2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

Distributed State Estimation in Discrete Event Systems

S. Xu, Member IEEE and R. Kumar, Fellow IEEE
Dept. of Elec. & Comp. Eng.
Iowa State University, Ames, IA
syxu,rkumar@iastate.edu

Abstract— **Knowledge of the current system state is crucial to many discrete event systems (DESs) applications such as control, diagnosis and prognosis. Due to limited sensing capabilities, the current state information is generally not available and needs to be estimated. In this paper, we propose a novel *distributed* state estimation algorithm for discrete event plants. According to the proposed algorithm, local sites maintain and update local state estimates based on their local observations of the plant behavior and the observations of the plant behavior sent from the other sites over communication channels with delays. For efficiency of storage, redundant history information about the possible plant evolution is truncated each time a local state estimate is updated. At each local site, the truncation is performed independently requiring no synchronization among the sites. The state estimate maintained at each of the local sites is shown to remain finite regardless of whether the system can execute an unbounded sequence of unobservable events. It is also shown that the proposed algorithm is sound and complete, i.e., each local estimate always contains the true current states (soundness), and it only contains the reachable states of the traces which give rise to a same history of observations (as received from the plant and the other local sites) as does the one executed by the plant (completeness). Also the proposed algorithm can support an architecture in which there is no communication from a certain site to certain other sites. An illustrative example is provided to demonstrate the proposed distributed state estimation algorithm.**

Keywords: Discrete event systems, distributed state estimation, communication delay.

I. Introduction

In many on-line applications of discrete event systems (DESs) such as control, diagnosis and prognosis, it is crucial to know the current state of the system. Due to limited sensing capabilities, the current state information is generally not available and needs to be estimated.

The task of state-estimation for DESs has received considerable attention (see for example [1], [2], [3], [4]). The state estimation problem has also been studied in the setting of Petri nets, where the goal is to find the set of markings consistent with the observation sequence (see for example [5], [6], [7], [8], [9]). There are also some work on state estimation in the max-plus setting (see for example [10]) and for stochastic DESs (see for example [11], [12], [13]).

In [14], distributed state estimation problem for DESs was proposed. Each estimator updates its local estimate each time an observation occurs or an estimate sent by another estimator arrives. Due to the delay of communication, the local estimate and the received estimate may be "out of sync", and care was taken in fusing the two estimates to ensure the correctness of the estimation process. A key idea was that each estimator maintains not only an estimate of the most recent states, but also of the past states. To cope with the issue that the history of the estimates of the current and the past states could continue to grow unboundedly, a distributed synchronized truncation strategy was devised to discard the older portion of the state estimates on an ongoing basis. The algorithm ensured that only the unwanted portion of the older history got discarded, and the truncation was performed in a fashion so that a same amount of history got discarded by all estimators.

There are two problems with the distributed state estimation algorithm proposed in [14]. First, each site communicates the entire local state estimate to all the other sites. This burdens the communication network. In our proposed algorithm only the new observations are transmitted by one site to the others. Secondly, the algorithm presented in [14] requires a complex distributed synchronization strategy for the truncation of the unwanted history of local state estimates. No such synchronization is required in our proposed algorithm. Each site performs the truncation locally independently of the other sites.

We introduce the notions of soundness and completeness for a state estimation algorithm. Soundness refers to the property that at each time a local estimate contains the true current system states, whereas completeness refers to the property that a local estimate only contains the reachable states of the traces which give rise to a same history of observations (as received from the plant and the other sites) as does the one executed by the plant. It is shown that the proposed new algorithm possesses these properties. An important feature of the proposed algorithm is that the state estimate history maintained at each of the local sites can be represented using a finite graph regardless of whether the plant can execute an unbounded sequence of unobservable events. An example is provided to illustrate the proposed distributed state estimation algorithm.

II. Notation and Preliminaries

Given an event set Σ, we use Σ^* to denote the set of all event-traces over Σ, including the zero-length trace ϵ, and $\overline{\Sigma}$ to denote the set $\Sigma \cup \{\epsilon\}$. A subset $L \subseteq \Sigma^*$ is called a language over Σ. A trace $s \in \Sigma^*$ is a prefix of a trace

The research was supported in part by the National Science Foundation under the grants NSF-ECCS-0424048, NSF-ECCS-0601570, NSF-ECCS-0801763, and NSF-CCF-0811541.

$t \in \Sigma^*$ if for some trace $u \in \Sigma^*$, $t = su$. The prefix-closure of $L \subseteq \Sigma^*$, denoted $pr(L)$, is the set of all prefixes of the traces in L. L is called prefix-closed or simply closed if $pr(L) = L$. Given $\widehat{\Sigma} \subseteq \Sigma$, the operation $\uparrow_{\widehat{\Sigma}}$ is used to denote the projection of a trace over $\widehat{\Sigma}$, and is inductively defined as follows:

$$\epsilon \uparrow_{\widehat{\Sigma}} := \epsilon; \forall s \in \Sigma^*, \sigma \in \Sigma : s\sigma \uparrow_{\widehat{\Sigma}} := \begin{cases} s \uparrow_{\widehat{\Sigma}} \sigma & \text{if } \sigma \in \widehat{\Sigma} \\ s \uparrow_{\widehat{\Sigma}} & \text{otherwise} \end{cases}$$

A discrete event system is modeled as an untimed automaton G that is a tuple (X, Σ, α, X_0), where X is a set of states, Σ is a finite set of events, $\alpha : X \times \overline{\Sigma} \to 2^X$ is a state transition function, and $X_0 \subseteq X$ is a set of initial states. The language generated by G consists of the traces executable in G and is denoted $L(G)$. It is obvious that $L(G)$ is nonempty and closed.

A path of G is a sequence of transitions in form of $x_1\sigma_1x_2 \cdots \sigma_nx_n \in X(\overline{\Sigma}X)^*$, where $x_{i+1} \in \alpha(x_i, \sigma_i)$ for $i = 1, \cdots, n-1$. We use $\Pi(G)$ to denote the set of all the paths of G. Given a path $\pi = x_1\sigma_1x_2 \cdots \sigma_nx_n$, $tr(\pi) := \sigma_1 \cdots \sigma_n$ denotes the event sequence associated with π. $first(\pi) := x_1$ (resp., $last(\pi) := x_n$) is used to denote the first state (resp., last state) of π. A path $\pi' \in X(\overline{\Sigma}X)^*$ is a prefix (suffix and subpath, respectively) of π if there exist $i, j : 1 \leq i, j \leq n$ such that $\pi' = x_1\sigma_1 \cdots \sigma_ix_i$ ($\pi' = x_i\sigma_i \cdots \sigma_nx_n$ and $\pi' = x_i\sigma_i \cdots \sigma_jx_j$, respectively). We use $pr(\pi)$ ($suff(\pi)$ and $sub(\pi)$, respectively) to denote the set of all prefixes (suffixes and subpaths, respectively) of π. Note that $sub(\pi) = suff(pr(\pi)) = pr(suff(\pi))$.

Let $I = \{1, \ldots, m\}$ denote the index set of all sites. The event sequences executed by the plant are observed at the ith site through its observation mask: $M_i : \Sigma \to \Delta_i \cup \{\epsilon\}$, where Δ_i is the set of observed symbols at site-i. The observation mask can be inductively extended to a sequence of events as: $M_i(\epsilon) = \epsilon$; $\forall s \in \Sigma^*, \sigma \in \Sigma$, $M_i(s\sigma) = M_i(s)M_i(\sigma)$.

III. Distributed State Estimation Algorithm

In this section, we introduce the basic idea of the proposed distributed state estimation algorithm and also give its formal presentation.

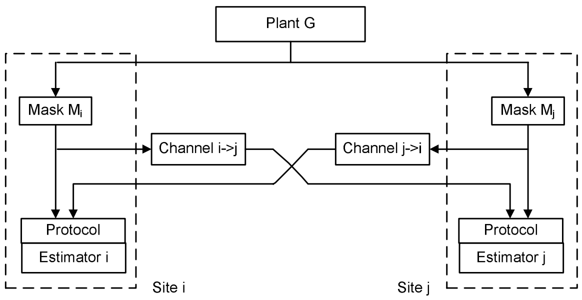

Fig. 1. Architecture of distributed state estimation

The architecture of the distributed state estimation is shown in Figure 1. Each local site consists of a state estimation algorithm (a state estimator) that computes the local state estimate based on the local observations of the plant behavior together with the observations of the plant behavior received from the other sites over loss-free, order-preserving channels that can introduce delay. Thus by the time the observation of an event is communicated by site-j to site-i, the plant may execute additional events. If the number of such events executed by the plant is bounded, then this is referred to as bounded-delay communication. It is assumed that each site knows the mask function of all the sites, and all local observations are communicated in the order of their occurrence. Also the proposed algorithm can support an architecture in which there is no communication from a certain site to certain other sites.

A. Basic Idea

We give the basic idea of the proposed distributed state estimation algorithm before giving its formal presentation. Each local site maintains the state estimate information consisting of a set of subpaths in the plant in form of a state estimate graph. The paths of such graph keep track of the recent relevant execution history of the plant, and are updated according to the local observations of the plant behavior together with the communicated observations of the plant behavior as sent by the other sites. The nodes of the state estimate graph are labeled by a subset of site indices to indicate the extent to which the plant history has evolved according to the information received from the plant and the other sites.

The update operations consist of:

- Extending the state estimate graph when a new observation from plant arrives,
- Relabeling the nodes of the state estimate graph when a new observation from plant or another site arrives, and
- Reducing and truncating the state estimate graph when a new observation from another site arrives.

The following example serves to illustrate the proposed algorithm.

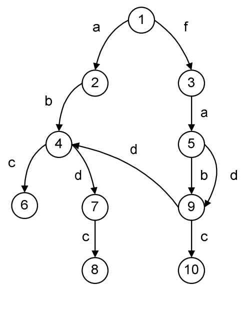

Fig. 2. Automaton model of plant

Example 1: Consider the plant $G = (X, \Sigma, \alpha, X_0)$ shown in Figure 2. Suppose there are two local sites, i.e., $I = \{1, 2\}$ with observation masks:

$$M_1(a) = a, M_1(b) = b, M_1(c) = c, M_1(d) = \epsilon, M_1(f) = \epsilon;$$

$$M_2(a) = a, M_2(b) = \epsilon, M_2(c) = c, M_2(d) = d, M_2(f) = \epsilon.$$

Suppose G executes the trace $abdc$.

It suffices to illustrate the state estimate information maintained by one of the sites, say site-1. Prior to any observation, site-1 initializes its state estimate graph to consist of all paths that produce no observation at site-1. The last state of each such path is labeled by an index $i \in I$ if the entire path produces no observation at site-i. A state labeled $i \in I$ corresponds to a possible current state of the plant according to the information received from site-i. Figure 3 shows the initial state estimate graph at site-1, which consists of the unobservable paths that can reach states 1 or 3. Since all paths are unobservable to both sites, all nodes of the state estimate graph are labeled with both site indices.

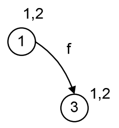

Fig. 3. Initial state estimate graph of site-1

When a new local observation arrives at site-1 directly from the plant, starting from the nodes labeled with the index-1, the state estimate graph of site-1 is extended by the traces that are indistinguishable from the new observation. Also, those site-1 labeled nodes where no such extension is possible are removed from site-1 state estimate graph. Then the index-1 labels are updated by propagating from the current locations to the last states of the extended paths.

For instance, when site-1 observes plant execution a, site-1 extends its state estimate graph with the traces observed as a. This consists of the trace a at state 1, and the traces $a, ad, add, addd$ at state 3, and is shown in Figure 4(a). Next the index-1 labels are propagated to nodes 2, 5, 9, 4, and 7 as shown in Figure 4(b).

Due to the communication delay between site-1 and 2, site-1 may observe the next plant execution b prior to receiving the first communication of a from site-2. When the local observation b is received, starting from the index-1 labeled nodes, site-1 extends its state estimate graph by the traces indistinguishable from b. It turns out that such extensions are not possible at states 4, 7, and 9, and thus these nodes are removed as shown in Figure 5(a), whereas the extension by b is possible at state 2 and the extensions by b, bd, bdd are possible at state 5, which are shown in Figure 5(b). Then index-1 labels are propagated to the last states of the extended paths, namely to states 4, 7, and 9.

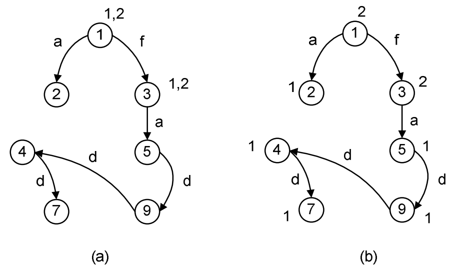

Fig. 4. Site-1 state estimate after local observation a

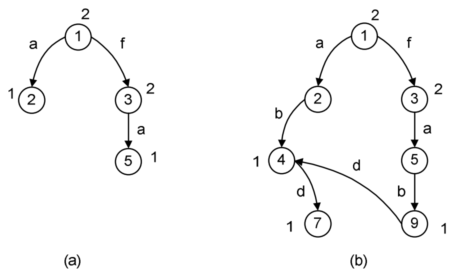

Fig. 5. Site-1 state estimate after local observation ab

When a delayed observation from another site j arrives, starting from the states labeled by the index-j, the paths of the state estimate graph that produce the same observation as reported by site-j are identified. Those j-labeled states where no such paths exist are removed from the local state estimate graph, and the index-j labels are propagated to the last states of the identified paths.

For instance, suppose after the local observation ab that came directly from plant, the communicated observation a from site-2 arrives at site-1. Note that state 1 and 3 are labeled with index-2, and at both states the possible paths that correspond to the received observation are the paths on the traces a and ab (recall b is unobservable to site-2). Accordingly, the index-2 labels are propagated to states 2, 4, 5, and 9 as shown in Figure 6(a). Now states 1 and 3 carry no labels since the estimate of the current state as determined by the local and communicated observations has evolved beyond those states. So maintaining such history in the state estimate graph is redundant and consequently the state estimate graph at site-1 is truncated by removing states 1 and 3 to obtain the new state estimate graph as shown in Figure 6(b).

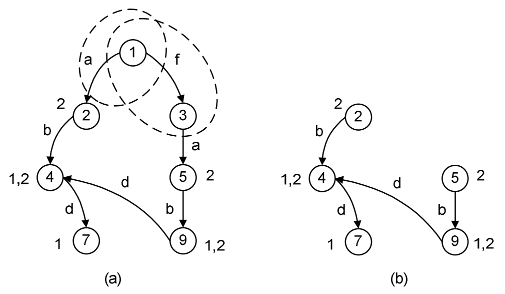

Fig. 6. Estimate after local obs. ab, and communicated obs. a

Remark 1: Note in the above, we have assumed that a local observation of a plant execution arrives prior to its communicated observation. The implication of this assumption is that the state estimate graph gets extended only when a new local observation arrives, whereas such an extension is not needed when a communicated observation arrives (only the labels need to be propagated and the estimate graph needs to be reduced and truncated). This set up can be easily modified to allow the more general case in which a communicated observation can arrive prior to a local observation.

B. Formal Presentation

Now we give a formal presentation of the proposed distributed state estimation algorithm.

Algorithm 1: Consider a plant $G = (X, \Sigma, \alpha, X_0)$ and a set of local sites $I = \{1, \cdots, m\}$, with M_i being the observation mask for site-i, that communicate among each other over loss-less and order-preserving channels. The state estimate graph maintained at site-i is a labeled graph consisting of a set of paths $\wp_i \subseteq \Pi(G)$ and a labeling function $lbl_i : X(\wp_i) \rightarrow 2^I$ that labels the states in $\wp_i$, denoted $X(\wp_i)$, by a subset of indices in I. The distributed state estimation algorithm defines the initial state estimate graph and its update operation each time a new observation from the plant or another site arrives.

1) Initialization:
   - Initialize paths: $\wp_i = \{\pi \mid first(\pi) \in X_0, M_i(tr(\pi)) = \epsilon\}$.
   - Initialize labels: $\forall x \in X(\wp_i), lbl_i(x) := \{i \in I \mid \exists \pi \in pr(\wp_i) : M_i(tr(\pi)) = \epsilon, last(\pi) = x\}$.
2) Update when observation $o \in \Delta_i$ from plant arrives:
   - Update paths: $\wp_i^{old} = \wp_i$, $\wp_i = \{\pi\sigma\pi' \mid \pi \in \wp_i^{old},\ i \in lbl_i(last(\pi)),\ M_i(\sigma) = o,\ M_i(tr(\pi')) = \epsilon\}$.
   - Update labels: $\forall \pi \in sub(\wp_i) : i \in first(\pi), M_i(tr(\pi)) = o$, $lbl_i(first(\pi)) \leftarrow lbl_i(first(\pi)) - \{i\}$, $lbl_i(last(\pi)) \leftarrow lbl_i(last(\pi)) \cup \{i\}$.
3) Update when observation $o \in \Delta_j$ from site-j arrives:
   - Update paths: $\wp_i^{old} = \wp_i$, $\wp_i = \wp_i^{old} - \{\pi \mid \forall \pi' \in sub(\pi) : j \in lbl_i(first(\pi')) \Rightarrow M_j(tr(\pi')) \neq o\}$.
   - Update labels: $\forall \pi \in sub(\wp_i) : j \in first(\pi), M_j(tr(\pi)) = o$, $lbl_i(first(\pi)) \leftarrow lbl_i(first(\pi)) - \{j\}$, $lbl_i(last(\pi)) \leftarrow lbl_i(last(\pi)) \cup \{j\}$.
   - Truncate paths: $\wp_i^{old} = \wp_i$, $\wp_i = \{\pi \in suff(\wp_i^{old}) \mid \exists \pi'\sigma\pi \in \wp_i^{old} : \forall x \in \pi', lbl_i(x) = \emptyset\}$.

The *estimate of the current state at site-i*, denoted $\widehat{X}_i \subseteq X$, is given by

$$\widehat{X}_i = \{x \in X(\wp_i) \mid i \in lbl_i(x)\}.$$

Remark 2: Compared with the distributed state estimation algorithm of [14], the algorithm in this paper is much simpler. First, each site communicates only its local observations. In contrast, in [14] each site communicated the entire current estimate set. Secondly, the truncation at each site is performed independently requiring no synchronization with the other sites. In contrast, the algorithm presented in [14] required a complex distributed synchronization strategy for a truncation to be performed.

Remark 3: Note when the communication delay of the channels among the sites is zero, each site will act as a centralized state estimator. On the other hand when the communication delay is infinite, each site will act as a decentralized state estimator.

IV. Illustrative Example

We revisit Example 1 to illustrate the distributed state estimation algorithm proposed in Algorithm 1. Supposing as before that the plant executes the trace $abdc$, we examine the following sequence of observations and the corresponding state estimate graphs.

1) Initialization: Since f is unobservable to both sites, the initial state estimate graph of site-2 is the same as that initialized at site-1 (as shown in Figure 3).
2) Both sites observe a from the plant: The state estimate graphs of the two sites after the local observation a are shown in Figure 4 and Figure 7 respectively. Note since b is unobservable to site-2, the observation of a is M_2-indistinguishable from the observation of ab. Notice that the index-2 labeled nodes in the site-2 state estimate graph are the states 2, 4, 5, and 9.

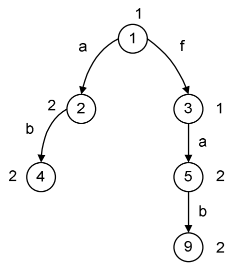

Fig. 7. Site-2 state estimate after local observation a

3) Site-1 observes b from the plant: The state estimate graph of site-1 after the local observation ab is shown in Figure 5.
4) Site-2 observes d from the plant: The state estimate graph of site-2 after the local observation ad is shown in Figure 8. This is obtained by extending the state estimate graph of Figure 7 by d or bd (both are M_2-indistinguishable to d) at index-2 labeled states 2, 4, 5, and 9, and propagating the index-2 labels to their d or bd reachable states. Note along the path $5d9$, 9 acquires the index-2 label, but along the path $5b9d4$, 9 cannot have the index-2 label (since along this path 9 is not a d- or bd-successor of 5). Hence to correctly extend the state estimate graph of Figure 7, two copies of 9 is required, only one of which is labeled by index-2. This is shown in Figure 8 where the first copy 9 is drawn with solid line and the other copy $9'$ with dashed line. (Note the extension of the paths requires a concatenation operation over two sets of paths, and the execution of the said concatenation operation will automatically create the necessary "copies" of the states.)
5) Site-1 (resp., site-2) receives the communicated observation a from site-2 (resp., site-1): The state estimate graph of site-1 after the local observation ab and the communicated observation a is shown in Figure 6,

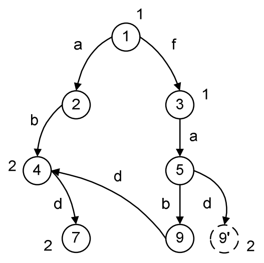

Fig. 8. Site-2 state estimate after local observation ad

whereas the state estimate graph of site-2 after the local observation ad and the communicated observation a is shown in Figure 9. Note that at site-2 the index-1 labels propagate from the states 1 and 3 (see Figure 8) to the states 2 and 5, rendering the labels for the states 1 and 3 to be empty. So these two states get removed in the truncation step, resulting in the state estimate graph of Figure 9.

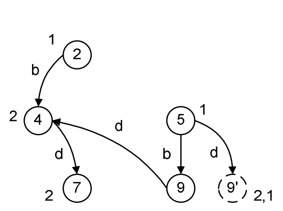

Fig. 9. Site-2 estimate after local obs. ad and communicated obs. a

6) Site-2 receives b from site-1: The state estimate graph of site-2 after the local observation ad and the communicated observation ab is shown in Figure 10. Note the observation of b is M_1-indistinguishable from those of b and bd. Accordingly the index-1 labels are propagated from states 2, 5, and $9'$ (see Figure 9) to the states 4, 7, and 9. Note no trace that is M_1-indistinguishable from b is defined at the index-1 labeled state $9'$ and it gets removed from the state estimate graph. Now states 2 and 5 carry empty labels and thus get removed in the truncation step.

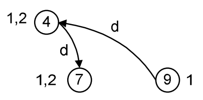

Fig. 10. Site-2 estimate after local obs. ad and communicated obs. ab

7) Both sites observe c from the plant: The state estimate graph of site-1 after the local observation abc and the communicated observation a, and the state estimate graph of site-2 after the local observation adc and the communicated observation ab are shown in Figure 11.

8) Site-1 receives d from site-2: The state estimate graph of site-1 after the local observation abc and the communicated observation ad is shown in Figure 12. First note that the states 2, 4, 5, and 9 are labeled by index-2 prior to the received observation (Figure 11). The trace bd (with $M_2(bd) = M_2(d)$) is feasible at the states 2 and 5, whereas the trace d is feasible at the states 4 and 9. Accordingly the index-2 labels are propagated to the states 4 and 7. Then the states 2, 5, and 9 carry empty labels and get removed in the truncation step. Also note that no trace that is M_2-indistinguishable from d is defined along the path $5b9c10$ that originates at an index-2 labeled state, and so this path gets removed from the state estimate graph. This yields the state estimate graph of Figure 12.

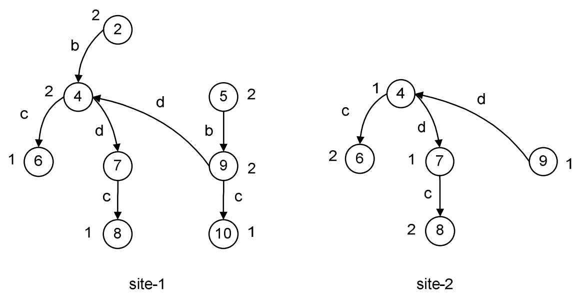

Fig. 11. Site-1 and Site-2 estimates after local obs. abc and adc respectively

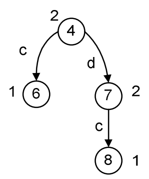

Fig. 12. Site-1 estimate after local obs. abc and communicated obs. ad

9) Site-1 (resp., site-2) receives c from site-2 (resp., site-1): The state estimate graph of site-1 after the local observation abc and the communicated observation adc, and the state estimate graph of site-2 after the local observation adc and the communicated observation abc is shown in Figure 13.

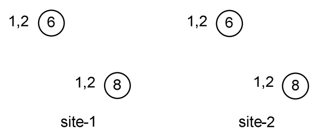

Fig. 13. Site-1 and Site-2 estimates after communicated obs. c

As can be seen both sites have an identical state estimate graph at this point. This is to be expected since no observations are pending to be received and so the estimate at each site should become the same as the one that a centralized estimator would construct.

V. Properties of the Proposed Algorithm

As mentioned above, the proposed distributed state estimation algorithm is simpler compared to the one given in [14]. We mention some of its additional properties in this section. We first define the set of possible observations at a site when a certain trace is executed by the plant. Without loss of generality, we assume that the observation symbols

$\{\Delta_i, i \in I\}$ are mutually disjoint. Otherwise we can rename an observation symbol of Δ_i by attaching to it a site-index (to indicate the site which makes that observation) to ensure that the observation symbols mutually disjoint. More precisely, for each $i \in I$, let $\Gamma_i := \{\delta_i | \delta \in \Delta_i\}$. Then the renamed observation symbols $\{\Gamma_i, i \in I\}$ will be mutually disjoint.

We inductively define the function $\mathcal{O}_i : L(G) \to 2^{(\bigcup_i \Delta_i)^*}$ that maps a plant trace to a set of possible observation sequences at a site-i as follows (here $s \in \Sigma^*, \sigma \in \Sigma$):

$$\begin{aligned}\mathcal{O}_i(\epsilon) &:= \{\epsilon\};\\ \mathcal{O}_i(s\sigma) &:= \{\nu M_i(\sigma)\mu \mid \nu \in \mathcal{O}_i(s), \mu \in 2^{(\bigcup_{j\in I-\{i\}} \Delta_j)^*},\\ &\qquad \forall j \in I - \{i\} : (\nu\mu) \uparrow_{\Delta_j} \in pr(M_j(s\sigma))\}.\end{aligned}$$

Note the observations of $s\sigma$ at site-i are of the form $\nu M_i(\sigma)\mu$, where ν is an observation of s at site-i, and the projection of $\nu\mu$ over the observation symbols of a site-j ($j \in I - \{i\}$) is simply a prefix of the local observations $M_j(s\sigma)$ of site-j.

We first show that the state estimate graph remains finite.

Theorem 1: Consider the setting of Algorithm 1. Then for each $i \in I$, the set of paths of the state estimate graph in $\wp_i$ is always a regular set.

Remark 4: The result of Theorem 1 is important since unlike many of the prior works (such as [14], [8], [9]), it does not require the absence of unbounded sequence of unobservable events in G, which was used in the said prior works to ensure the finiteness of the state estimate graph.

We next establish the soundness and completeness of the proposed distributed state estimation algorithm. Algorithm 1 computes, at each site-i, an estimate of the current state $\widehat{X}_i$, depending on an observation sequence in $\mathcal{O}_i(L(G))$ received at site-i. Thus the estimate of the current state can be viewed as a map $\widehat{X}_i : \mathcal{O}_i(L(G)) \to 2^X$. Then the soundness and completeness of a state estimate are defined as follows.

Definition 1: Consider the setting of Algorithm 1. Then for each $s \in L(G)$ and $\nu \in \mathcal{O}_i(s)$, $\widehat{X}_i(\nu)$ is said to be

- *sound* if $\alpha(X_0, s) \subseteq \widehat{X}_i(\nu)$, and
- *complete* if $\widehat{X}_i(\nu) \subseteq \bigcup_{t\in L(G):\nu\in\mathcal{O}_i(t)} \alpha(X_0, t)$.

Note soundness refers to the fact that the true states are always included in the estimate, whereas completeness refers to the fact that no redundant states are included in the estimate.

The following theorem shows that the distributed state estimation algorithm of Algorithm 1 is sound and complete.

Theorem 2: Consider the setting of Algorithm 1. Then for each $i \in I$ and $\nu \in \mathcal{O}_i(L(G))$, the state estimate $\widehat{X}_i(\nu)$ as computed by Algorithm 1 is sound and complete.

The proof of Theorem 2 is omitted due to the limit of space.

VI. Conclusion

In this paper we proposed a new distributed state estimation algorithm for discrete event systems. Under the proposed algorithm, each local site maintains a state estimate graph consisting of a set of labeled plant subpaths that gets updated based on the local observations of the plant behavior and the observations of the plant behavior received from the other sites communicated over channels with delay. Each time the state estimate graph is updated, redundant estimation history is discarded independently at each site without requiring any synchronization with the other sites. The state estimate graph remains finite regardless of whether the plant can execute an unbounded sequence of unobservable events. It is also shown that the proposed algorithm is sound and complete, i.e., the true plant states are always included in the local estimates, while only the states that are reachable by the traces indistinguishable from the one observed are included in the local estimates. Also the proposed algorithm can support an architecture in which there is no communication from a certain site to certain other sites.

References

[1] C. M. Özveren and A. S. Willsky, "Observability of discrete event dynamical systems," *IEEE Transactions on Automatic Control*, vol. 35, no. 7, pp. 797–806, 1990.

[2] ——, "Tracking and restrictability in discrete event dynamical systems," *SIAM Journal of Control and Optimization*, vol. 30, no. 6, pp. 1423–1446, 1992.

[3] S. Bose, A. Patra, and S. Mukhopadhyay, "On observability with delay: antitheses and syntheses," *IEEE Transactions on Automatic Control*, vol. 39, no. 4, pp. 803–806, 1994.

[4] C. Cao, F. Lin, and Z.-H. Lin, "Why event observation: observability revisited," *Journal of Discrete Event Dynamical Systems: Theory and Applications*, vol. 7, no. 2, pp. 127–150, 1997.

[5] I. Rivera-Rangel, L. Aguirre-Salas, A. Ramirez-Trevino, and E. Lopez-Mellado, "A Petri net scheme for DES state estimation," in *Proceedings of IEEE Conference on Decision and Control*, Sydney, Australia, 2000, pp. 2260–2265.

[6] A. Bourjij and D. Koenig, "An original Petri net state estimation by a reduced Luenberger observer," in *Proceedings of American Control Conference*, San Diego, USA, 1999, pp. 1986–1989.

[7] A. Giua, D. Corona, and C. Seatzu, "State estimation of λ-free labeled Petri Nets with contact-free nondeterministic transitions," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 15, pp. 85–108, 2005.

[8] Y. Ru and C. Hadjicostis, "State estimation in discrete event systems modeled by labeled Petri nets," in *Proceedings of IEEE Conference on Decision and Control*, San Diego, USA, 2006, pp. 6022–6027.

[9] M. P. Cabasino, A. Giua, and C. Seatzu, "Marking estimation of Petri nets with arbitrary transition labeling," in *Proceedings of the 1st IFAC Workshop on Dependable Control of Discrete Systems*, Pairs, France, June 2007, pp. 109–114.

[10] L. Hardouin, B. C. C. A. Maia, and M. Lhommeau, "Observer design for max-plus linear systems," in *Proceedings of the 1st IFAC Workshop on Dependable Control of Discrete Systems*, Pairs, France, June 2007, pp. 195–200.

[11] J.Lunze and J.Scheröder, "State observation and diagnosis of discrete-event systems decribed by stochastic automata," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 11, pp. 319–369, 2001.

[12] E. Fabre, A. Benveniste, C. Jard, L. Ricker, and M. Smith, "Distributed state reconstrucation for discrete event systems," in *Proceedings of IEEE Conference on Decision and Control*, Sydney, Australia, 2000, pp. 2252–2257.

[13] A. Aghasaryan, E. Fabre, A.Benveniste, R. Boubour, and C. Jard, "Fault detection and diagnosis in distributed systems: an approach by partially stochastic Petri nets," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 8, pp. 203–231, 1998.

[14] W. Qiu and R. Kumar, "A protocol for distributed state estimation in discrete event systems," in *Proceedings of the 1st IFAC Workshop on Dependable Control of Discrete Systems*, Pairs, France, June 2007, pp. 97–102.