Supervisor Synthesis for Networked Discrete Event Systems with Delays against Non-FIFO Communication Channels

Yuting Zhu, Liyong Lin, Ruochen Tai, Rong Su

Abstract— **In this work, we study the problem of supervisory synthesis for networked discrete event systems against non-FIFO communication channels with bounded delays. Both the observation and control communication channels are represented by finite state automata under the assumption that all communication delays are bounded, the resilient networked supervisor synthesis problem is then reduced to supervisor synthesis for non-deterministic automata. We firstly analyze the structure of networked discrete-event systems with delays, and then describe the message transmission process through the communication channels which connect the plant and the supervisor. The assumption of the networked discrete event systems: 1)The buffer size of the communication channels is limited; 2)There is a maximum delay boundary for each event in the communication channel; 3)The number of multiple copies of the same event in the communication channel is limited. The content of the observation and control channel will be represented by the timed finite state automata. Finally, an example will illustrate how the networked resilient supervisor will be synthesized.**

Bounded delays, non-FIFO channel, networked discrete-event systems, supervisory control

I. Introduction

In networked discrete event systems, the plant and the supervisor are remotely connected by communication channels. The observation messages from the plant and the control command messages issued by the supervisor may experience non-negligible transmission delays, due to network flow jam or other reasons. Therefore, maintaining the safe operation of a supervisory control system under communication delays is an important issue to address.

A natural model for networked supervisory control is described by the timed discrete-event systems [1], [2], [3]. In Balemi's work [4], [5], the solution for the networked supervisory control problem, with bounded communication delays and under full observation, depends on the existence of a controllable, delay-insensitive language. The problem of networked control under partial observation is investigated in [6], [7], [8]. The assumption in these works is that all the controllable events are observable and each controllable event is permitted to occur only when the control command issued by the delay-robust non-blocking supervisor arrives after some bounded communication delays. Lin's work [9] introduces two observation mappings that represent observation delays and observation losses, respectively. Based on the two mappings, a necessary and sufficient condition for the existence of a networked supervisor is found based on the definition of network observability and network controllablity. The control command issued by the supervisor is in general not unique when it observes the same string generated by the plant. Shu [10] considered a networked control problem with bounded communication delays by specifying the behaviour of the supervised system with a minimum required language and a maximum admissible language. An algorithm is provided to check the existence of a supervisor and then a state-based control policy is implemented. There are also extensions to decentralized networked supervisory control in [11], [12], [13].

This paper is mainly inspired by Lin's work [9] on the supervisor synthesis problem for networked discrete event systems with communication delays and losses. In that work, the composition of two observation mappings are used to describe the observation delays and losses. The notion of network observability is extended from the original definition of observability and an observer is constructed to compute state estimates under observation delays. In addition, there is a restriction in Lin's work that considers the communication channels to be FIFO. The problem we consider in this work is described as follows. For a given plant G that is remotely controlled by a supervisor through non-FIFO communication channels with bounded delays, how shall we compute a resilient supervisor S against the network delays so that the safe operation of the plant G can be maintained. Different from the approach in [9], we shall construct two (flexible) finite state automata to model the observation channel and the control channel with bounded delays separately; after a dedicated transformation of the plant and specification, the networked synthesis problem is then converted to the non-deterministic supervisory control problem, due to the non-determinism in the transformed plant caused by the non-FIFO channels. The contributions of this work are as follows.

1) This paper proposes a new and flexible method to model the communication channels with bounded communication delays as finite state automata. In particular, messages sent over the same channel can have different transmission delays, but the transmission delays are upper bounded by the same delay bound as determined by the channel. This removes an unnecessary restriction imposed in [14] which requires all messages to be delayed by the same number of time steps.

The authors are affiliated with School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. Email: yuting002@e.ntu.edu.sg, liyong.lin@ntu.edu.sg ruochen001@ntu.edu.sg, rsu@ntu.edu.sg. This work is financially supported by Singapore National Research Foundation via Delta-NTU Corporate Lab Program (DELTA-NTU CORP LAB-SMA-RP2 SU RONG M4061925.043) and from Singapore Ministry of Education Tier 1 Academic Research Grant (M4011982 RG91/18-(S)-SU RONG (VP)) are gratefully acknowledged.

As an instantiation of our modelling approach, we construct non-deterministic finite state models for non-FIFO channels with bounded communication delays.

2) We provide a procedure for the transformation of the plant and the specification, by constructing a networked command execution automaton, which allows us to reduce the resilient networked supervisor synthesis problem to the problem of supervisor synthesis for non-deterministic plants[1] [15]. As a result, we do not need to define notions such as network controllability or network observability with non-FIFO channels. Consequently, the definitions of networked controllability and networked observability are reduced to standard definitions of controllability and observability [15].

Compared with [16], in this paper we assume non-FIFO communication channels instead of FIFO communication channels and will only consider network delays while ignoring communication losses. In reality, many communication networks allow messages to overtake each other, making it possible that events are not necessarily observed in the same order as they take place in the plant. Therefore, the non-FIFO property of the channel is arguably more realistic. The non-FIFO property of the communication channels brings the new feature that the transformed plant is in general non-deterministic. Thus, we need to deal with supervisor synthesis for non-deterministic plants.

The paper is organized as follows. First, we provide some preliminaries that are necessary for understanding this work in Section II. Then, in Section III, we describe the structure of networked control systems based on the discrete time framework and present a new construction that allows us to model non-FIFO communication channels with bounded communication delays as finite state automata. In Section IV, we provide the procedure for the transformation of plant and specification. The resilient networked supervisor synthesis problem is then reduced to supervisor synthesis for non-deterministic automata. An example is used to get one synthesized supervisor in Section V. Finally, conclusions and some future research directions are discussed in Section VI.

II. Preliminaries

We assume the reader to be familiar with the basic theories of finite automata and supervisory control [17]. We shall recall some additional notations and terminologies in the following.

Let Σ be a finite alphabet. A (non-deterministic) finite state automaton G over alphabet Σ is a 5-tuple $(Q, \Sigma, \delta, Q_0, Q_m)$, where Q is the finite set of states, $\delta \subseteq Q \times \Sigma \times Q$ the transition relation, $Q_0 \subseteq Q$ the initial state set and $Q_m \subseteq Q$ is the marked state set. When $|Q_0| = 1$ and δ is a partial function, we then say G is deterministic. For two finite state automata $G_1 = (Q_1, \Sigma_1, \delta_1, Q_{1,0}, Q_{1,m}), G_2 = (Q_2, \Sigma_2, \delta_2, Q_{2,0}, Q_{2,m})$, we write $G := G_1 \| G_2$ to denote their synchronous product. In this work, whenever we talk about a plant, we mean a (non-deterministic) finite state automaton. Let $G = (Q, \Sigma, \delta, Q_0, Q_m)$ denote the plant. The alphabet Σ is partitioned into Σ_c and Σ_{uc}, representing the sets of controllable and uncontrollable events, respectively; the alphabet Σ is also partitioned into Σ_o and Σ_{uo}, representing the sets of observable and unobservable events, respectively. A (finite state) supervisor over control constraint (Σ_c, Σ_o) is a deterministic finite state automaton $S = (X, \Sigma, \xi, x_0, X_m)$ that satisfies the controllability and observability constraints:

i) (controllability) for any state $x \in X$ and any uncontrollable event $\sigma \in \Sigma_{uc}$, $\xi(x, \sigma)!$,

ii) (observability) for any state $x \in X$ and any unobservable event $\sigma \in \Sigma_{uo}$, $\xi(x, \sigma)!$ implies $\xi(x, \sigma) = x$,

A specification ϕ often specifies a set of bad states to avoid in the plant. Let $Q_{bad} \subseteq Q$ denote the set of bad states to avoid in the plant. Then, several existing tools like TCT [18], SuSyNA and Supremica [19] can be used for synthesizing supervisors S such that no state $(x, q_{bad}) \in X \times Q_{bad}$ can be reached in the closed-loop system $S\|G$ and $S\|G$ is non-blocking, i.e. every reachable state is co-reachable (that is., can reach some marked state). Supervisor synthesis tools like TCT and Supremica are targeted for the supervisor synthesis for deterministic plants and SuSyNA can in addition deal with supervisor synthesis for non-deterministic plants. The mechanism of control of S over G is via sending, receiving and executing control commands. The set of all possible control commands is denoted by $\Gamma = \{\gamma \subseteq \Sigma \mid \gamma \supseteq \Sigma_{uc}\}$; each $\gamma \in \Gamma$ specifies the events $\sigma \in \gamma$ that are allowed to happen.

III. Networked control for discrete event systems

The networked control architecture for discrete event systems is shown in Fig. 1. The system consists of the plant G, the supervisor S, the observation channel through which the observation messages transfer from the plant to the supervisor, the control channel through which the control command messages transfer from the supervisor to the plant, and the networked command execution automaton which picks an event from the received control command to execute. These five components interact with each other and form the closed-loop control system. The event execution mechanism of the plant G and the message transmission process of the observation channel and the control channel (with delays) will be explained in the following.

In the closed-loop networked control system shown in Fig. 1, the first assumption is that the time spent for the execution of an event is much greater than the spent in the supervisor for it to receive an event signal and send a control command. Therefore, we assume ticks are only consumed in the execution of plant events. This assumption is reasonable since G is a physical system and event execution usually requires realistic operation in physical world while S is often

[1]The tool **Su***pervisor* **Sy***nthesis for* **N***on-deterministic* **A***utomata* (SuSyNA), which can be used for the synthesis of deterministic supervisors for non-deterministic plants, is available for download at https://www.ntu.edu.sg/home/rsu/Downloads.htm.

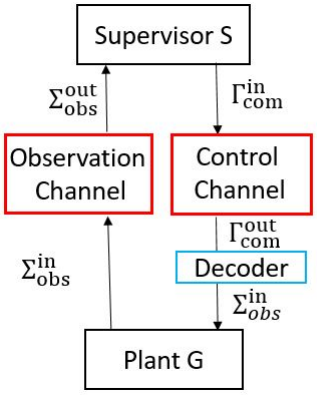

Fig. 1. Networked control structure for discrete event systems

implemented by a digital controller which can process digital information very quickly. At the initial state, the supervisor S sends the control command Γ to the plant G, then the plant will execute one of the events contained in the control command, i.e. execute some $\sigma \in \Gamma$.

The plant G will continuously hold the old control command until it observes the occurrence of a newly issued control command. According to the controllablity and observability of the event executed in the plant G, the execution mechanism for the next event of plant G can be divided into the following cases:

(1) σ is unobservable ($\sigma \in \Sigma_{uo}$): after the plant G executes event σ, no execution message will be pushed into the observation channel. Since the supervisor S observes nothing, it will hold on and not issue any new control command. The plant then follows the old control command.

(2) σ is observable ($\sigma \in \Sigma_o$): after the plant G executes event σ, the execution message will be pushed into the observation channel and might undergo some delays until it finally gets popped out of the observation channel. Once the supervisor S observes the occurrence of σ, it will issue a new control command, and the newly issued control command will pass through the control channel, possibly with some delays. Any uncontrollable event is always allowed to occur, as long as it is defined in the plant, even if no control command has been received. This effectively removes all the uncontrollable events from a control command, which is only used to control the controllable events. While the classical, non-networked supervisory control framework encapsulates all the uncontrollable events within a control command and thus also amounts to only control the controllable events, the design choice of removing uncontrollable events from a control command looks more natural in the networked supervisory control setup as the execution of uncontrollable events are not supposed to be delayable by communication delays, in contrast to control commands. Thus, in the remaining of this work we redefine the set of control commands to be $\Gamma = \{\gamma \subseteq \Sigma \mid \gamma \supseteq \Sigma_{uc}\}$.

A. Automata for the observation channel and control channel

Observation Channel: The observation channel is assumed to be non-FIFO. Whenever the plant executes an observable event $\sigma \in \Sigma_o$, it sends message m_σ, indicating the occurrence of σ in the plant, over the observation channel; the event of sending this message is denoted by $\sigma^{in} \in \Sigma_{obs}^{in}$, which is a relabelled copy of Σ_o with superscript in and subscript obs. The event σ^{in} updates the content of the observation channel by adding message m_σ. The event of receiving message m_σ by the supervisor is denoted by $\sigma^{out} \in \Sigma_{obs}^{out}$, which is a relabelled copy of Σ_o with superscript out and subscript obs, indicating the occurrence of $\sigma^{in} \in \Sigma_{obs}^{in}$ (some moment ago), and updates the content of the observation channel by removing message m_σ. If this message is delayed during the transmission process, the content of the observation automata will be updated with the event transition labelled by $tick$.

Control Channel: The control channel is also assumed to be non-FIFO and works quite similar to the observation channel. The event that the supervisor sends a (control command) message m_γ over the control channel is denoted by $\gamma^{in} \in \Gamma_{com}^{in}$, where Γ_{com}^{in} is a copy of Γ with superscript in and subscript com. The event γ^{in} updates the content of the control channel by adding message m_γ. The plant can receive a control command message from the control channel. The event of receiving message m_γ by the plant is denoted by $\gamma^{out} \in \Gamma_{com}^{out}$, where Γ_{com}^{out} is a copy of Γ with superscript out and subscript com, and updates the content of the control channel by removing message m_γ. If this message is delayed during the transmission process, the content of the observation automata will be updated with the transition labelled by $tick$.

Communication Delays: We denote the fixed delay bounds of the observation channel and control channel as num^o and num^c, respectively ($num^o, num^c \in N$), which corresponds to the maximum number of time steps messages can stay in the channels.

Bounds on Channel Size: In this work, the communication channels are assumed to be of finite capacity, in order for us to deal with finite systems. We use the integer N^{obs} to denote the maximum number of stored messages in the observation channel and use the integer N^{com} to denote the maximum number of stored messages in the control channel, respectively. When the storage of the channel becomes full, it can not store any additional messages. Any other message will be thrown away.

Automata Models: The state space of the observation channel automaton is denoted as Q^{obs}. The amount of time steps a message will stay in the observation channel, i.e., the time-to-leave of a message, is an integer $0 \leq i \leq num^o$ that is artificially attached to the message, for us to distinguish between the same messages with different time-to-leaves. For each $\sigma \in \Sigma_o$, each $0 \leq i \leq num^o$ and each $k \geq 0$, let $((m_\sigma, i), k)$ denote the multi-set that consists of k copies of (m_σ, i). We define $|((m_\sigma, i), k)| = k$,

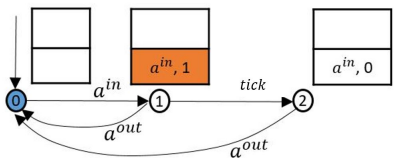

Fig. 2. Single Event a with Bounded Delay in the Observation Channel

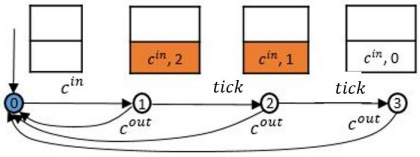

Fig. 3. Single Event c with Bounded Delay in the Observation Channel

which is the cardinality of the multi-set. Then, the content of the observation channel can be represented by $s = \{((m_\sigma, i), k) \mid \sigma \in \Sigma_o, 0 \leq i \leq num^o, k \geq 0\}$ with the constraint that $|s| \leq N^{obs}$; here $|s|$ is defined to be the sum of the cardinality of each multi-set in s. The set of all possible contents of the observation channel is denoted by S_{obs}. Then, we have $Q^{obs} = \{q^s \mid s \in S_{obs}\}$. The transition relation $\delta^{obs} \subseteq Q^{obs} \times (\Sigma^{in}_{obs} \cup \Sigma^{out}_{obs} \cup \{tick\}) \times Q^{obs}$ can be defined, in a way that is similar to the partial transition function of [16] for FIFO observation channels. The observation channel automaton is then given by $G_{obs} = (Q^{obs}, \Sigma^{in}_{obs} \cup \Sigma^{out}_{obs} \cup \{tick\}, \delta^{obs}, q^{\{\}})$, where $q^{\{\}} \in Q^{obs}$ is the initial state that corresponds to the empty observation channel. Similarly, the control channel automaton is given by $G_{com} = (Q^{com}, \Gamma^{in}_{com} \cup \Gamma^{out}_{com} \cup \{tick\}, \delta^{com}, q^{\{\}})$. We shall remark that, due to the non-FIFO feature, δ^{obs} and δ^{com} are in general non-deterministic.

We denote the control channel automaton as $G_{com} = (Q^{com}, \Sigma^{com}, \delta^{com}, q^{com}_0, Q^{com}_m)$, The state space of the control channel automata is denoted as Q^{com}. Let us denote the number of stored messages in the control command channel is i. For each $\gamma \in \Gamma$, each $0 \leq i \leq num^c$ and each $k \geq 0$, let $\{((m_\gamma, i), k)\}$ denote the multi-set that consists of k copies of (m_γ, i). We define $|\{((m_\gamma, i), k)\}| = k$, which is the cardinality of the multi-set s. Then, the content of the control channel can be represented by $s = \{((m_\gamma, i), k) \mid \gamma \in \Gamma, 0 \leq i \leq num^c, k \geq 0\}$ with the constraint that $|s| \leq N^{com}$; here $|s|$ is defined to be the sum of the cardinality of each multi-set in $s \in M$. The set of all possible contents of the control channel automaton is denoted by S_{com}. Then, we have $Q^{com} = \{q^s \mid s \in S_{com}\}$. The transition relation $\delta^{com} \subseteq Q^{com} \times (\Sigma^{com} \cup \{tick\}) \times Q^{com}$ is defined.

All events are controllable and observable. We assume all events $\Sigma = \Sigma_c = \Sigma_o = \{a, b, c\}$. That is to say, $\Sigma^{in}_{obs} = \{a^{in}, b^{in}, c^{in}\}$ and $\Sigma^{out}_{obs} = \{a^{out}, b^{out}, c^{out}\}$. The buffer size for the observation channel is two; the maximum delay bound for event a is one, and for b and c is two. The number of limited multiple event set for event b is two and

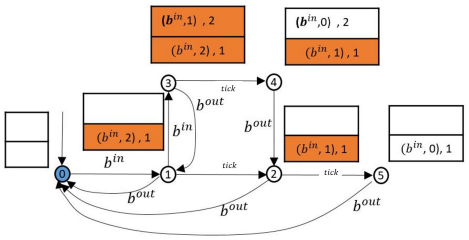

Fig. 4. Multiple Event b Set with Bounded Delay in the Observation Channel

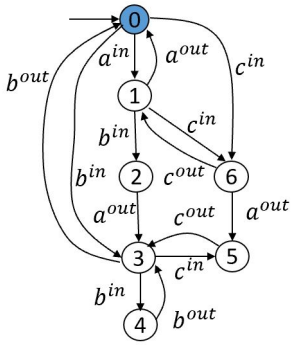

Fig. 5. Observation Channel Automata

for event a, c is one, and then we can easily construct the observation channel automata by the composition operation of the single event automata. We can clearly see from Fig. 2 that transferring from the initial state 0 to state 1, the observable event a^{in} entering the observation channel with the maximum one delay bound; Then the content of the observation channel will contain event pair $(m_\alpha, 1)$, message m_α may exit from the observation channel immediately with the occurrence of a^{out} (corresponding to the event transition from state 1 to state 0) or stay in the observation channel until reach the maximum delay bound then be popped out the channel forcibly (corresponding to the process that event transition from state 1 to state 2, followed by the transition back to state 0). Similarly, we construct the automata model for event b and c. The multiple event set b with different delay bounds is constructed in the same way and shown in Fig. 4.

We now create the (deterministic) networked command execution automata (Decoder). D^{net} which shows how a control command is executed. Intuitively, after the plant receives a control command γ, it needs to choose an arbitrary event $\sigma \in \gamma$ to fire. After that, we create all possible control messages by creating an event for each possible control decision recursively by constructing the networked command execution automata (Decoder) D^{net}. We denote the decoder automaton as $D^{net} = (Q^{D^{net}}, \Sigma^{D^{net}}, \delta^{D^{net}}, q_0^{D^{net}}, Q_m^{D^{net}})$, where $Q^{D^{net}} = \{q^\gamma \mid \gamma \in \Gamma\} \cup \{q_{wait}\}$, $\Sigma^{D^{net}} = \Gamma^{out}_{com} \cup \Sigma^{in}_{obs}$, $q_0^{D^{net}} = q_{wait}$ and $Q_m^{D^{net}} = \{q_{wait}\}$. $\delta^{D^{net}} : Q^{D^{net}} \times \Sigma^{D^{net}} \to Q^{D^{net}}$ is defined as follows.

1) for any $\gamma \in \Gamma$, $\delta^{D^{net}}(q_{wait}, \gamma^{out}_{com}) = q^\gamma$.
2) for any q^γ, if $\sigma \in \Sigma_{uo} \cap \gamma$, $\delta^{D^{net}}(q^\gamma, \sigma) = q^\gamma$.

3) for any q^{γ}, if $\sigma \in \Sigma_o \cap \gamma$, $\delta^{D^{net}}(q^{\gamma}, \sigma^{in}) = q_{wait}$.
4) no other transitions are defined

IV. Supervisor Synthesis

After we have finished the modelling of the communication channels, we now need to obtain a transformation of the plant and specification, so that the synthesis problem can be reduced to the supervisor synthesis for non-deterministic plants.

Transformation of Plant: The overall plant consists of, and thus is the synchronous product of, four different components. That is, it consists of a relabeled plant G_{mod}, two automata G_{obs}, G_{com} for the communication channels and a networked command execution automaton D^{net}. Intuitively, each automata for the communication channel interacts with both the plant and supervisor; from the supervisor's point of view, only those events involving interaction with the supervisor are treated as observable.

We denote the overall plant as G^{all}, which is the composition $G^{all} = G_{mod} \| D^{net} \| G_{obs} \| G_{com}$. We only need to explain the construction of G_{mod} to complete the description of G^{all}. Let $G = (Q, \Sigma, \delta, q_0)$ denote the original plant. We denote the relabeled plant as $G_{mod} = (Q^{mod}, \Sigma^{mod}, \delta^{mod}, q_0^{mod})$, where we have $Q^{mod} = Q$, $\Sigma^{mod} = \Sigma_{uo} \cup \Sigma_{obs}^{in}$ and $q_o^{mod} = q_0$. For any $\sigma \in \Sigma_{uo}$, $\delta^{mod}(q, \sigma) = q'$ iff $\delta(q, \sigma) = q'$. For any $\sigma \in \Sigma_o$, $\delta^{mod}(q, \sigma^{in}) = q'$ iff $\delta(q, \sigma) = q'$. That is, if the label of a transition is an observable event $\sigma \in \Sigma_o$ in the original plant G, the transition will be relabeled by σ^{in} in G_{mod}.

Transformation of Specification: It is straightforward to transform the specification. Suppose we would like to enforce the state avoidance property (in addition to non-blockingness property). Let $Q_{bad} \subseteq Q$ denote the set of bad states to avoid in the original plant. Then, the new specification ϕ^{all} will specify the avoidance of Q_{bad} in G_{mod} states.

Theorem 1: In the paradigm of standard Ramadge Wonham, $L(S) \subseteq L(S/G)$. $V(s)$ is the subset of all the controllable events including in the channel. Assuming that there is a string s that belongs to the controllability. We need to prove that the equavalence relationship between transferred supervisor under the non-delay condition. There is a reference saying that there is another existence of the new supervisor that will satisfy the requirement of the new standard definition of the the formulation part.

V. Case Study

In this section, we will see how the networked command execution automata interact with observation channel automata and control channel automata.

An illustrating example will be demonstrated in this section. Consider the plant G shown in Fig. 6. Let $\Sigma_o = \Sigma_c = \Sigma = \{a, b, c\}$. We assume the capacity of each communication channel is equal to 1 ($N^{obs} = N^{com} = 1$), and the delay bound for each channel is 1 as well ($num^o = num^c = 1$). To reduce the size of communication automata, we assume that during the message transmission, event a of the observation channel and control command $\{a, c\}$ of

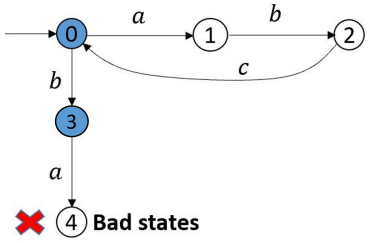

Fig. 6. The System Plant G

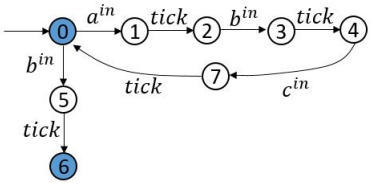

Fig. 7. The Transformed System Plant G_{mod}

the control channel will get delayed. That is, there are three observation channels, each for an observable event, and each with its own delay bound. Similarly, the same remark is applicable to control channel automata.

The construction of the overall plant G^{all} requires four automata $G_{mod}, G_{obs}, G_{com}$, and D^{net}, which are shown in Fig. 7 and Fig. 10 - Fig. 11. Then, we can manually compute the non-deterministic networked supervisor S_{net} in Fig. 13. Actually, from the final result of supervisor, the γ^{in} is the control command issued by the supervisor and σ^{out} is the observable event received by the supervisor. The closed-loop behaviour follows the composition result of plant G_{mod} and supervisor S_{net}, events in Σ_{obs}^{in}, Σ_{obs}^{out}, Γ_{com}^{in} and Γ_{com}^{out} will be transformed into the elements in the tuple separately.

VI. Conclusion

In this work, we have provided a new construction that reduces the networked supervisory control problem under non-FIFO assumption to the problem of supervisor synthesis for non-deterministic plants. Our current research work can

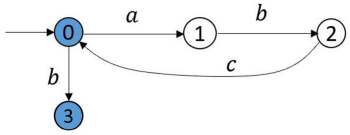

Fig. 8. Specification H

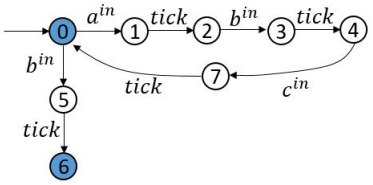

Fig. 9. The Modified Specification H

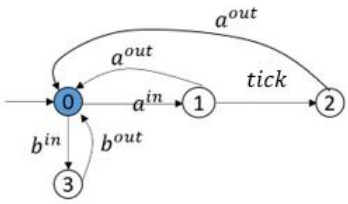

Fig. 10. Observation Channel Automata G_{obs}

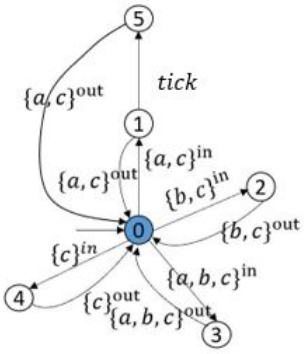

Fig. 11. Control Channel Automata G_{com}

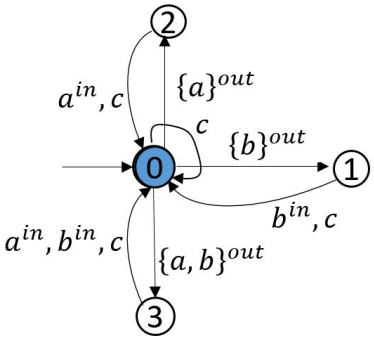

Fig. 12. Networked Command Execution Automata D^{net}

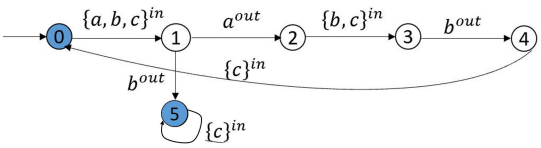

Fig. 13. Synthesized Networked Supervisor S_{net}

be formulated and fit into the generalized framework for networked systems. The synthesis problem under FIFO assumption of the communication channels has been discussed in [16]; Finally, it can be extended to other control architectures naturally, e.g., distributed, hierarchical and decentralized networked supervisory control.

REFERENCES

[1] Bertil A Brandin and W Murray Wonham. Supervisory control of timed discrete-event systems. *IEEE Transactions on Automatic Control*, 39(2):329–342, 1994.

[2] Aida Rashidinejad, Michel Reniers, and Lei Feng. Supervisory control of timed discrete-event systems subject to communication delays and non-fifo observations. *IFAC-PapersOnLine*, 51(7):456–463, 2018.

[3] F Lin and WM Wonham. Supervisory control of timed discrete-event systems under partial observation. *IEEE Transactions on Automatic Control*, 40(3):558–562, 1995.

[4] S Balemi and UA Brunner. Supervision of discrete event systems with communication delays. In *1992 American Control Conference*, pages 2794–2798. IEEE, 1992.

[5] Silvano Balemi. Input/output discrete event processes and communication delays. *Discrete Event Dynamic Systems*, 4(1):41–85, 1994.

[6] S-J Park and J-T Lim. Robust and nonblocking supervisory control of nondeterministic discrete event systems using trajectory models. *IEEE Transactions on Automatic Control*, 47(4):655–658, 2002.

[7] Seong-Jin Park and Kwang-Hyun Cho. Delay-robust supervisory control of discrete-event systems with bounded communication delays. *IEEE Transactions on Automatic Control*, 51(5):911–915, 2006.

[8] Seong-Jin Park and Kwang-Hyun Cho. Supervisory control of discrete event systems with communication delays and partial observations. *Systems & control letters*, 56(2):106–112, 2007.

[9] Feng Lin. Control of networked discrete event systems: dealing with communication delays and losses. *SIAM Journal on Control and Optimization*, 52(2):1276–1298, 2014.

[10] Shaolong Shu and Feng Lin. Supervisor synthesis for networked discrete event systems with communication delays. *IEEE Transactions on Automatic Control*, 60(8):2183–2188, 2015.

[11] Shaolong Shu and Feng Lin. Decentralized control of networked discrete event systems with communication delays. *Automatica*, 50(8):2108–2112, 2014.

[12] Stavros Tripakis. Decentralized control of discrete-event systems with bounded or unbounded delay communication. *IEEE Transactions on Automatic Control*, 49(9):1489–1501, 2004.

[13] Seong-Jin Park and Kwang-Hyun Cho. Decentralized supervisory control of discrete event systems with communication delays based on conjunctive and permissive decision structures. *Automatica*, 43(4):738–743, 2007.

[14] Shaolong Shu and Feng Lin. Supervisor synthesis for networked discrete event systems with communication delays. *IEEE Transactions on Automatic Control*, 60(8):2183–2188, 2015.

[15] Rong Su, Jan H van Schuppen, and Jacobus E Rooda. Model abstraction of nondeterministic finite-state automata in supervisor synthesis. *IEEE Transactions on automatic control*, 55(11):2527–2541, 2010.

[16] Yuting Zhu, Liyong Lin, Simon Ware, and Rong Su. Supervisor synthesis for networked discrete event systems with communication delays and lossy channels. *IEEE Conference on Decision and Control*, 2019.

[17] WM Wonham, Kai Cai, and Karen Rudie. Supervisory control of discrete-event systems: A brief history–1980-2015. *IFAC-PapersOnLine*, 50(1):1791–1797, 2017.

[18] Lei Feng and W Murray Wonham. Tct: A computation tool for supervisory control synthesis. In *2006 8th International Workshop on Discrete Event Systems*, pages 388–389. IEEE, 2006.

[19] Knut Akesson, Martin Fabian, Hugo Flordal, and Arash Vahidi. Supremica—a tool for verification and synthesis of discrete event supervisors. In *11th mediterranean conference on control and automation*, 2003.