MUTUAL EXCLUSION IN CYBER-PHYSICAL SYSTEMS

Sumeet Gujrati and Gurdip Singh
Computing and Information Sciences, Kansas State University, Nichols 234, Manhattan, U.S.A.

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) $U1,\dots,Un$ is layered on top of a mutual exclusion algorithm with processes $P_1,\dots,P_n$. User Ui 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 $P_1,\dots,P_n$, where P_i executes on node V_i, and a strict layered structure is used wherein user Ui 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.

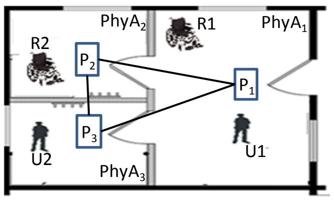


Figure 1: Cyber-infrastructure superimposed on a physical system.

In a cyber-physical system (CPS), the cyber- infrastructure may have been superimposed on an existing physical system (Figure 1). 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 $U1$ observes that resource $R1$ is available, it may acquire $R1$ without waiting for a response from P_1, 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 $P1$ in Figure 1, may start using $R1$ 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. 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. The contribution of this paper is three-fold:

- We propose a model which views a CPS as a triple $= (CyS, PhyS, Int)$, where CyS models the cyber-infrastructure superimposed on physical system modeled as $PhyS$, and Int captures the interactions between them. We call CyS and $PhyS$ cyber-subsystem and physical-subsystem respectively.
- We propose a set of algorithms based on the proposed model for the mutual exclusion problem. Each algorithm has two components, one describing the behavior of the users in $PhyS$ and the other describing the mutual exclusion algorithm (or cyber algorithm) in 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.

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 $V.processes \in CP$, 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 (Bulgannawar and Vaidya, 1995; Makki et al., 1992; Srimani and Reddy, 1992; Walter et al., 2001; Raymond, 1989).

In (Walter et al., 2001), a 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 $height_i$, 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 $height_0 = (2,3,0)$ and $height_1 = (2,2,1)$, then $height_0 > height_1$ 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, $height_i$ 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 $height_n > height_i$. 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 $height_j > height_i$, 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 $(CyS, PhyS, Int)$. CyS is defined in the same way as in a TDS. $PhyS$ is defined as a pair (PE, G_P), where PE 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. 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 PE 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 $PhA \in PA$ 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*. 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 $r \in PhA.rs$.

• **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.
- The action *Acquire(rs)* represents the attempt by U to physically acquire a resource *rs* which results in *rs.state* being set to *busy*.
- *Release(rs)* is used to physically release a resource *rs* which results in *rs.state* being set to *free*.

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. 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 message. However, as it moves, it also observes each intermediate *PhA* for a free resource; if available, it will attempt to acquire it.

Behavior B_3: In B_2, 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. B_3 is a variation of B_2 wherein U can dynamically accept updated paths from the cyber-subsystem while it is moving, and follows these shorter paths.

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 (Gujrati and Singh, 2011).

4.2 Cyber Algorithms

To accommodate the different behaviors (B_1, B_2, and B_3), 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_i 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. Due to space limitations, we do not describe centralized algorithm and certain details such as how cyber-subsystem reacts when a resource reserved for a user is acquired by some other user.

In this paper, we have explored two distributed strategies to solve mutual exclusion in CPS. The first strategy, termed as *KRL_CPS*, is a variation of the *KRL* algorithm wherein we perform edge reversal only when an active entity moves the resource to another location. Thus, in the scenario in Figure 2(a), the P_1 remains the sink node as long as there is a resource in PhA_1.

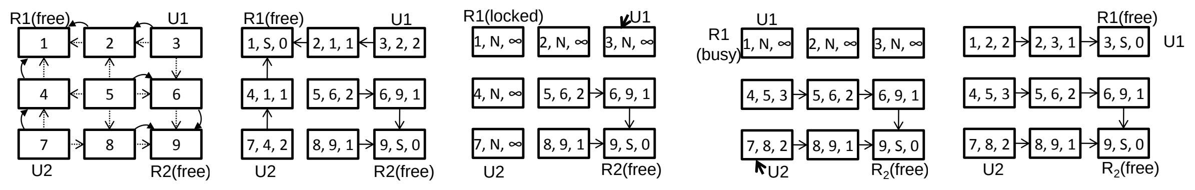
The second strategy, termed *Shortest Path Resource Allocation* (*SPRA*), disregards the existing path information and creates paths on a on-demand basis. *SPRA* maintains a set of trees called *SPTree*s. Each $SPTree_i$ is rooted at P_i where V_i senses a free resource. Each P_i maintains two variables, $ptrR_i$ and $height_i$. $attr_i$ is a tuple $(ptrR_i, height_i)$. Initially, for all P_i, $attr_i$ = $(NULL, \infty)$. Each P_i also maintains a set Nbr_Attr_i which contains the most recent $attr$ elements received from the neighboring nodes.

Figure 2(b) shows an initial setup showing two *SPTree*s rooted at P_1 and P_9. When a process makes a request, the request is forwarded via the parent pointers to the tree root. For example, when $U1$ located in PhA_3 makes a request, P_3 will propagate the request to P_1. On receiving this request, P_1 sets $R1.state$ to *locked* and sends confirmation back to P_3 via intermediate child pointers. When $U1$ receives the confirmation, it has to move along the path to reach PhA_1. *SPTree*s

Algorithm 1: SPTree management.

```
update $attr_i$ in $Nbr\_Attr_j$;
if $(ptrR_j = SELF$ or $ptrR_i = P_j)$
    /* $V_j$ either senses at least one free resource
        or $P_i$ points to $P_j$ itself. */
    exit();
else
    $min\_height \leftarrow min(height_k)$, $attr_k \in Nbr\_Attr_j$;
    if $(height_j \leq min\_height + 1)$
        /*$height_j$ is already minimum.*/
        exit();
    else
        $attr_j \leftarrow (P_k, height_k + 1)$;
        broadcast $attr_j$;
```

are created and maintained as follows. As soon as P_i senses a free resource, it sets $attr_i$ to $(SELF, 0)$. Whenever $attr_i$ changes, P_i broadcasts the new value to its neighbors. If V_j is neighbor of V_i, on receiving $attr_i$, P_j executes Algorithm 1 which first updates Nbr_Attr_j, and then updates $attr_j$ if the value received from P_i provides a lower cost path to a free resource. When a resource, say $R1$ in PhA_1 in Figure 2(b) is locked, P_1 changes $attr_1$ to $(NULL, \infty)$, and sends this value to its children (which are propagated further). Hence, all



(a) *KRL* initial DAG. (b) *SPRA* initial setup. (c) $R1$ is locked. (d) P_7 discovers new path. (e) $U1$ releases $R1$.

Figure 2: (a): *KRL* Algorithm, (b) to (d): *SPRA* Algorithm - $S = SELF$, $N = NULL$, triple at each node = (P_i, $ptrR_i$, $height_i$).

the nodes in the tree will set their *attr* to $(NULL, \infty)$ (shown in Figure 2(c)). Subsequently, these nodes connect to trees on a on-demand basis. For example, when P_7 receives a request from $U2$, P_7 will attempt to rediscover a resource. It initiates a breadth first search. The resulting $SPTree$s are shown in Figure 2(d). Figure 2(e) shows $SPTree$s after $U1$ moves $R1$ to PhA_3 and releases it there.

5 SIMULATION AND RESULTS

We used the OMNeT++ Discrete Event Simulation System (Varga, 2001) to simulate the algorithms. MiXiM (Wessel et al., 2009), an extension of OMNeT++ to simulate wireless and mobile networks, provides detailed models of the lower layers of the protocol stack. Our simulation is built on top of MiXiM. See (Gujrati and Singh, 2011) for detailed discussion on node architecture in the simulation.

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 O_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 SPRA.*
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 released at a random location (which is more realistic). As shown in Figure 3, for configuration $< 8, 8, B_1, 6, 3 >$, *NH* is 18.3 in *KRL-D*, and 12.4 in *KRL-S*. The difference is due to the fact that when the wheelchair moves from its original location, the edges are reversed (hence, a linear chain of parent pointers will be created from its original location to the new location). The corresponding *NH* for *SPRA* is 8.3 with wheelchairs released at random locations. As can be seen in Figure 3, as the N_W is increased (with N_P kept constant), the difference between the performances of the three algorithms reduce. This is due to the fact that with more wheelchairs (e.g., 10 wheelchairs in a G_{8*8}), the trees have smaller depths. As *NH* is higher for *KRL_CPS* as compared to *SPRA* algorithm, one would expect *AT* also to be higher. Figure 4 shows the performance of these algorithms with respect to *AT*. As can be seen, *SPRA* outperforms both *KRL-D* and *KRL-S*. We also simulated similar configurations with G_{12*12} and the results follow a similar pattern. However, the performance gain for *SPRA* comes at the expense of increased number of messages. To recreate paths on demand, we have to conduct a breadth first search when a wheelchair is requested. Whereas *NM* is 31 for *KRL-S* and 47 for *KRL-D*, it is 93 for *SPRA* for configuration $< 8, 8, B_1, 6, 3 >$. However, as N_W is increased (keeping N_P fixed), we find that the cost of re-creating paths drops as it is more likely that nearby resources can be found (and hence, the breadth first search terminates in relatively fewer number of hops). Our simulation show that *NM* drops from 93 to 61 as N_W is increased from 3 to 6.

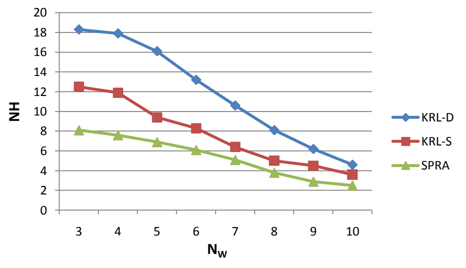


Figure 3: *NH* vs N_W for $< 8, 8, B_1, 6, N_W >, 3 \leq N_W \leq 10$.

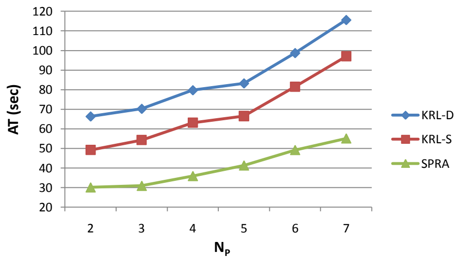
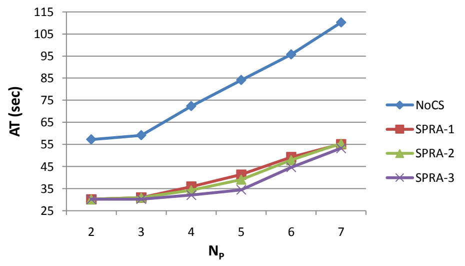


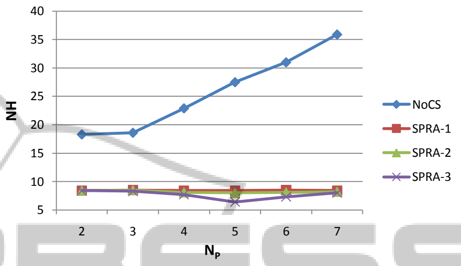
Figure 4: *AT* vs N_P for $< 8,8,B_1,N_P,3 >, 3 \leq N_P \leq 7$.

• *Comparison of Different Behaviors.*
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 5. As discussed earlier, in B_0 (referred to as *NoCS* in Figure 5(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 is 7 and N_W is 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 5(b)).

• *Impact of* O_R.
In this setup, we increased O_R of each person from 1 to 8 for configurations $< 8,8,B_i,5,3 >, 1 \leq i \leq 3$. The results for *SPRA* are shown in Figure 6. The performance of B_1 is not impacted by O_R. The performance of B_2 improves as O_R 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 O_R is increased further, performances of B_2 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.



(a) *AT* vs N_P.



(b) *NH* vs N_P.

Figure 5: Impact of varying N_P on *AT* and *NH* when $N_W = 3$ for G_{8*8}.

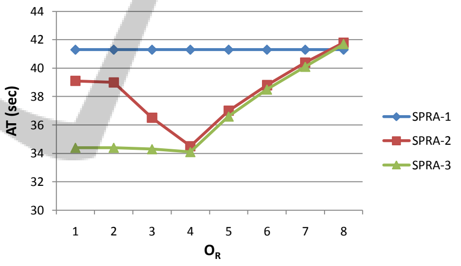


Figure 6: Impact of increasing O_R on *AT*.

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

Bapat, S. and Arora, A. (2008). Message efficient termination detection in wireless sensor networks. In *Proceedings of the 2008 INFOCOM Workshops*.

Bulgannawar, S. and Vaidya, N. (1995). A distributed k-mutual exclusion algorithm. In *Proceedings of the 15th International Conference on Distributed Computing Systems*, pages 153–160.

Chandy, K. M. and Lamport, L. (1985). Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1).

Chinrungrueng, J., Sunantachaikul, U., and Triamlumlerd, S. (2007). Smart parking: an application of optical-wireless sensor network. In *Proceedings of the 2007 International Symposium on Applications and the Internet Workshops*.

Conner, W. S., Heidemann, J., Krishnamurthy, L., Wang, X., and Yarvis, M. (2004). Workplace applications of sensor networks. In *USC/ISI Technical Report ISI-TR-2004-591*.

Dijkstra, E. (1965). Solution of a problem in concurrent programming control. *Communications of the ACM*.

Dijkstra, E. (1971). Hierarchical ordering of sequential processes. *Acta Informatica*, 1:115–138.

Gujrati, S. and Singh, G. (2011). Mutual exclusion in cyber-physical systems. Technical report, Kansas State University CIS TR 2011-2.

Kaveti, L., Pulluri, S., and Singh, G. (2009). Event ordering in pervasive sensor networks. In *5th IEEE International Workshop on Sensor Networks and Systems for Pervasive Computing*.

Kurian, H., Rakshit, A., and Singh, G. (2009). Detecting termination in pervasive sensor networks. In *9th IEEE International Symposium on Asynchronous Decentralized Systems*.

Liu, M., Mihaylov, S. R., Bao, Z., Jacob, M., Ives, Z. G., Loo, B. T., and Guha, S. (2010). Smartcis: Integrating digital and physical environments. *SIGMOD Record*.

Lynch, N. (1980). Fast allocation of nearby resources in a distributed system. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing*, pages 70–81.

Makki, K., Banta, P., Been, K., Pissinou, N., and Park, E. (1992). A token based distributed k mutual exclusion algorithm. In *IEEE Proceedings of the Symposium on Parallel and Distributed Processing*, pages 408–411.

Rajamani, V. and Julien, C. (2010). Blurring snapshots: Temporal inference of missing and uncertain data. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications*.

Raymond, K. (1989). A distributed algorithm for multiple entries to a critical section. In *Information Processing Letters*, volume 30, pages 189–193.

Reif, J. H. and Spirakis, P. (1982). Real time resource allocation in distributed systems. In *Proceedings of ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 84–94.

Romer, K. (2003). Temporal message ordering in wireless sensor networks. In *Annual Mediterranean Ad Hoc Networking Workshop*.

Srimani, P. and Reddy, R. (1992). Another distributed algorithm for multiple entries to a critical section. In *Information Processing Letters*, volume 41, pages 51–57.

Varga, A. (2001). The omnet++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM 2001)*.

Walter, J., Cao, G., and Mohanty, M. (2001). A k-mutual exclusion algorithm for wireless ad hoc networks. In *Proceedings of the First Annual Workshop on Principles of Mobile Computing*.

Wessel, K., Swigulski, M., , Kpke, A., and Willkomm, D. (2009). Mixim: the physical layer an architecture overview. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*.

Wieland, M., Kopp1, O., Nicklas, D., and Leymann, F. (2007). Towards context-aware workflows. In *Proceedings of the CAISE07 Workshops and Doctoral Consortium*.