

We started by simulating KRL-CPS algorithm discussed in Section IV B. In the first scenario of KRL-CPS, which we call KRL-S, a wheelchair is released at the same location where it was acquired. KRL-D refers to a scenario in which wheelchair is
Next, we wanted to analyze the impact of different behaviors of active entities on \( AT \) and \( NH \). In what follows, \( SPRA-N \) denotes \( SPRA \) algorithm for behavior \( B_N \) where \( 1 \leq N \leq 3 \). We first studied the impact of releasing wheelchairs in random areas on \( AT \) and \( NH \) by keeping \( N_W \) constant and varying \( N_P \). The results are shown in Figure 7. As discussed earlier, in \( B_0 \) (referred to as \( NoCS \) in Figure 7(a)), a person attempts to visit areas on its own (without help of the cyber-subsystem). This results in a high value of \( AT \). For the other three behaviors, we observed the following: When \( N_P = 7 \) and \( N_W = 3 \), there is increased competition for wheelchairs. As a result, it is less likely that a person will locate another free wheelchair when it is moving to the location of the free wheelchair initially identified by the cyber-subsystem (which it tries to do in \( B_2 \) and \( B_3 \)). Similarly, it is less likely that the cyber-subsystem will be able to provide an updated path. Hence, the performance of the three behaviors coincide for this scenario. As the number of persons is decreased (from 7 to 5), there is less competition and the scenarios wherein free wheelchairs can be located by the person or the cyber-subsystem become more probable, and \( SPRA-3 \) outperforms \( SPRA-2 \), which in turn outperforms \( SPRA-1 \). As \( N_P \) is further decreased (say to 2), we find that a free resource will always be available and hence, the initial location identified by the cyber-subsystem is most likely to be the nearest one. Hence, the performances again converge. The impact on \( NH \) is similar (see Figure 7(b)). We observed similar pattern of variation in \( AT \) and \( NH \) for configurations simulated on \( G_{12,12} \).

To further study the impact of various behaviors, we conducted a set of experiments with localized release of wheelchairs. Figure 8 shows the setup of \( G_{12,17} \) with four distinct parts of the grid labeled \( A_1 \), \( A_2 \), \( A_3 \), and \( A_4 \). Initially, \( A_1 \), \( A_2 \), \( A_3 \), and \( A_4 \) has 5, 0, 3 and 3 wheelchairs respectively. We assume that \( A_2 \) is similar to an entrance area and all requests are made by persons in this area (we assume a total of 15 persons). We assume that a wheelchair acquired in grid area \( A_i \) is released within that grid itself (localized release). Figure 9 shows the impact on \( AT \) when the distance between areas \( A_1 \) and \( A_2 \) (dist(\( A_1, A_2 \))) is increased from 0 to 6. The scenario shown in Figure 8 corresponds to dist(\( A_1, A_2 \)) = 0, and we incrementally move \( A_2 \) closer.
to A₁ in each experiment. In Figure 9, we see that AT for B₃ is lower than B₂ and B₁ when A₂ is close to either A₁ or to A₃. This can be explained as follows: Consider the scenario where dist(A₁, A₂) = 0, and a wheelchair is released in A₁. Just prior to this moment, assume that a user U in A₂ had requested a wheelchair and was supplied a path to a free wheelchair in A₃ or A₄. In this case, it is likely that the cyber subsystem will find the newly released wheelchair in A₁ closer, and will deliver a shorter path to U. However, as dist(A₁, A₂) increases, this scenario becomes less likely and hence the performance of B₃ converges to that of B₂ (see Figure 9). However, as dist(A₂, A₃) decreases, the scenarios with free resources in A₃ or A₄ become likely, and again B₁ starts performing better than B₂. The results for the centralized algorithm follow a similar pattern.

**Impact of Server location**

In this setup, we increased the server location for the centralized algorithm in Figure 8 such that each time server communicates only with V₁,n, 1 ≤ n ≤ 17. We noticed that as the server distance from grid A₂ (from where persons make requests) increases, AT also increases. This is due to the fact that it takes longer for each request to travel to the central server. This shows that server location should be chosen carefully if one decides to opt for a centralized solution. Figure 10 shows the detailed results.

**Impact of OR**

In this setup, we increased OR of each person from 1 to 8 for configurations < 8,8,Bᵢ,5,3 >, 1 ≤ i ≤ 3. The results for SPRA are shown in Figure 11. The performance of B₁ is not impacted by OR. The performance of B₂ improves as OR is increased from 1 to 4 – this is due to the fact that a person can observe more areas and hence the chances of finding a nearby free wheelchairs increase. However, as OR is increased further, performances of B₂ starts degrading because the observation zones of the users overlap a lot. Hence, there are more chances that whenever a wheelchair becomes free, multiple users might observe it and deviate from their original paths towards this free wheelchair. Since only one of them will be successful, others will
have to incur additional hops. The performance of $B_3$ show a similar pattern except that when $O_R$ is increased from 1 to 4, we do not see much change. In this case, we find that the cyber-subsystem is able to provide quick updates of newly freed wheelchairs which are close.

- **Impact of cyber-subsystem delay ($CS_D$)**

In the experiments above, we assumed that the cyber subsystem delay, which is the time gap between two sensing activities, as 100ms. Note that the value of state variables ($rs.state$, $PhA.ae$ and $PhA.rs$) available to processes correspond to the last sensing activity. Hence, as $CS_D$ increases, the probability that the state information available to the cyber-subsystem is stale increases. We increased $CS_D$ from 100ms to 3500ms and analyzed the impact on the performance of various behaviors. Since the allocation is completely controlled by the cyber-subsystem in $B_1$, the performance degrades linearly as $CS_D$ is increased. However, in $B_2$ and $B_3$, it is possible that a person, say $U_1$, may acquire a wheelchair $R_1$ in area $PhA_1$ which was not allocated to $U_1$ by the cyber-subsystem. However, this fact will only become known to the cyber-subsystem during the next sensing activity. Increasing $CS_D$ will increase this period, and may result in the cyber algorithm making decisions based on stale information. As a result, we find that the performance of $B_2$ and $B_3$ degrades significantly as the cyber subsystem delay is increased. Figure 12 shows the results for configurations $<8,8,B_i,5,3>, 1 \leq i \leq 3$.

- **Cooperative user behavior** We also experimented with a scenario in which the users cooperate with each other. When a user $U_1$ is using a wheelchair $R_1$ and another user $U_2$ observes $U_1$ using $R_1$, $U_2$ assumes that $U_1$ will release $R_1$ at some point of time. In our existing algorithm, $U_2$ will wait for a notification of the release and then move towards the location where $R_1$ has been released. From our experiments, we found that the time spent in moving to the resource after the release notification adds significantly to the acquire time ($AT$). We therefore modified the behavior as follows: If $U_2$ finds that $U_1$ is using $R_1$, $U_2$ starts following $U_1$ and as soon as $U_1$ releases $R_1$, $U_2$ will attempt to acquire it. This overlaps the time when $R_1$ is being used with the time spent in moving after the resource is released. Furthermore, if a third person $U_3$ observes that $U_2$ is moving towards $U_1$, $U_3$ assumes that $U_2$ wants to acquire $R_1$ after $U_1$ will releases it. In this case, $U_3$ does not follow $U_1$. In our earlier experiment (Figure 11), $AT$ increased when observation radius is increased beyond a threshold due to increased competition. Figure 13 shows the results for NoCS as well as the updated SPRA-1, SPRA-2, and NoCS behaviors in which persons cooperate as described above. As one can see, the cooperative behavior is able to resolve the conflicts due to competition and $AT$ decreases as radius is increased.

6 Conclusions and future work

Graph based models with various assumptions related to message transmission and processing times have provided a strong foundation to study distributed algorithms in a TDS. This paper provides a step towards studying similar algorithms for CPSs. We presented a graph based formalism to model both the cyber- and the physical-subsystems and the interactions between them. Based on this model, we presented algorithms for the mutual exclusion problem. Each algorithm had two components, one describing the behavior of users in the physical-subsystem and the other describing the cyber algorithm. We identified several characteristic of a CPS which make solutions for TDS inapplicable to a CPS. We simulated all the presented algorithms using OMNeT++. The results provide suggestions on the best algorithm to use in different scenarios. For example, the results show that when fewer resources are present, it might be best to rely completely on the cyber-subsystem; otherwise, participation of users in locating resource can improve performance. The model proposed in this paper opens the possibility of studying more complex scenarios and algorithms for CPSs. These possibilities include
associating properties with the reachability edges in $G_P$ and cooperation between the users in locating resources. Finally, in the SPRA algorithm, identifying mechanisms via which existing tree information could be utilized in creating on-demand paths to reduce number of messages is a subject of future research.

ACKNOWLEDGEMENTS

This work was supported in part by NSF Grant 0615337 and K-State Targeted Excellence program.

REFERENCES


