Appendix A. Proof of Propositions

To facilitate analysis, a generic production scenario is given with a limited length of steps, which is called production execution. A production execution E contains a finite number of parallel or sequential (or both) production function executions (i.e., δ) on multiple machines. *Proposition 1* shows that the design can correctly replicate the production process state to the simulation.

Proposition 1 (State Replication). Let S_p^t and $S'_p{}^t$ be the production state and the simulation state respectively. Given a production execution E on n entities with m external inputs, and the same initial state for a physical twin and an intelligent being, then $S_p^t = S'_p{}^t$ after executing E.

Proof: Let PT_i and IB_i be a physical twin and an intelligent being. As PT_i and IB_i have the same initial state, then $S_i^0 = S'_i{}^0$ and $u_i = u'_i$, which is implied by the input replication design in PT-IB communication. First, if PT_i is driven by an external input u_i^0 at time 0, then the state transition of IB_i have the same input u_i^0, which is implied by the input replication design in PT-IB communication. In this case, $S_i^t = S'_i{}^t$ can be derived, as PT_i and IB_i are constructed with the state transition function $\delta_i(S_i^0, u_i^0) = \delta_i(S'_i{}^0, u_i^0)$.

Second, let PT_j and IB_j be the input neighbors of PT_j and IB_j. Assume PT_j and IB_j can reach the same state after executing a production function, and output v_j and v'_j to PT_j and IB_j. Defined by E, the next step is to executing the function δ_i on PT_i and IB_i. Same as before, PT_i and IB_i will have the same input $u_i = v_j$. Also, in PT-IB communication, it is assumed that u_i can be reliably delivered to IB_i in the order of sending time. Then, all the inputs preceding u_i from PT_i should have been received by IB_i. According to the execution condition in IB state update, u_i will only be handled by IB_i after all preceding inputs have been handled in the order of sending time, which implies that both PT_i and IB_i have the same state before handling u_i: $S_i^t = S'_i{}^t$. Therefore, $S_i^{t+1} = S'_i{}^{t+1}$ can be derived from $\delta_i(S_i^t, u_i) = \delta_i(S'_i{}^t, u_i)$.

By induction, $S_i^t = S'_i{}^t$ for PT_i and IB_i at the end of E, which can be applied to any intelligent being by generalization. Moreover, as the state S_p^t and $S'_p{}^t$ are determined by the composition of all physical entities and intelligent beings, they can be defined by $S_p \in \boldsymbol{S_p} = \boldsymbol{S_1} \times \boldsymbol{S_2} \times \ldots \boldsymbol{S_k}$. and $S_p \in \boldsymbol{S'_p} = \boldsymbol{S'_1} \times \boldsymbol{S'_2} \times \ldots \boldsymbol{S'_k}$ respectively. Therefore, $S_p^t = (S_1^t, S_2^t, \ldots, S_k^t) = (S'_1{}^t, S'_2{}^t, \ldots, S'_k{}^t) = S'_p{}^t$ after executing E. ■

The rest analyses rely on a consistency of state transition between a simulation and a physical production. As a distributed system state transition is defined as the collection of distributed components, the state transitions of two replicated distributed systems are equal, if the two systems have the same sequence of state transition on replicated components and they finally reach the same state. *Lemma 2* shows that the design ensures that an intelligent being will transits to a new state only after it sees all the preceding updates of its neighbors.

Lemma 2 (Transition Consistency). Let S_p^t and $S'_p{}^t$ be the production state and the simulation state respectively. Given a production execution E on n entities with m external inputs, and the same initial state for a physical twin and an intelligent being, then $(S_p^0 \rightarrow_p S_p^t) = (S'_p{}^0 \rightarrow_p S'_p{}^t)$ after executing E, where $\rightarrow_p$ denotes the state transition process of a distributed system.

Proof: Let PT_i and IB_i be a physical twin and a intelligent being. First, if PT_i is driven by an external input u_i^0 at time 0, then by default, both PT_i and IB_i have the same sequence of state transition from an empty precedent.

Second, let PT_j, PT_{j+1}, …, PT_{j+p} and IB_j, IB_{j+1}, …, IB_{j+p} be the input neighbors of PT_i and IB_i respectively. PT_i will handle input $u_i = v_j$ after PT_j, PT_{j+1}, …, PT_{j+p} execute δ_j, δ_{j+1}, … δ_{j+p}. On IB_i, it will receive u_i^t from PT_i, and then execute $\delta_i(S'_i{}^t, u_i)$ only if $V(IB_i(u_i)) = V(IB_i)$, which will be satisfied only after IB_i has seen the updates on IB_j, IB_{j+1}, …, IB_{j+p}. According to the IB state update rule, IB_j, IB_{j+1}, …, IB_{j+p} will send output v'_j, v'_{j+1}, …, v'_{j+p} to IB_i only after they execute δ_j, δ_{j+1}, … δ_{j+p}. Thus, it can be inferred that IB_i will execute $\delta_i(S'_i{}^t, u_i^t)$ only after IB_j, IB_{j+1}, …, IB_{j+p} execute δ_j, δ_{j+1}, … δ_{j+p}, which follows the same sequence of state transition as PT_{j+1}, …, PT_{j+p}, and PT_i.

By induction, the production system and the simulation system have the same sequence of state transitions between physical entities and intelligent beings. Moreover, from *Proposition 1*, they both can reach the same state S_p^t and $S'_p{}^t$ after executing E. Therefore, $(S_p^0 \rightarrow_p S_p^t) = (S'_p{}^0 \rightarrow_p S'_p{}^t)$. ■

Proposition 1 and *Lemma 2* can then be applied to prove the satisfaction of data acquisition consistency, control consistency, and configuration change consistency in the design, which are shown in *Proposition* 3-6 respectively. Particularly, the process change consistency is separated into entity entrance consistency (*Proposition 5*) and entity removal consistency (*Proposition 6*) for covering the two typical cases of interest update. The scenario of only input-output relation change has already been embedded in these two cases. Thus, it is not explicitly shown.

Proposition 3 (Production monitoring Consistency). During the execution of E, let S''_1, S''_2, … S''_n, be the state of virtual objects in a client C_0. Also let $S''_p = (S''_1, S''_2, \dots S''_n)$ be the received entity states. Then, $(S_p^0 \rightarrow_p S_p^t) = (S''_p{}^0 \rightarrow_p S''_p{}^t)$ during executing E and $S_p^t = S''_p{}^t$ after E.

Proof: As c_0 applies $S'_i{}^t$ (or $\Delta S'_i{}^t$) to update virtual objects, the state of each object will be synchronized to the corresponding intelligent being. Thus, $S''_p =_p S'_p$. For a virtual object O_i and an input neighbor O_j, when $\Delta S'_i{}^t$ has been receive by c_0, $V(O_j)[j] \geq V(IB_i(m))[j]$ holds only when $\Delta S'_j{}^t$ has already been applied by c_0. This can be generalized to all the input neighbors of O_i, and thus, $(S'_p{}^0 \rightarrow_p S'_p{}^t) = (S''_p{}^0 \rightarrow_p S''_p{}^t)$. Moreover, *Proposition 2* has shown that $(S_p^0 \rightarrow_p S_p^t) = (S'_p{}^0 \rightarrow_p S'_p{}^t)$. By combining the above relations, it can be inferred that $(S_p^0 \rightarrow_p S_p^t) = (S''_p{}^0 \rightarrow_p S''_p{}^t)$ after executing E. ■

Proposition 4 (Control Consistency). During the execution of E, let z_i be a control instruction sent to a intelligent being IB_i. Then $S_p^t =_p S'^t_p$ after executing E.

Proof: When IB_i receives z_i, it does not execute z_i straightforward. Instead, it will forward z_i to PT_i. Thus, z_i can be treated as an external input to PT_i and *Proposition 1* can be applied by changing m external inputs to $m + 1$ external inputs, while the conclusion still holds. That is, $S_p^t =_p S'^t_p$ after executing E. ■

Proposition 5 (Entity Entrance Consistency). During the execution of E, add PT_{n+1} and IB_{n+1} to the production with the same initial state. Then, $(S_p^0 \rightarrow_p S_p^t) = (S'^0_p \rightarrow_p S'^t_p)$ during executing E and $S_p^t = S''^t_p$ after E.

Proof: Let t_1 be the time that PT_{n+1} and IB_{n+1} are added into the production. *Proposition 2* has shown that $(S_p^0 \rightarrow_p S_p^{t_1}) = (S'^0_p \rightarrow_p S'^{t_1}_p)$. When PT_{n+1} and IB_{n+1} are added into SG, the interest update procedure is triggered. An input neighbor of IB_{n+1}, denoted by IB_i, As IB_{n+1} has synchronized $V(IB_{n+1})$ to the input neighbors, their version vectors are consistent. Moreover, as $\{v'_{j\rightarrow i}\}$ have been synchronized to IB_{n+1} from any input neighbor IB_j, the inputs of PT_{n+1} and IB_{n+1} are synchronized (Note that if a v_j has not been sent to PT_{n+1}, then $u'_i = v'_j$ will not be handled by IB_{n+1}). Therefore, SG' is homogeneous to SG with a larger number of entities. So, *Proposition 2* can also be applied here: $(S_p^{t_1} \rightarrow_p S_p^t) = (S'^{t_1}_p \rightarrow_p S'^t_p)$. By combining the above results, it can be inferred that $(S_p^0 \rightarrow_p S_p^t) = (S'^0_p \rightarrow_p S'^t_p)$ after executing E. ■

Proposition 6 (Entity Removal Consistency). During the execution of E, remove PT_i and IB_i from SG. Then, $(S_p^0 \rightarrow_p S_p^t) = (S''^0_p \rightarrow_p S''^t_p)$ during executing E and $S_p^t = S''^t_p$ after E.

The proof of *Corollary 4* is similar to the one of *Corollary 3*. So, it will not be repeated here.