RESILIENCE BOUNDS OF NETWORK CLOCK SYNCHRONIZATION WITH FAULT CORRECTION

LINSHAN JIANG, RUI TAN, ARVIND EASWARAN

Abstract. Naturally occurring disturbances and malicious attacks can lead to faults in synchronizing the clocks of two network nodes. In this paper, we investigate the fundamental resilience bounds of network clock synchronization for a system of N nodes against the peer-to-peer synchronization faults. Our analysis is based on practical synchronization algorithms with time complexity down to $O(N^3)$ that attempt to correct the faults by checking the consistency among the following three types of data: 1) the estimated faults, 2) the estimated clock offsets among the nodes, and 3) the measured clock offsets from the potentially faulty peer-to-peer synchronization sessions. Our analysis gives the following three major results. First, the maximum number of faults that can be corrected by the algorithms has a tight bound of $\lfloor N/2 \rfloor - 1$ when every node pair performs a synchronization session. Second, by converting the fault resilience problem to a graph-theoretic edge connectivity problem and applying Menger's theorem, we develop an algorithm to compute the tight bound when not every node pair performs a synchronization session. Third, the number of synchronization sessions to achieve the capability of correcting any K faults has a lower bound of $\lceil N(2K+1)/2 \rceil$; we also develop an algorithm to schedule the synchronization sessions to approach the lower bound. The above results provide basic understanding and useful guidelines to the design of resilient clock synchronization systems. For instance, our results suggest that, the 4-node network achieves the highest degree of resilience that is defined as the ratio of the maximum number of correctable faults to the number of synchronization sessions. Therefore, by organizing a large-scale clock synchronization system into a hierarchy of multiple tiers with each consisting of 4-node synchronization groups, we can achieve satisfactory and understood resilience against faults with reduced synchronization sessions.

1. Introduction

For network systems such as wireless sensor networks (WSNs) and coordinated robots, accurate clock synchronization among the distributed nodes is important. Correct timestamps make sense the sensing data; synchronized clocks enable punctual coordinated operations among multiple robotic arms that collaborate on a

This manuscript was accepted by *ACM Transactions on Sensor Networks* on June 25th 2020. A preliminary version of this work appeared in *The 24th IEEE International Conference on Parallel and Distributed Systems (ICPADS)* held in Singapore, December 2018.

The authors were with School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Ave, Singapore 639798. E-mail: {linshan001, tanrui, arvinde}@ntu.edu.sg.

This research was supported in part by an Nanyang Technological University (NTU) Start-up Grant and in part by the Delta-NTU Corporate Laboratory for Cyber-Physical Systems with funding support from Delta Electronics Inc. and the National Research Foundation (NRF), Singapore under the Corp Lab @ University Scheme. The authors acknowledge Mr. Jothi Prasanna Shanmuga Sundaram for discussions during the early stage of this work and Dr. Yi Li for useful feedback on this work.

production line in a manufacturing system. In contrast, desynchronized clocks will undermine system performance and even lead to physical damages (e.g., clashing of robotic arms) and system disruptions in time-critical systems. However, as the distributed nodes are often deployed in complex physical environments with various naturally occurring disturbances and even malicious attacks, maintaining resilient system-wide clock synchronization can be challenging.

Network Time Protocol (NTP) [19] is the foremost means of clock synchronization that is widely known and adopted. The nodes in a system running NTP are organized into a layered hierarchy, in which a stratum-n node acts as a slave in synchronizing itself with a stratum-$(n-1)$ node, and as a master when providing its clock values to stratum-$(n+1)$ and other stratum-n nodes. Thus, in an NTP system, the global time that is directly accessed by the stratum-1 nodes is disseminated to all the nodes stratum by stratum. The intact dissemination highly depends on the successful peer-to-peer (p2p) clock synchronization sessions. A p2p synchronization session can be realized by the direct communication between the two nodes or by the multi-hop communications via several relay nodes. The p2p synchronization session estimates the offset between the clocks of two synchronizing nodes (referred to as *clock offset* in this paper) using a round-trip timing approach. Specifically, with the one-way packet delivery time estimated as half of the measured round-trip time, the clock offset can be computed based on the two nodes' respective clock values when the packet leaves one node and arrives at the other. With the estimated clock offset, a slave node can reset its clock value or calibrate its clock advance speed to achieve synchronization with the master node. This round-trip timing approach is also the basis of the Precision Time Protocol (PTP) [11] that is adopted in industrial Ethernets for higher synchronization accuracy. To reduce communication overhead, the clock synchronization approaches developed for WSNs (e.g., RBS [6], TPSN [7], and FTSP [18]) often assume near-zero wireless signal propagation times by accessing radio chips' hardware interrupts and therefore can synchronize two nodes with a one-way communication.

However, the integrity of the p2p synchronization sessions can be compromised. A basis of the round-trip timing approach is that the communication link between the two synchronizing nodes is symmetric, which may not hold faithfully in practice, however. For instance, in a packet-switched network, the uplink to and downlink from the master node may take different routes with distinct end-to-end delays. Over a wireless link, the media access control (MAC) may introduce uncertain delays of up to hundreds of milliseconds in transmitting a packet [18]. However, not all systems can perform the packet timestamping in the MAC layer to exclude this uncertainty from the clock offset estimation. Moreover, as discussed in RFC 7384 [20], the attackers may introduce controlled delays to the deliveries of the packets and breach the symmetric link assumption. As shown in [25], this *packet delay attack* can be implemented in a wired network via a compromised network router. Moreover, this threat cannot be solved by conventional security measures such as crytographic authentication and encryption [20, 21, 29]. The violations of the symmetric link assumption caused by any of the above reasons will lead to errors in estimating the clock offset and faulty p2p clock synchronization sessions. The one-way synchronization approaches adopted for WSNs are also susceptible to the delay. In [8], the packet delay attack against a low-power wide-area network is implemented by a combination of malicious packet collision and delayed replay.

In this paper, we investigate the fundamental resilience bounds of *network clock synchronization* (NCS) for a system of N nodes against the p2p synchronization faults. Specifically, we study the problem of deriving the maximum number of p2p synchronization faults that the network can correct to maintain the clock synchronization among all the nodes, as well as the dual problem of deriving the minimum number of p2p synchronization sessions to ensure the network's ability to correct a specified number of p2p synchronization faults. Our analysis is based on an NCS algorithm that attempts to correct the p2p synchronization faults. The algorithm is as follows. Consider an *NCS graph* $G = (V, E)$, where V and E respectively denote the set of the nodes in the network and the set of the p2p synchronization sessions each performed between two nodes. We use $|E|$ to denote the cardinality of the set E. The kth step of the algorithm assumes that k out of totally $|E|$ p2p synchronization sessions are faulty, exhaustively tests all possible $\binom{|E|}{k}$ combinations of the assumed faulty p2p synchronization sessions among the $|E|$ sessions, and yields a solution once the estimated clock offsets and the estimated p2p synchronization faults agree with all the p2p clock offset measurements. Starting from $k = 0$, the algorithm increases k by one in each step and terminates once a solution is found. This algorithm is practical in that it does not require any run-time knowledge about the p2p synchronization faults, including the actual number of the faults and which synchronization sessions out of totally $|E|$ sessions are actually faulty. The implementation of NCS needs a central node that can execute the compute-intensive NCS algorithm and can communicate reliably with all the nodes in the network to collect the results of p2p synchronization sessions. In a sensor network, the gateway with sufficient compute resources can serve as the NCS central node.

However, analyzing the resilience bounds of the NCS algorithm is challenging. First, the approach of analyzing all possible cases of the actual faults and assumed faults incurs prohibitive overhead. Specifically, in the kth step of the algorithm, we need to analyze a total of $\binom{|E|}{k}$ possible distributions of the assumed faulty p2p synchronization sessions among the $|E|$ synchronization sessions. Thus, this approach becomes infeasible when $|E|$ is large. Second, for larger networks with more nodes, it becomes difficult to enumerate all possible isomorphic NCS graphs with a certain number of edges. Thus, enumerating all possible cases to analyze the resilience bounds is not a promising approach.

In this paper, to analyze the resilience bounds, we introduce fault-free NCS subgraphs and convert the NCS resilience problem to a graph-theoretic problem. Assisted with the existing results in graph theory, we obtain the following main analytic results for the resilient NCS problem:

- For a complete NCS graph in which every node pair performs a p2p synchronization session, the tight bound of the maximum number of p2p synchronization faults that can be corrected by the NCS algorithm is $\lfloor \frac{N}{2} \rfloor - 1$. In other words, the NCS algorithm can synchronize all nodes when the number of p2p synchronization faults is no greater than the tight bound; otherwise, some nodes in the network will be desynchronized due to the faults.
- For any NCS graph that may be incomplete, we convert the fault resilience problem to a graph-theoretic edge connectivity problem. From our analysis based on Menger's theorem [1], we develop an algorithm to compute the tight bound of the maximum number of p2p synchronization faults

that can be corrected by the NCS algorithm. Moreover, we develop a new NCS algorithm with a time complexity of $O(N^3)$ that achieves the same fault correction capability as the original NCS algorithm that has a time complexity of $O\left(\sqrt{2}^N\right)$.

- We study the *minimum* NCS graph that uses the least edges to provide resilience against a specified number of p2p synchronization faults (denoted by K). We develop an algorithm to compute the minimum NCS graphs. We prove that $\left\lceil \frac{N\cdot(2K+1)}{2} \right\rceil$ is a lower bound of the number of edges of any NCS graph that is resilient against K p2p synchronization faults. Numeric results suggest that the lower bound is tight. The lower bound can be used to understand the order of magnitude of the number of edges in any minimum NCS graph.

The analytic results in this paper provide important understanding and useful guidelines to the design of clock synchronization systems that are resilient to p2p clock synchronization faults. They are useful to time-critical systems such as industrial wireline and wireless Ethernets. Particularly, in addition to the analysis, this paper discusses the design of a clock synchronization architecture that strikes a good trade-off between the p2p synchronization communication overhead and resilience to p2p faults. Specifically, we use the *degree of resilience* (DoR) as the resilience metric, which is defined as the ratio between the number of faults that can be corrected and the number of edges in an NCS graph. Based on our analytic results, we show that a 4-node network with complete NCS graph achieves the highest DoR of 1/6. From this observation, we propose a tiered clock synchronization architecture for large-scale networks, in which the nodes in a network are grouped into 4-node synchronization groups that are organized into multiple tiers. This architecture provides reduced but well understood resilience against p2p clock synchronization faults (i.e., every 4-node synchronization group can have one fault), compared with the original large-scale network with a minimum NCS graph. The number of p2p synchronization sessions in the proposed architecture is much reduced, compared with that of the minimum NCS graph.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 states the problem. Section 4 analyzes the resilience of several small-scale networks to illustrate the key challenges in the resilience analysis. Section 5 derives the tight resilience bound of complete NCS graphs. Section 6 develops the algorithms to compute the resilience bound of any graph. Section 7 develops a fast NCS algorithm with a cubic time complexity that achieves the same fault correction capability as the original NCS algorithm. Section 8 studies minimum NCS graphs. Section 9 discusses the implications of our results and a tiered clock synchronization architecture for fault resilience. Section 10 concludes the paper.

2. Related Work

In this section, we review the implementations of clock synchronization for different network systems and the existing studies on the fault tolerance of NCS.

2.1. **Implementations of Clock Synchronization.**

Highly stable time sources are often ill-suited for network systems. Despite an initial study of using chip-scale atomic clock (CSAC) on WSN platforms [5], CSAC is still too expensive ($1,500

per unit [5]) for wide adoption. Thus, how to synchronize the nodes in different kinds of network systems has received extensive research.

On the Internet, NTP [19] has been widely used to synchronize computer hosts. It is universal because it imposes few requirements, i.e., it only requires the host to timestamp the transmission and reception of the synchronization packets in the operating system (OS). Compared with NTP, PTP [11] additionally requires the network interfaces of the synchronizing hosts and all the switches on the network paths among the hosts to have hardware-level timestamping capability. As such, PTP can exclude the uncertain OS and packet switching delays from the packet delivery time measurements, largely improving the accuracy in estimating the clock offsets. However, malfunctioned hardware-level timestamping will lead to p2p synchronization faults. The reference implementations of NTP and PTP have various mechanisms to improve their robustness against p2p synchronization faults. For instance, in NTP, when the round-trip time exceeds one second, the current p2p synchronization session is considered faulty and will not be used to calibrate the host's clock. Moreover, a slave node will average the clock offset estimation results obtained with multiple master nodes to guide its clock calibration. Despite these heuristics in the protocol implementation for fault resilience, an analytic understanding regarding the network's resilience against the p2p synchronization faults is still lacking.

Over the past decade, WSN clock synchronization has been widely studied. There are accurate global time broadcasts from the Global Positioning System (GPS) and timekeeping radio stations (e.g., WWVB in U.S.). However, GPS and radio receivers have various limitations such as high power consumption, poor signal reception in indoor environments (e.g., 47% good time for WWVB [2]), and susceptibility to wireless spoofing attacks [22]. Thus, GPS and radio receivers are often employed on a limited number of time masters to provide global time to the slave nodes via some clock synchronization protocol. The resilience of the clock synchronization between the master and the slaves is the focus of this paper.

Early studies have designed clock synchronization protocols based on message passing, such as RBS [6], TPSN [7], and FTSP [18]. Recent studies exploit various external periodic signals for clock synchronization [30, 25, 34], time fingerprinting [17, 9, 30, 15, 18], and clock calibration [26, 16, 10, 14]. Time fingerprinting approaches focus on studying the global time information embedded in the sensing data such as microseisms [17], sunlight [9], powerline voltage [30], and electromagnetic radiation [15]. They can be a basis for clock synchronization. For instance, the work [30] achieves microseconds clock synchronization accuracy by using the time fingerprints found in the electric voltages of a building's power network. Different from clock synchronization that ensures the clocks to have the same value, *clock calibration* ensures different clocks to advance at the same speed. The approaches presented in [26, 16, 10, 14] exploit powerline electromagnetic radiation, fluorescent lamp flickering, Wi-Fi beacons, and FM Radio Data System broadcasts to calibrate the clocks of WSN nodes. Clock calibration may not need any internode communications, whereas clock synchronization must need communicating one or more timestamps between two synchronizing nodes. However, all the above studies [6, 7, 18, 30, 25, 34, 17, 9, 15, 26, 16, 10, 14] focus on devising clock synchronization/calibration approaches. They fall short of analyzing the resilience of the system against potential clock synchronization faults.

2.2. **Fault Tolerance of NCS.** The fault tolerance of NCS against Byzantine clock faults has been studied [4, 13]. A Byzantine faulty clock gives an arbitrary clock value whenever being read. It has been proved that, to guarantee the synchronization of non-faulty clocks in the presence of m faulty clocks, a total of at least $(3m + 1)$ clocks are needed. Different from the Byzantine faulty clock model, we consider faulty p2p synchronization sessions between the clocks. The conversion of our problem to the Byzantine clock synchronization problem by considering either node involving a faulty p2p synchronization session as a faulty clock is *invalid*, because this faulty clock after the conversion is not a Byzantine faulty clock, unless all p2p synchronization sessions involving this clock are faulty. As our problem does not have this assumption, the the analysis in [4, 13] and the resulted fault tolerance bound are not applicable to our problem. Moreover, different from the fault-tolerant systems in [4, 13] that do not try to correct the faults, our resilient NCS system tries to correct the p2p synchronization faults.

Our prior work [28] presented the formulation of the resilience of NCS against p2p synchronization faults. It developed an algorithm to compute a lower bound and derived a closed-form upper bound of the maximum number of faults that can be corrected for any complete NCS graph. In this paper, we derive the closed-form tight bound of resilience for any complete NCS graph, which represents a substantial improvement to the results in [28]. Moreover, this paper studies three new problems: (1) the resilience bounds of NCS graphs that can be incomplete (Section 6) , (2) fast NCS algorithm with polynomial complexity to achieve the same fault correction capability as the original NCS algorithm (Section 7), and (3) the minimum NCS graphs providing a specified level of resilience (Section 8).

3. Problem Statement

This section presents the system model (Section 3.1) and states the research problem (Section 3.2). In Section 3.3, we discuss several abstractions in our system model and their relations with real NCS systems.

3.1. **System Model and NCS.** To improve the robustness of clock synchronization against p2p synchronization faults, this section proposes an approach to cross-check the p2p synchronization results among multiple nodes and correct the faults if present.

Let V denote the set of N nodes in a network, i.e., $V = \{n_0, n_1, \ldots, n_{N-1}\}$, where n_i represents the ith node. Let δ_{ij} denote the clock offset between the nodes n_i and n_j, which is unknown and to be estimated. Specifically, $\delta_{ij} = c_i(t) - c_j(t)$, where $c_i(t)$ and $c_j(t)$ are the clock values of n_i and n_j at any given Newtonian time instant t, respectively. We assume that δ_{ij} is time-invariant. In Section 3.3, we will discuss the validity of this assumption in real systems. By designating n_0 as the *reference node*, we have the relationship $\delta_{ij} = \delta_{i0} - \delta_{j0}$, which will be used for analysis in the rest of this paper.

Denote by $n_i \leftrightarrow n_j$ the p2p synchronization session between n_i and n_j. Denote by $\widetilde{\delta}_{ij}$ the measured clock offset via $n_i \leftrightarrow n_j$. If the synchronization session $n_i \leftrightarrow n_j$ is successful (i.e., non-faulty), $\widetilde{\delta}_{ij} = \delta_{ij}$; if the synchronization session is faulty, $\widetilde{\delta}_{ij} = \delta_{ij} + e_{ij}$, where e_{ij} is the p2p synchronization fault which is a non-zero and finite real number. Let E denote the set of all p2p synchronization sessions performed in a *synchronization round*. In our NCS approach, the result of at most

Algorithm 1 NCS algorithm with fault correction.

Given: $\{\widetilde{\delta}_{ij}|\forall n_i \leftrightarrow n_j \in E\}$
Output: $\{\hat{\delta}_{j0}|\forall j \in [1, N-1]\}$ and $\{\hat{e}_{ij}|\forall n_i \leftrightarrow n_j \in E\}$
1: $k \leftarrow 0$
2: **while** $k \leq |E|$ **do**
3: **for** each distribution of the k estimated p2p synchronization faults among the $|E|$ p2p synchronization sessions **do**
4: **if** the corresponding variant of Eq. (1) with the k estimated p2p synchronization faults has a solution **then**
5: return $\{\hat{\delta}_{j0}|\forall j \in [1, N-1]\}$ and $\{\hat{e}_{ij}|\forall n_i \leftrightarrow n_j \in E\}$
6: **end if**
7: **end for**
8: $k \leftarrow k + 1$
9: **end while**

one p2p synchronization session performed between any pair of nodes is used in one synchronization round. Note that the analysis of this paper is agnostic to the technique used for each synchronization session. For instance, a round-trip timing process can be used to obtain $\widetilde{\delta}_{ij}$ between n_i and n_j. Moreover, since at most one synchronization session between n_i and n_j is used in one synchronization round, the edge $n_i \leftrightarrow n_j$ can be modeled undirected.

For a synchronization round, the undirected graph $G = (V, E)$ is called the NCS graph. In a complete NCS graph, every node pair performs a p2p synchronization session, resulting in $|E| = \binom{N}{2} = \frac{N(N-1)}{2}$. For any NCS graph $G = (V, E)$ that may be incomplete, the NCS is performed as follows. All the clock offset measurements are transmitted to a central node, which runs the NCS algorithm that is shown in Algorithm 1. The central node can undertake compute-intensive NCS algorithm and can communicate reliably with all the nodes. It can be an external entity (e.g., a cloud service) or any connected node in the network. For the latter case, the central node may not be the reference node; various strategies can be used to select the central node. For example, in a battery-powered network that concerns about the nodes' energy consumption, a node with the most remaining battery energy can perform the role of central node to receive the clock offset measurements and run the NCS algorithm. We assume that the p2p clock synchronization sessions are separate from the transmissions of the clock offset measurements to the central node. In certain cases, the transmission of the measured clock offset can be avoided. For instance, if the round-trip timing approach is used and the central node initiates the round-trip timing, the central node obtains $\widetilde{\delta}_{ij}$ on the completion of the round-trip timing and requires no separate transmission of $\widetilde{\delta}_{ij}$. Note that, when every node performs a p2p synchronization session with the reference node, the NCS graph G will have a star topology centered at the reference node. Algorithm 1 and all analytic results in this paper are also applicable to this star NCS graph.

We now explain Algorithm 1. Denote by $\hat{\delta}_{ij}$ and $\hat{e}_{ij}$ the estimates for δ_{ij} and e_{ij}. A general equation system assuming that all synchronization sessions are faulty is

$$\begin{cases} \hat{\delta}_{j0} + \hat{e}_{j0} = \widetilde{\delta}_{j0}, & \forall n_j \leftrightarrow n_0 \in E, j \neq 0; \\ \hat{\delta}_{i0} - \hat{\delta}_{j0} + \hat{e}_{ij} = \widetilde{\delta}_{ij}, & \forall n_i \leftrightarrow n_j \in E, i \neq 0, j \neq 0. \end{cases} \tag{1}$$

The variables to be solved are the unknowns $\{\hat{\delta}_{j0}|\forall j \in [1, N-1]\}$ and $\{\hat{e}_{ij}|\forall n_i \leftrightarrow n_j \in E\}$, where $\hat{\delta}_{j0}$ is the estimated clock offset between n_j and the reference node n_0; $\hat{e}_{ij}$ is the estimated p2p clock synchronization fault between n_i and n_j if they have performed the p2p synchronization session in the current synchronization round. Note that regardless of E, we aim at solving the clock offset between every node and the reference node. If there are too few edges in E, Eq. (1) may have infinite number of solutions. Our resilience definition in Section 3.2 accounts for this situation.

Let k denote the assumed number of faults, which can be different from the actual number of faults. The scattering of the k assumed faults on the $|E|$ p2p synchronization sessions is called *distribution* of the assumed faults. As shown in Algorithm 1, the NCS algorithm starts by assuming there are no faults (i.e., $k \leftarrow 0$). In each iteration of the algorithm that increases k by one, the algorithm solves the variants of Eq. (1) that capture all $\binom{|E|}{k}$ possible distributions of the k assumed faulty p2p synchronization sessions among all the $|E|$ p2p synchronization sessions. Specifically, a variant of Eq. (1) is generated by keeping k estimated p2p synchronization faults (i.e., $\hat{e}_{j0}$ or $\hat{e}_{ij}$) in Eq. (1) and removing other estimated p2p synchronization faults. Once a solution is found, Algorithm 1 returns the estimates $\{\hat{\delta}_{j0}|\forall j \in [1, N-1]\}$ and $\{\hat{e}_{ij}|\forall n_i \leftrightarrow n_j \in E\}$. If $\hat{\delta}_{j0} = \delta_{j0}$, $\forall j \in [1, N-1]$, we say Algorithm 1 can correct the faults.

Algorithm 1 requires neither the actual number nor the actual distribution of the p2p synchronization faults. Whether it can correct the faults and how many faults it can correct will be the focus of this paper. Algorithm 1 is a centralized algorithm executed on the central node. The time complexity of the kth step of Algorithm 1 is $O\left(\binom{|E|}{k}\right)$. Thus, the time complexity upper bound of Algorithm 1 is $O\left(\sum_{k=0}^{|E|}\binom{|E|}{k}\right) = O\left(2^{|E|}\right)$. In Section 7.2, we further show that the time complexity lower bound of Algorithm 1 is $\Omega\left(\sqrt{2}^{N}\right)$ for complete NCS graphs. Therefore, Algorithm 1 has an exponential time complexity. In Section 7.2, based on a graph-theoretic analysis, we will develop a fast NCS algorithm with a time complexity of $O(N^3)$ that provides the same fault correction capability as Algorithm 1. With the fast algorithm, the centralized NCS is scalable to network size. The resilience of the centralized NCS provides important baseline understanding on the resilience of synchronizing a network of nodes.

3.2. **Problem Statement.** Let $\mathbb{Z}_{\geq 0}$ denote the set of non-negative integers.

Definition 1 (K-resilience)**.** Let $K \in \mathbb{Z}_{\geq 0}$ denote the number of faulty p2p synchronization sessions among a total of $|E|$ sessions in an NCS graph $G = (V, E)$. The network with the NCS graph G is K-resilient if Algorithm 1 can correct *any* K p2p synchronization faults. □

From Algorithm 1, we define the K-resilience condition that can be used to check whether a network with G is K-resilient.

Definition 2 (K-resilience condition)**.** A network with G is K-resilient if the following conditions are satisfied:

(1) $\forall k \in [0, K)$, the variant of Eq. (1) corresponding to *any* distribution of the K actual p2p synchronization faults and *any* distribution of the k estimated p2p synchronization faults has no solutions;

(2) When $k = K$, for *any* distribution of the K actual p2p synchronization faults and *any* distribution of the k estimated p2p synchronization faults,
   (a) if the distribution of the k estimated p2p synchronization faults is identical to the distribution of the actual faults, Eq. (1) has a unique solution;
   (b) otherwise, Eq. (1) has no solutions. □

Note that under the condition (2)-(a) of Definition 2, if Eq. (1) has a unique solution, the solution must give the correct estimates of the clock offsets and the p2p synchronization faults, since these correct estimates form a valid solution. Note that if the K-resilience condition in Definition 2 is satisfied, Algorithm 1 must be able to correct any K faults. However, when the condition (2)-(b) of Definition 2 is not satisfied, Algorithm 1 can still correct K faults with a specific distribution of the faults. This occurs when the first attempted distribution of the K estimated faults happens to be identical to the actual distribution of the K faults. However, in this case, the network is not K-resilient, because Definition 1 requires that Algorithm 1 can correct *any* K faults to claim K-resilience. Thus, Definition 2 gives a sufficient condition for Algorithm 1 to correct any K faults; it is a sufficient and necessary condition for K-resilience.

Let $\mathbb{G}$ denote the infinite set of all NCS graphs. We define the following resilience bounds:

Definition 3 (Lower bound of maximum resilience)**.** A function $f_l(G) : \mathbb{G} \mapsto \mathbb{Z}_{\geq 0}$ is a lower bound of maximum resilience for a network with an NCS graph G if the network is K-resilient for $K \leq f_l(G)$. □

Definition 4 (Upper bound of maximum resilience)**.** A function $f_u(G) : \mathbb{G} \mapsto \mathbb{Z}_{\geq 0}$ is an upper bound of maximum resilience for a network with an NCS graph G if the network is not K-resilient for $K > f_u(G)$. □

Definition 5 (Tight bound of maximum resilience)**.** A function $f_t(G) : \mathbb{G} \mapsto \mathbb{Z}_{\geq 0}$ is a tight bound of maximum resilience for a network with an NCS graph G if the network is K-resilient for $K \leq f_t(G)$ and not K-resilient for $K > f_t(G)$. □

This paper aims at investigating the above resilience bounds under various NCS graph (e.g., complete or not) and the dual problem of what NCS graph condition can ensure a certain resilience bound.

3.3. **Relations with Real Clock Synchronization Systems.** The system model described in Section 3.1 includes several abstractions to clearly formulate the K-resilience concept in Section 3.2 and allow us to focus on the essence of the problem. In this section, we discuss the potential deviations of the real systems from these abstractions and the impact of such deviations on our analysis in the reminder of this paper.

3.3.1. *Definition of fault.* In this paper, we focus on the faults that are caused by erroneous clock offset estimates. We do not consider other faults such as missing clock offset estimates. In Section 3.1, any deviation of the measured clock offset from its true value is regarded as a fault. Under this rigorous definition of fault, we can describe the NCS algorithm and define the K-resilience without any vagueness. In real systems, a p2p clock synchronization session may have some clock offset estimation error due to inevitable random noises. The system designer often has

good knowledge of these random noises (e.g., their sources and probabilistic distributions) and designs the clock synchronization mechanism to limit the resulted clock offset estimation errors to acceptable ranges. In practice, the clock offset estimation errors that are caused by unforeseen situations (e.g., hardware malfunction and packet delay attack [25, 20, 21, 29]) and beyond the acceptable ranges can be regarded as faults. Following this principle, in this section, we discuss how to extend our formulation to address 1) a class of *sensing-based* clock synchronization systems and 2) other systems under more general settings.

In the sensing-based clock synchronization systems [30, 25, 34, 15], the clock offset estimation errors follow a discrete pattern. Specifically, the error is given by $e_{ij} = \epsilon_{ij} + m_{ij} \cdot T$, where T is the period of the external signal being sensed, m_{ij} is an integer, and ϵ_{ij} is a random noise with magnitude much smaller than T. For instance, in the study [30] that exploits powerline voltages to synchronize nodes in a city, T is 20 milliseconds in a 50 Hz power grid and the absolute value of ϵ_{ij} is about 0.1 milliseconds (i.e., 0.5% of T). The discrete pattern is caused by abnormal noises of the used external signals and some integer nature of the clock synchronization algorithms to leverage on the periodicity of the external signals. For these systems, the p2p synchronization sessions with $m_{ij} \neq 0$ can be regarded as faulty sessions. Due to the random noises ϵ_{ij}, Eq. (1) generally has no exact solutions even when there are no faulty p2p synchronization sessions (i.e., $K = 0$) and the considered $k = 0$. Instead, a candidate solution to Eq. (1) can be obtained by minimizing the following overall residual:

$$\sum_{\substack{\forall n_j \leftrightarrow n_0 \in E \\ \forall j \neq 0}} \left| \hat{\delta}_{j0} + \hat{e}_{j0} - \widetilde{\delta}_{j0} \right|^2 + \sum_{\substack{\forall n_i \leftrightarrow n_j \in E \\ \forall i \neq 0, \forall j \neq 0}} \left| \hat{\delta}_{i0} - \hat{\delta}_{j0} + \hat{e}_{ij} - \widetilde{\delta}_{ij} \right|^2 . \tag{2}$$

The candidate solution can be substituted into each equation in Eq. (1) to check if the absolute residual exceeds some threshold set according to the distribution of the random noise ϵ_{ij}. For example, we can set one millisecond for the system in [30]. If every absolute residual does not exceed the threshold, we view the candidate solution as a valid solution to Eq. (1) in Line 4 of Algorithm 1. The integer programming formulation in Eq. (2) exploits the discrete pattern of the synchronization faults. It reduces the impact of random synchronization errors on the accuracy of determining whether Eq. (1) has a solution.

For systems with a more general error pattern of $e_{ij} = \epsilon_{ij} + x_{ij} \cdot F_{ij}$ where x_{ij} is 0 or 1 and F_{ij} is an arbitrary number beyond the range of ϵ_{ij}, the synchronization sessions with $x_{ij} = 1$ can be regarded as faulty sessions. The residual minimization and candidate solution checking approaches described above can be applied as well. In Section 4.3, we will present a set of simulation results that consider the general error pattern.

Despite the above variations to address acceptable clock offset estimation errors, our abstracted formulation in Section 3.1 and Section 3.2 capture the essence of the problem. The analysis based on the formulation will provide insightful understanding regarding the fault resilience of the NCS mechanism in Algorithm 1.

3.3.2. *Time-invariant clock offset.* In Section 3.1, we assume that the clock offset δ_{ij} is time-invariant. In practice, the clock offset δ_{ij} can be time-varying because the clocks of n_i and n_j may advance at different speeds. However, the change of δ_{ij} during a p2p synchronization session is often negligible compared with the

clock offset estimation errors of successful p2p synchronization sessions. In most clock synchronization systems, a p2p synchronization session takes a short time (e.g., tens of milliseconds in NTP, PTP, and sensing-based clock synchronization such as [25]). Typical crystal oscillators found in microcontrollers and personal computers have drift rates of 30 to 50 parts-per-million (ppm) [10]. Thus, the change of the clock offset during a synchronization session of 100 milliseconds is at most 5 microseconds only, whereas the clock offset estimation errors of successful synchronization sessions are at sub-millisecond [30, 25] or milliseconds levels [34]. Thus, the small variation of the clock offset during a synchronization session can be viewed as a nearly negligible part of the clock offset estimation error, where the latter is further much smaller than the synchronization faults as discussed in Section 3.3.1. Therefore, we can safely ignore the variation of clock offset in studying the resilience of NCS against synchronization faults.

4. K-Resilience Analysis for Example Networks

In this section, we present the vectorization of Eq. (1) to facilitate our analysis (Section 4.1) and then analyze the K-resilience for several small-scale networks with complete NCS graphs (Section 4.2). The analysis illustrates the challenges in the general analysis of the K-resilience for any network, but also provides guiding insights. Lastly, we provide a set of simulation results to show the impact of non-faulty synchronization errors on the NCS algorithm (Section 4.3).

4.1. Vectorization.

We vectorize the representation of Eq. (1) that is solved by Line 4 of Algorithm 1. Define $\hat{\boldsymbol{\delta}} \in \mathbb{R}^{N-1}$ composed of all clock offset estimates, i.e., $\hat{\boldsymbol{\delta}} = \left(\hat{\delta}_{10}, \hat{\delta}_{20}, \ldots, \hat{\delta}_{(N-1)0}\right)^{\intercal}$. Define $\hat{\mathbf{e}} \in \mathbb{R}^k$ composed of the k p2p synchronization fault estimates. Eq. (1) can be rewritten as $(\mathbf{A}_1\mathbf{A}_2)\begin{pmatrix} \hat{\boldsymbol{\delta}} \\ \hat{\mathbf{e}} \end{pmatrix} = \mathbf{b}$, where $\mathbf{A}_1 \in \mathbb{R}^{|E|\times(N-1)}$ and $\mathbf{A}_2 \in \mathbb{R}^{|E|\times k}$ are two matrices composed of -1, 0, and 1 containing coefficients corresponding to $\hat{\delta}_{\cdot 0}$ and $\hat{e}_{\cdot\cdot}$, respectively; the vector $\mathbf{b} \in \mathbb{R}^{|E|}$ consists of all the measured clock offsets. To simplify notation, we define $\mathbf{A} = (\mathbf{A}_1\mathbf{A}_2)$ and $\mathbf{x} = \begin{pmatrix} \hat{\boldsymbol{\delta}} \\ \hat{\mathbf{e}} \end{pmatrix}$. The $\mathbf{Ax} = \mathbf{b}$ is called *NCS equation system*. From the Rouché-Capelli theorem [27], the necessary and sufficient condition that $\mathbf{Ax} = \mathbf{b}$ has no solutions is $\text{rank}(\mathbf{A}|\mathbf{b}) \neq \text{rank}(\mathbf{A})$, where $\mathbf{A}|\mathbf{b}$ is the augmented matrix.

4.2. K-Resilience of Small-Scale Networks.

This section presents the analysis on the K-resilience of several small-scale networks with complete NCS graphs.

Proposition 1. *A 3-node network with a complete NCS graph is not 1-resilient.*

Proof. Consider a case where the p2p synchronization session $n_1 \leftrightarrow n_2$ is faulty. When $k = 0$ in Algorithm 1, the vectorized equation system in Eq. (1) is

$$\underset{\mathbf{A}}{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 1 \end{pmatrix}} \quad \underset{\mathbf{x}}{\begin{pmatrix} \hat{\delta}_{10} \\ \hat{\delta}_{20} \end{pmatrix}} \quad = \quad \underset{\mathbf{b}}{\begin{pmatrix} \delta_{10} \\ \delta_{20} \\ \delta_{20} - \delta_{10} + e_{21} \end{pmatrix}}.$$

Note that $\mathbf{A}_2$ and $\hat{\mathbf{e}}$ are empty. With $e_{21} \neq 0$, Gaussian elimination shows that $\text{rank}(\mathbf{A}|\mathbf{b}) \neq \text{rank}(\mathbf{A})$. Thus, the equation system has no solutions and Algorithm 1

will move on to the case of $k = 1$. The algorithm will attempt to test all the $\binom{|E|}{k} = \binom{3}{1} = 3$ possible cases of a single faulty p2p synchronization session. For instance, when the algorithm assumes that $n_0 \leftrightarrow n_1$ is faulty, the NCS equation system is

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \hat{\delta}_{10} \\ \hat{\delta}_{20} \\ \hat{e}_{10} \end{pmatrix} = \begin{pmatrix} \delta_{10} \\ \delta_{20} \\ \delta_{20} - \delta_{10} + e_{21} \end{pmatrix}.$$

With $e_{21} \neq 0$, we have $\text{rank}(\mathbf{A}|\mathbf{b}) = \text{rank}(\mathbf{A})$ and $\mathbf{A}$ has full column rank. Thus, the NCS equation system has a unique solution. Therefore, the condition (2)-(b) of Definition 2 is not satisfied and the network is not 1-resilient. The unique solution is $\{\hat{\delta}_{10} = \delta_{10} - e_{21}, \hat{\delta}_{20} = \delta_{20}, \hat{e}_{10} = e_{21}\}$, which gives wrong clock offset estimates. □

Proposition 2. *A 4-node network with a complete NCS graph is 1-resilient.*

We provide a sketch of the proof as follows instead of a complete proof for presentation conciseness. In fact, this proposition is a corollary of Theorem 1 with a complete proof in Section 5.3. Thus, the omission of the complete proof here does not cause loss of rigor. Consider a case where the p2p synchronization session $n_0 \leftrightarrow n_2$ is faulty. When $k = 0$ in Algorithm 1, similar to Proposition 1, the NCS equation system has no solutions and Algorithm 1 will move on to the case of $k = 1$. The algorithm will test all the $\binom{|E|}{k} = \binom{6}{1} = 6$ possible cases of a single faulty p2p synchronization session. For instance, when the algorithm assumes $n_0 \leftrightarrow n_1$ is faulty, the NCS equation system is

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \hat{\delta}_{10} \\ \hat{\delta}_{20} \\ \hat{\delta}_{30} \\ \hat{e}_{10} \end{pmatrix} = \begin{pmatrix} \delta_{10} \\ \delta_{20} + e_{20} \\ \delta_{30} \\ \delta_{20} - \delta_{10} \\ \delta_{30} - \delta_{20} \\ \delta_{30} - \delta_{10} \end{pmatrix}. \tag{3}$$

As $\text{rank}(\mathbf{A}|\mathbf{b}) \neq \text{rank}(\mathbf{A})$, the NCS equation system has no solutions. An exhaustive check shows that, only when the algorithm assumes the synchronization session between n_0 and n_2 is faulty, the NCS equation system has a unique solution (i.e., $\text{rank}(\mathbf{A}|\mathbf{b}) = \text{rank}(\mathbf{A})$ and $\mathbf{A}$ has full column rank). Thus, the algorithm can correct the fault. In fact, it can be verified that, for the complete 4-node NCS graph, no matter which p2p synchronization session is faulty, the algorithm can correct the fault. Therefore, the 4-node system is 1-resilient.

Proposition 3. *A 4-node network with a complete NCS graph is not 2-resilient.*

Proof. Consider the 4-node network with two faulty p2p synchronization sessions: $n_0 \leftrightarrow n_1$ and $n_0 \leftrightarrow n_2$. When $k = 0$, the equation system has no solutions. When $k = 1$, consider a case where $n_0 \leftrightarrow n_3$ is assumed to be faulty by the algorithm. The NCS equation system is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \hat{\delta}_{10} \\ \hat{\delta}_{20} \\ \hat{\delta}_{30} \\ \hat{e}_{30} \end{pmatrix} = \begin{pmatrix} \delta_{10} + e_{10} \\ \delta_{20} + e_{20} \\ \delta_{30} \\ \delta_{20} - \delta_{10} \\ \delta_{30} - \delta_{10} \\ \delta_{30} - \delta_{20} \end{pmatrix}. \tag{4}$$

If $e_{10} \neq e_{20}$, $\text{rank}(\mathbf{A}|\mathbf{b}) \neq \text{rank}(\mathbf{A})$ and the equation system has no solutions. However, if $e_{10} = e_{20}$, $\text{rank}(\mathbf{A}|\mathbf{b}) = \text{rank}(\mathbf{A})$ and $\mathbf{A}$ has full column rank; the equation system has a unique solution of $\{\hat{\delta}_{10} = \delta_{10} + e_{10}, \hat{\delta}_{20} = \delta_{20} + e_{10}, \hat{\delta}_{30} = \delta_{30} + e_{10}, \hat{e}_{30} = -e_{10}\}$, which gives wrong clock offset estimates. Although this counterexample against the 4-node network's 2-resilience is obtained under a certain condition of $e_{10} = e_{20}$, we can conclude that the 4-node network is not 2-resilient. □

To gain more insights, we also analyze a case of $k = 2$ with $n_0 \leftrightarrow n_1$ and $n_0 \leftrightarrow n_3$ assumed to be faulty by the algorithm. The NCS equation system is

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \hat{\delta}_{10} \\ \hat{\delta}_{20} \\ \hat{\delta}_{30} \\ \hat{e}_{10} \\ \hat{e}_{30} \end{pmatrix} = \begin{pmatrix} \delta_{10} + e_{10} \\ \delta_{20} + e_{20} \\ \delta_{30} \\ \delta_{20} - \delta_{10} \\ \delta_{30} - \delta_{10} \\ \delta_{30} - \delta_{20} \end{pmatrix}. \tag{5}$$

As $\text{rank}(\mathbf{A}|\mathbf{b}) = \text{rank}(\mathbf{A})$ and $\mathbf{A}$ has full column rank, the equation system has a unique solution, which violates the 2-resilience condition. In fact, the equation system has a unique solution that gives wrong clock offset estimates and does not require any relationship between e_{10} and e_{20}. This solution is $\{\hat{\delta}_{10} = \delta_{10} + e_{20}, \hat{\delta}_{20} = \delta_{20} + e_{20}, \hat{\delta}_{30} = \delta_{30} + e_{20}, \hat{e}_{10} = e_{10} - e_{20}, \hat{e}_{30} = -e_{20}\}$.

Proposition 4. *A 5-node network with a complete NCS graph is 1-resilient.*

We provide a sketch of the proof as follows instead of a complete proof due to space limit. This proposition is in fact a corollary of Theorem 1 with a complete proof. Thus, the omission here does not cause loss of rigor. Consider a 5-node network with one p2p synchronization fault. The resilience is independent from how we name the nodes. We name the two nodes involved in the faulty synchronization session as n_0 and n_1. An exhaustive check over all the $\binom{|E|}{k} = \binom{10}{1} = 10$ possible cases for a single assumed faulty synchronization session shows that the 1-resilience condition is satisfied. Thus, the 5-node network is 1-resilient.

Proposition 5. *A 5-node network with a complete NCS graph is not 2-resilient.*

Proof. We consider a 5-node network, in which (i) the p2p synchronization sessions $n_0 \leftrightarrow n_1$ and $n_1 \leftrightarrow n_4$ are faulty and (ii) the p2p synchronization sessions $n_1 \leftrightarrow n_2$ and $n_1 \leftrightarrow n_3$ are assumed by the algorithm to be faulty. The NCS equation system is

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & 0 & 1 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \hat{\delta}_{10} \\ \hat{\delta}_{20} \\ \hat{\delta}_{30} \\ \hat{\delta}_{40} \\ \hat{e}_{21} \\ \hat{e}_{31} \end{pmatrix} = \begin{pmatrix} \delta_{10} + e_{10} \\ \delta_{20} \\ \delta_{30} \\ \delta_{40} \\ \delta_{20} - \delta_{10} \\ \delta_{30} - \delta_{10} \\ \delta_{40} - \delta_{10} + e_{41} \\ \delta_{30} - \delta_{20} \\ \delta_{40} - \delta_{20} \\ \delta_{40} - \delta_{30} \end{pmatrix}. \tag{6}$$

If $e_{10} = -e_{41}$, the equation system has a unique solution of $\{\hat{\delta}_{10} = \delta_{10} + e_{10}, \hat{\delta}_{20} = \delta_{20}, \hat{\delta}_{30} = \delta_{30}, \hat{\delta}_{40} = \delta_{40}, \hat{e}_{21} = e_{10}, \hat{e}_{31} = e_{10}\}$, which violates the resilience condition. Thus, a 5-node network is not 2-resilient. □

In the proofs of Propositions 1, 3, and 5, we adopt an approach of enumerating counterexamples to prove that a network is not K-resilient. In the proofs of Propositions 3 and 5, if the actual faults satisfy certain conditions, the rank of $\mathbf{A}|\mathbf{b}$ may change, presenting a pitfall to the approach of enumerating counterexamples. This is a challenge in pursuing the general analysis for K-resilience. To address this challenge, in Section 5, we will introduce a fault-free NCS subgraph method to analyze the tight bound of complete NCS graphs.

4.3. **Simulations with Non-Faculty Synchronization Errors.** We conduct simulations to evaluate the impact of the non-faculty synchronization errors discussed in Section 3.3.1 on the performance of Algorithm 1. The simulations are for the small-scale example network topologies analyzed in Section 4.2. The p2p synchronization session follows a general error pattern of $e = \epsilon + x \cdot F$, where ϵ is a Gaussian noise following the standard normal distribution, the absolute value of the fault F is uniformly distributed within the range of [2, 8], the binary cofficient x is randomly sampled from $\{0, 1\}$. The synchronization error with $x = 1$ is viewed as a fault. The clock offset of each node with respect to the reference node is randomly and uniformly sampled from $[-10, 10]$. We use the least squares approach in Eq. (2) to generate the candidate solution of the NCS equation system. As discussed in Section 3.3.1, we apply a threshold of $\eta = 2$ to check the residual of each equation of the NCS equation system to decide whether a candidate solution is a valid solution. With the setting of $\eta = 2$, the probability of misclassifying a non-faulty synchronization error as a fault is $\Pr(|\epsilon| \geq 2) = 0.046$. For each network, we simulate a number of cases with different numbers of faults. For each case, we report the mean-square error (MSE) of the estimated clock offsets of all the nodes and whether the distribution of the estimated p2p synchronization faults is identical to the distribution of the actual faults.

Tables 1, 2, and 3 show the simulation results for the 3-node, 4-node, and 5-node networks, respectively. From Table 1, we can see that Algorithm 1 can opportunistically correct one fault, in which the distributions of the estimated faults and actual faults are identical and the MSE is small. When the distributions of the estimated p2p synchronization faults and the actual faults are not identical, the MSE of the estimated clock offsets is large. This means that Algorithm 1 cannot correct the faults. From Table 1, Algorithm 1 cannot correct more than one fault. This result is consistent with Proposition 1. Note that we have explained in Section 3.2 that, for a network that is not K-resilient, Algorithm 1 may be able to correct K faults with a specific distribution that happens to be identical to the first attempted fault distribution in Algorithm 1. From Tables 2 and 3, we can see that Algorithm 1 can always correct one fault and opportunistically correct two faults. This result is consistent with Propositions 2, 3, 4, and 5. From this set of simulation results, we can see that our analytic results provide good understanding for the scenarios with non-faulty synchronization errors.

TABLE 1. NCS results of a 3-node network with non-faulty synchronization errors.

Case	Number of faults	Identical?*	MSE of estimated clock offsets
1	1	yes	2.8721
2	1	no	14.5525
3	1	no	23.4256
4	1	no	37.2651
5	2	no	34.5164
6	2	no	53.2659
7	2	no	67.5983
8	3	no	79.3514

*This column indicates whether the distributions of estimated faults and actual faults are identical.

TABLE 2. NCS results of a 4-node network with non-faulty synchronization errors.

Case	Number of faults	Identical?*	MSE of estimated clock offsets
1	1	yes	1.1329
2	1	yes	1.9569
3	1	yes	0.5390
4	1	yes	2.1312
5	2	yes	2.0863
6	2	no	13.3885
7	2	no	40.8532
8	3	no	58.8646

*This column indicates whether the distributions of estimated faults and actual faults are identical.

TABLE 3. NCS results of a 4-node network with non-faulty synchronization errors.

Case	Number of faults	Identical?*	MSE of estimated clock offsets
1	1	yes	2.1296
2	1	yes	1.0586
3	1	yes	2.2766
4	2	yes	2.5198
5	2	no	13.5984
6	2	no	10.5354
7	3	no	19.8177
8	3	no	22.5147

*This column indicates whether the distributions of estimated faults and actual faults are identical.

5. TIGHT BOUND OF MAXIMUM RESILIENCE OF ANY NETWORK WITH COMPLETE NCS GRAPH

In this section, our analysis shows that the tight bound of maximum resilience of any N-node network with a complete NCS graph is $f_t(N) = \left\lfloor \frac{N}{2} \right\rfloor - 1$. Note that in

this section we change the notation $f_t(G)$ defined in Definition 5 to $f_t(N)$, because the complete NCS graph G solely depends on N. In what follows, we introduce the *fault-free NCS subgraph* (Section 5.1) and prove two lemmas (Section 5.2). The lemmas will be used to prove the tight bound of maximum resilience (Section 5.3).

5.1. **Fault-Free NCS Subgraph.** For a certain distribution of the estimated p2p synchronization faults among the $|E|$ sessions, we retain all the equations in Eq. (1) that contain neither estimated fault $\hat{e}_{ij}$ nor actual fault e_{ij} to generate an equation subsystem $\mathbf{A}_s\hat{\boldsymbol{\delta}} = \mathbf{b}_s$. This equation subsystem corresponds to a fault-free NCS subgraph $G_s = (V, E_s)$, where each edge in E_s represents a p2p synchronization session associated with neither estimated nor actual synchronization fault. The G_s is a subgraph of the original complete NCS graph G.

For instance, to generate the $\mathbf{A}_s\hat{\boldsymbol{\delta}} = \mathbf{b}_s$ of Eq. (6), we can remove the rows and columns of Eq. (6) as follows:

$$\begin{pmatrix} \cancel{1} & \cancel{0} & \cancel{0} & \cancel{1} & \cancel{0} \\ \cancel{0} & \cancel{1} & \cancel{0} & \cancel{0} & \cancel{0} \\ \cancel{0} & \cancel{0} & \cancel{1} & \cancel{0} & \cancel{1} \\ -1 & 1 & 0 & \cancel{0} & \cancel{0} \\ -1 & 0 & 1 & \cancel{0} & \cancel{0} \\ 0 & -1 & 1 & \cancel{0} & \cancel{0} \end{pmatrix} \begin{pmatrix} \hat{\delta}_{10} \\ \hat{\delta}_{20} \\ \hat{\delta}_{30} \\ \cancel{\hat{e}_{10}} \\ \cancel{\hat{e}_{30}} \end{pmatrix} = \begin{pmatrix} \cancel{\delta_{10} + e_{10}} \\ \cancel{\delta_{20} + e_{20}} \\ \cancel{\delta_{30}} \\ \delta_{20} - \delta_{10} \\ \delta_{30} - \delta_{10} \\ \delta_{30} - \delta_{20} \end{pmatrix}.$$

The first and the third rows of $\mathbf{A}$ and $\mathbf{b}$ are removed because they involve estimated faults $\hat{e}_{10}$ and $\hat{e}_{30}$. Specifically, the fourth element of $\mathbf{A}$'s first row that corresponds to $\hat{e}_{10}$ is 1; the last element of $\mathbf{A}$'s third row that corresponds to $\hat{e}_{30}$ is 1. The second row of $\mathbf{A}$ is removed because it involves the actual fault e_{20} from the second row of $\mathbf{b}$. The last two columns of $\mathbf{A}$ are removed because we no longer have $\hat{e}_{10}$ and $\hat{e}_{30}$. The remainders form $\mathbf{A}_s\hat{\boldsymbol{\delta}} = \mathbf{b}_s$, i.e.,

$$\begin{pmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} \hat{\delta}_{10} \\ \hat{\delta}_{20} \\ \hat{\delta}_{30} \end{pmatrix} = \begin{pmatrix} \delta_{20} - \delta_{10} \\ \delta_{30} - \delta_{10} \\ \delta_{30} - \delta_{20} \end{pmatrix}.$$

5.2. **The Lemmas.** Following the convention of graph theory, we say an undirected graph is *connected* when there is a path between every pair of vertices. We have the following lemmas.

Lemma 1. *For a complete NCS graph $G = (V, E)$ and a certain distribution of the estimated p2p synchronization faults, if the fault-free NCS subgraph G_s is connected, the NCS equation system $\mathbf{Ax} = \mathbf{b}$ has at most one solution.*

Proof. Since G_s is connected, we can find a traversal of G_s starting from n_0 and ending at any node n_i, which is represented by a list $\langle n_0, n_{w_1}, n_{w_2}, \ldots, n_{w_p}, n_i\rangle$. Note that, in the above list, two different symbols n_{w_x} and n_{w_y} may refer to the same node in the network. We can formulate a system of equations along the above traversal, where each equation corresponds to an edge connecting two consecutive nodes in the traversal. The equation system consists of $\hat{\delta}_{w_1 0} = \delta_{w_1 0}$, $\hat{\delta}_{w_1 0} - \hat{\delta}_{w_2 0} = \delta_{w_1 0} - \delta_{w_2 0}$, $\hat{\delta}_{w_2 0} - \hat{\delta}_{w_3 0} = \delta_{w_2 0} - \delta_{w_3 0}$, …, $\hat{\delta}_{w_{p-1} 0} - \hat{\delta}_{w_p 0} = \delta_{w_{p-1} 0} - \delta_{w_p 0}$, $\hat{\delta}_{w_p 0} - \hat{\delta}_{w_i 0} = \delta_{w_p 0} - \delta_{w_i 0}$. By substituting the solution of the previous equation to the next equation in the above chain of equations, we have a unique solution of $\hat{\boldsymbol{\delta}} = \boldsymbol{\delta}$.

We substitute the solution $\hat{\boldsymbol{\delta}} = \boldsymbol{\delta}$ into the original equation system $\mathbf{Ax} = \mathbf{b}$ to solve the remaining unknown variables $\hat{\mathbf{e}}$. There are three cases for the equations in $\mathbf{Ax} = \mathbf{b}$ but not in $\mathbf{A}_s\hat{\boldsymbol{\delta}} = \mathbf{b}_s$:

(1) For an actually faulty edge $n_i \leftrightarrow n_j$ that is correctly assumed to be faulty, the equation is $\hat{\delta}_{i0} - \hat{\delta}_{j0} + \hat{e}_{ij} = \delta_{i0} - \delta_{j0} + e_{ij}$. By substituting $\hat{\delta}_{i0} = \delta_{i0}$ and $\hat{\delta}_{j0} = \delta_{j0}$ (which are from $\hat{\boldsymbol{\delta}} = \boldsymbol{\delta}$) into the above equation, we have $\hat{e}_{ij} = e_{ij}$.
(2) For an actually non-faulty edge $n_i \leftrightarrow n_j$ that is wrongly assumed to be faulty, the equation is $\hat{\delta}_{i0} - \hat{\delta}_{j0} + \hat{e}_{ij} = \delta_{i0} - \delta_{j0}$. The solution is $\hat{e}_{ij} = 0$.
(3) For an actually faulty edge $n_i \leftrightarrow n_j$ that is wrongly assumed to be non-faulty, the equation is $\hat{\delta}_{i0} - \hat{\delta}_{j0} = \delta_{i0} - \delta_{j0} + e_{ij}$. Since $\hat{\delta}_{i0} = \delta_{i0}$, $\hat{\delta}_{j0} = \delta_{j0}$, and $e_{ij} \neq 0$ (which is the given condition), the above equation does not hold.

The $\mathbf{Ax} = \mathbf{b}$ that contains case (3) has no solution; the $\mathbf{Ax} = \mathbf{b}$ that does not contain case (3) has a unique solution that gives correct clock offset estimates. Thus, $\mathbf{Ax} = \mathbf{b}$ has at most one solution. □

Denote by $A \setminus B$ the relative complement of a set B with respect to a set A, i.e., the set of elements in A but not in B. We have the following lemma.

Lemma 2. *For a complete NCS graph $G = (V, E)$ and any edge subset $M \subseteq E$, a sufficient condition for the subgraph $G' = (V, E \setminus M)$ to be connected is $|M| \leq 2 \cdot \left(\lfloor \frac{N}{2} \rfloor - 1\right)$, where $N = |V|$.*

Proof. Let C_1 denote the clause of $|M| \leq 2 \cdot \left(\lfloor \frac{N}{2} \rfloor - 1\right)$; let C_2 denote the clause of G' is connected. From logic, we have the following equivalence: $(C_1 \Rightarrow C_2) \Leftrightarrow (\neg C_1 \Leftarrow \neg C_2)$, where $\neg$ represents negation. The clause $\neg C_1$ is $|M| > 2 \cdot \left(\lfloor \frac{N}{2} \rfloor - 1\right)$. As $|E| = \frac{N(N-1)}{2}$, the clause $\neg C_1$ is also equivalent to $|E \setminus M| \leq \frac{N(N-1)}{2} - 2 \cdot \left(\lfloor \frac{N}{2} \rfloor - 1\right)$. The clause $\neg C_2$ is that G' is disconnected. From the above reasoning, the sufficient condition to be proved is equivalent to the following: a sufficient condition for $|E \setminus M| \leq \frac{N(N-1)}{2} - 2 \cdot \left(\lfloor \frac{N}{2} \rfloor - 1\right)$ is that G' is disconnected. In the following, we prove this equivalent sufficient condition.

Since G' is disconnected, we assume that it has a total of P partitions, where $P \geq 2$. Let $N_p \in \mathbb{Z}_{>0}$ denote the number of vertices in the pth partition. Thus, $\sum_{p=1}^{P} N_p = N$. Define $N_r = N - N_1 = \sum_{p=2}^{P} N_p$. We have

$$\begin{aligned}
\frac{N_r(N_r-1)}{2} &= \frac{\left(\sum_{p=2}^{P} N_p\right)\left(\sum_{p=2}^{P} N_p - 1\right)}{2} \\
&= \sum_{p=2}^{P} \frac{N_p\left(N_p-1\right)}{2} + \sum_{\forall p,q \in [2,P], p \neq q} N_p N_q \\
&\geq \sum_{p=2}^{P} \frac{N_p(N_p-1)}{2}. \qquad (7)
\end{aligned}$$

As the number of edges of the pth partition is no greater than $\frac{N_p(N_p-1)}{2}$, we have $|E \setminus M| \leq \sum_{p=1}^{P} \frac{N_p(N_p-1)}{2} = \frac{N_1(N_1-1)}{2} + \sum_{p=2}^{P} \frac{N_p(N_p-1)}{2} \leq \frac{N_1(N_1-1)}{2} + \frac{N_r(N_r-1)}{2}$, where the last inequality follows from Eq. (7). By substituting $Nr = N - N_1$

into the above inequality, we have $|E \setminus M| \leq \frac{N(N-1)}{2} + N_1(N_1 - N)$. Note that $N_1 \in [1, N-1]$. When $N_1 = 1$ or $N_1 = N-1$, the quadratic $N_1(N_1 - N)$ achieves its maximum value of $-(N-1)$. Thus, $|E \setminus M| \leq \frac{N(N-1)}{2} - (N-1)$. As $-(N-1) < -2 \cdot \left(\lfloor \frac{N}{2} \rfloor - 1\right)$, we have $|E \setminus M| \leq \frac{N(N-1)}{2} - 2 \cdot \left(\lfloor \frac{N}{2} \rfloor - 1\right)$. □

5.3. **Tight Bound of Maximum Resilience.**

Theorem 1. *The tight bound of maximum resilience of any N-node network with a complete NCS graph $G = (V, E)$ is $f_t(N) = \lfloor \frac{N}{2} \rfloor - 1$.*

Proof. First, we prove that, if $K \leq f_t(N)$, the network is K-resilient. Let k denote the assumed number of faults in Algorithm 1, where $k \leq K$. For any distribution of the estimated faults, let M denote the set of edges excluded from E to generate the fault-free NCS subgraph G_s. Thus, $|M| \leq k + K$. Moreover, since $k \leq K \leq f_t(N)$, we have $|M| \leq k + K \leq 2 \cdot f_t(N) = 2 \cdot \left(\lfloor \frac{N}{2} \rfloor - 1\right)$. From Lemma 2, G_s is connected. From Lemma 1, the NCS equation system $\mathbf{Ax} = \mathbf{b}$ has at most one solution. Now, we verify the K-resilience condition in Definition 2 as follows:

(1) When $k < K$: There must exist an actually faulty edge wrongly assumed to be non-faulty, i.e., Case (3) in the proof of Lemma 1. Thus, the $\mathbf{Ax} = \mathbf{b}$ has no solution and Algorithm 1 will not return when $k < K$.
(2) When $k = K$: Only when the distribution of the estimated faults is correct, the $\mathbf{Ax} = \mathbf{b}$ does not encompass Case (3) in the proof of Lemma 1 and it must yield a solution that gives correct clock offset estimates. Otherwise, $\mathbf{Ax} = \mathbf{b}$ must encompass Case (3) in the proof of Lemma 1 and it has no solution.

Since the K-resilience condition holds, the network is K-resilient.

Second, we prove that, if $K > f_t(N)$, the network is not K-resilient. We prove it using an example network that is not K-resilient. The example network has the following two properties: (1) all actually faulty synchronization sessions and all assumed faulty synchronization sessions involve a certain node n_i; (2) each of the edges involving n_i is either actually faulty or assumed to be faulty, or both. For this example network, $k + K \geq N - 1$, where $N - 1$ is the total number of edges involving n_i. Moreover, as $k \leq K$, we have $2K \geq k + K \geq N - 1$ and $K \geq \frac{N-1}{2} > \lfloor \frac{N}{2} \rfloor - 1$. Thus, the example network satisfies $K > f_t(N)$. We now prove that this example network is not K-resilient. The fault-free NCS subgraph G_s is disconnected and has two partitions. One of them involving all nodes except n_i is a complete NCS graph without any fault. Thus, a unique partial solution that give correct clock offset estimate can be obtained for this partition. By substituting the partial solution into the original equation system $\mathbf{Ax} = \mathbf{b}$, any remaining equation that must involve n_i will be in one of the following three forms: (1) $\hat{\delta}_{i0} = \delta_{i0} + e_{ik}$, (2) $\hat{\delta}_{i0} + \hat{e}_{ik} = \delta_{i0} + e_{ik}$, and (3) $\hat{\delta}_{i0} + \hat{e}_{ik} = \delta_{i0}$. If all actual faults have identical value, i.e., $e_{ik} = e$, the remaining equations will yield a solution that gives wrong clock offset estimates, in which (1) $\hat{\delta}_{i0} = \delta_{i0} + e$, (2) $\hat{e}_{ik} = 0$, and (3) $\hat{e}_{ik} = -e$ that respectively correspond to the three forms. Thus, the network is not K-resilient. □

From Theorem 1, for networks with complete NCS graphs, the maximum number of correctable faults increases with N in a nearly linear manner. However, the number of edges increases with N quadratically. This suggests that, for networks with complete NCS graphs, the fault correction capability decreases with N. Thus, it is interesting to study whether we can remove edges from a complete NCS graph

while maintaining K-resilience. To answer this question, we analyze the resilience bounds for NCS graphs that may be incomplete (Section 6) and then analyze the minimum number of edges needed to ensure K-resilience (Section 8).

6. Algorithm to Compute Tight Bound of Maximum Resilience of any Network

In this section, we study the tight bound for any NCS graphs that may be incomplete. In Section 6.1, we interpret the K-resilience of any NCS graph from the edge connectivity of the graph. The interpretation is mainly from the Menger's theorem [1]. Based on the edge-connectivity interpretation, in Section 6.2, we present an algorithm to compute the tight bound of maximum resilience for any given NCS graph. Note that, different from Section 5 that gives the closed-form tight bound of maximum resilience of any complete NCS graph, the closed-form tight bound may not exist for NCS graphs that may be incomplete, because the K-resilience depends on the topology of the incomplete NCS graph.

6.1. Graph-Theoretic K-Resilience Condition.

First, we define *fault-free* path between any two nodes in a connected NCS graph. For any path between node n_i and n_j, if every edge in the path is associated with neither estimated nor actual synchronization fault, the path is called fault-free path. Note that for a certain distribution of the estimated p2p synchronization faults among the sessions, any path is either fault-free path or non-fault-free path.

Now, we introduce the concept of *edge-connectivity* [32] of any graph and *minimum edge cut* of any pair of nodes to extend the fault-free NCS subgraph method. In graph theory, a connected graph is *L-edge-connected* if it remains connected when any no greater than L edges are removed from the graph. The edge-connectivity of a graph is the largest L for which the graph is still L-edge-connected [32]. A minimum edge cut of any pair of nodes is an edge cut of the pair such that there is no other edge cut of the pair containing fewer edges. The Menger's theorem [1] stated below will be used to analyze the resilience of any NCS graph.

Theorem 2. ***Menger's theorem*** [1]**.** *In a graph G, the size of the minimum edge cut of any pair of nodes is equal to the maximum number of disjoint paths that can be found between the node pair. Extended to all node pairs, G is L-edge-connected if and only if every node pair has L edge-disjoint paths connecting them.*

Based on Menger's theorem, we have the following theorem regarding the resilience of any NCS graph. In the proof, we use examples provided in *italic text* to help understanding.

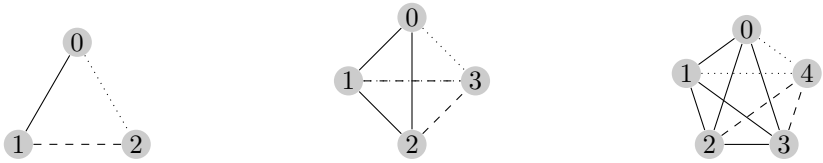
Theorem 3. *An NCS graph G is K-resilient if and only if it is $(2K+1)$-edge-connected.*

Proof. Let C_1 denote the clause that an NCS graph G is K-resilient; let C_2 denote the clause that the NCS graph is $(2K+1)$-edge-connected.

Proof of backward implication $C_2 \Rightarrow C_1$. Assuming that the NCS graph G is $(2K+1)$-edge-connected, from the Menger's theorem, there exist $2K+1$ edge-disjoint paths between any node n_j and the reference node n_0. In the case that the network has K actual faults and k estimated faults where $k \leq K$, the total number of edges associated with the estimated or actual synchronization faults is less than $K+k$. Note that $K+k \leq 2K$. From the principle of drawers,

there exists at least one fault-free path among the $2K+1$ edge-disjoint paths connecting n_j and n_0. We can formulate a system of equations along the above fault-free path where each equation corresponds to an edge. We denote this fault-free path as $\langle n_0, n_{w_1}, n_{w_2}, \ldots, n_{w_p}, n_j \rangle$. The equation system on the path consists of $\hat{\delta}_{w_1 0} = \delta_{w_1 0}$, $\hat{\delta}_{w_1 0} - \hat{\delta}_{w_2 0} = \delta_{w_1 0} - \delta_{w_2 0}$, $\hat{\delta}_{w_2 0} - \hat{\delta}_{w_3 0} = \delta_{w_2 0} - \delta_{w_3 0}$, …, $\hat{\delta}_{w_{p-1} 0} - \hat{\delta}_{w_p 0} = \delta_{w_{p-1} 0} - \delta_{w_p 0}$, $\hat{\delta}_{w_p 0} - \hat{\delta}_{w_j 0} = \delta_{w_p 0} - \delta_{w_j 0}$. Similar to the proof of Lemma 1, we can substitute the solution of the previous equation to the next equation in the above chain of equations to generate the solution of $\hat{\delta}_{j0} = \delta_{j0}$. By repeating the above process for every non-reference node n_j, we obtain a unique solution $\hat{\boldsymbol{\delta}} = \boldsymbol{\delta}$. We substitute the solution $\hat{\boldsymbol{\delta}} = \boldsymbol{\delta}$ into the original equation system $\mathbf{Ax} = \mathbf{b}$ to solve the remaining unknown variables $\hat{\mathbf{e}}$. As shown in the proof of Lemma 1, when $k = K$ and the distribution of the estimated p2p synchronization faults is identical to the distribution of the actual p2p synchronization faults, the NCS equation system $\mathbf{Ax} = \mathbf{b}$ has a unique solution that gives correct clock offset estimates. Otherwise, it has no solution. Therefore, if an NCS graph G is $(2K+1)$-edge-connected, it is K-resilient, i.e., $C_2 \Rightarrow C_1$.

Proof of forward implication $C_1 \Rightarrow C_2$. We have the following equivalence: $(C_1 \Rightarrow C_2) \Leftrightarrow (\neg C_1 \Leftarrow \neg C_2)$. The $\neg C_2$ means that the NCS graph G is not $(2K+1)$-edge-connected; $\neg C_1$ means that G is not K-resilient. In what follows, we prove $\neg C_1 \Leftarrow \neg C_2$. From the definition of K-resilience condition in Definition 2, G is not K-resilient if we can find any of the following counterexamples: (1) Algorithm 1 returns a solution when the distribution of the estimated p2p synchronization faults is different from the distribution of the actual faults or (2) Algorithm 1 returns more than one solution when the distribution of the estimated p2p synchronization faults is identical to the the distribution of the actual faults. In the following, we find such counterexamples when G is not $(2K+1)$-edge-connected. From the Menger's theorem, since G is not $(2K+1)$-edge-connected, there is a minimum edge cut C including at most $2K$ edges for a certain pair of nodes n_i and n_j. The minimum edge cut C partitions G into two connected subgraphs G_i and G_j that are disconnected from each other, where $n_i \in \mathcal{V}(G_i)$, $n_j \in \mathcal{V}(G_j)$, and $\mathcal{V}(G)$ represents the set of G's vertexes. Note that the reference node n_0 is either in



(A) $N = 3$, $K = 1$. (B) $N = 4$, $K = 2$. (C) $N = 5$, $K = 2$.

FIGURE 1. The counterexamples for the small-scale NCS graphs with $N = 3, 4, 5$. Dotted edges denote the edges containing the actual synchronization faults; dashed edges denote the edges containing the estimated synchronization faults; the dashed-dotted edges denote the edges containing both the estimated and actual synchronization faults. The combination of all the above three types of edges composes of the minimum cut C in a certain NCS graph. The solid edges are fault-free edges.

the subgraph G_i or the subgraph G_j. Without loss of generality, we assume that $n_0 \in \mathcal{V}(G_i)$. Our counterexamples satisfy the following conditions: (1) there are K actual faulty sessions and the number of estimated faults is equal to K, (2) all the estimated faults and the actual faults are on the edge cut C and each edge of the edge cut C is associated with an estimated fault, or an actual fault, or both of them, (3) all the actual faults have an identical value e. *For example, Fig. 1 shows the counterexamples for the small-scale networks discussed in Section 4.2.*

For the counterexamples described above, the following Eq. (8) is a solution of Eq. (1):

$$\begin{cases} \hat{\delta}_{k0} = \delta_{k0}, & \forall n_k \in \mathcal{V}(G_i); \\ \hat{\delta}_{k0} = \delta_{k0} - e, & \forall n_k \in \mathcal{V}(G_j); \\ \hat{e}_{ij} = 0, & \text{if there is an actual fault on } n_i \leftrightarrow n_j; \\ \hat{e}_{ij} = -e, & \text{if there is no actual fault on } n_i \leftrightarrow n_j. \end{cases} \tag{8}$$

For example, in Fig. 1(b), Eq. (8) is

$$\begin{cases} \hat{\delta}_{10} = \delta_{10}; \\ \hat{\delta}_{20} = \delta_{20}; \\ \hat{\delta}_{30} = \delta_{30} - e; \\ \hat{e}_{13} = 0; \\ \hat{e}_{23} = -e. \end{cases} \tag{9}$$

Now, we prove that Eq. (8) is a solution of Eq. (1) in the counterexamples. We prove it by substituting Eq. (8) to Eq. (1). If there is no conflict on each equation of the equation system in Eq. (1) (i.e., Eq. (1) still holds after incorporating Eq. (8)), Eq. (8) is a solution of Eq. (1). Note that Eq. (1) can be separated into three disjoint subequation systems corresponding to the subgraph G_i, G_j and the edge cut C. Thus, if we substitute Eq. (1) to the three disjoint subequation systems and there is no conflict on each equation, Eq. (8) is a solution of Eq. (1). Now, we analyze each of the disjoint subequation systems incorporated with Eq. (1) as follows.

(1) In the subequation system associated with G_i, since there are no edges associated with the estimated or actual synchronization fault, the corresponding equation in Eq. (1), denoted by $\mathcal{E}$, has the form of $\hat{\delta}_{w0} - \hat{\delta}_{v0} = \widetilde{\delta}_{wv} = \delta_{w0} - \delta_{v0}$, where $n_w \in \mathcal{V}(G_i)$ and $n_v \in \mathcal{V}(G_i)$. By substituting Eq. (8) to the left-hand side of $\mathcal{E}$, we have $\hat{\delta}_{w0} - \hat{\delta}_{v0} = \delta_{w0} - \delta_{v0}$, which is equal to the right-hand side of $\mathcal{E}$. Thus, there is no conflict when substituting Eq. (8) to the subequation system associated with G_i. *For example, in Fig. 1(b), the subequation system associated with G_i is $\{\hat{\delta}_{10} = \delta_{10}, \hat{\delta}_{20} = \delta_{20}, \hat{\delta}_{10} - \hat{\delta}_{20} = \delta_{10} - \delta_{20}\}$. Substituting Eq. (9) into the above subequation system does not result in any conflict.*

(2) In the subequation system associated with G_j, since there are no edges associated with the estimated or actual synchronization fault, the corresponding equation $\mathcal{E}$ in Eq. (1) has the form of $\hat{\delta}_{w0} - \hat{\delta}_{v0} = \widetilde{\delta}_{wv} = \delta_{w0} - \delta_{v0}$, where $n_w \in \mathcal{V}(G_j)$ and $n_v \in \mathcal{V}(G_j)$. By substituting Eq. (8) to the left-hand side of $\mathcal{E}$, we have $\hat{\delta}_{w0} - \hat{\delta}_{v0} = (\delta_{w0} - e) - (\delta_{v0} - e) = \delta_{w0} - \delta_{v0}$, which is equal to the right-hand side of $\mathcal{E}$. Thus, there is no conflict when

substituting Eq. (8) to the subequation system associated with G_j. *For example, in Fig. 1(b), the subequation system associated with G_j is an empty set $\emptyset$, because there is only one node in the subgraph G_j. Thus, there is no conflict between the empty set and the solution in Eq. (9).*

(3) In the subequation system associated with C, since each edge is associated with at least one fault between estimated and actual synchronization fault, the corresponding equation $\mathcal{E}$ in Eq. (1) can be in any of the following three forms:
   (a) If the edge corresponding to $\mathcal{E}$ contains an estimated synchronization fault but no actual synchronization fault, $\mathcal{E}$ has the form of $\hat{\delta}_{w0} - \hat{\delta}_{v0} + \hat{e}_{wv} = \widetilde{\delta}_{wv} = \delta_{w0} - \delta_{v0}$, where $n_w \in \mathcal{V}(G_i)$ and $n_v \in \mathcal{V}(G_j)$. By substituting Eq. (8) to the left-hand side of $\mathcal{E}$, we have $\hat{\delta}_{w0} - \hat{\delta}_{v0} + \hat{e}_{wv} = \delta_{w0} - (\delta_{v0} - e) + (-e) = \delta_{w0} - \delta_{v0}$, which is equal to the right-hand side of $\mathcal{E}$. Thus, there is no conflict when substituting Eq. (8) to the subequation system associated with C in this case. *For example, in Fig. 1(b), the subequation system associated with C in this case is $\hat{\delta}_{03} = \delta_{03} + e$, which is also in Eq. (9). Thus, there is no conflict.*
   (b) If the edge corresponding to $\mathcal{E}$ contains an actual synchronization fault but no estimated synchronization fault, $\mathcal{E}$ has the form of $\hat{\delta}_{w0} - \hat{\delta}_{v0} = \widetilde{\delta}_{wv} = \delta_{w0} - \delta_{v0} + e_{wv}$, where $n_w \in \mathcal{V}(G_i)$ and $n_v \in \mathcal{V}(G_j)$. By substituting Eq. (8) to the left-hand side of $\mathcal{E}$, we have $\hat{\delta}_{w0} - \hat{\delta}_{v0} = \delta_{w0} - (\delta_{v0} - e) = \delta_{w0} - \delta_{v0} + e$. Note that in our counterexamples, all the actual faults have an identical value e, i.e., $e_{wv} = e$. Thus, $\delta_{w0} - \delta_{v0} + e$ is equal to the left-hand side of $\mathcal{E}$. There is no conflict when substituting Eq. (8) to the subequation system associated with C in this case. *For example, in Fig. 1(b), the subequation system associated with C in this case is $\hat{\delta}_{20} - \hat{\delta}_{30} + \hat{e}_{23} = \delta_{30} - \delta_{20}$. Note that in Eq. (9), $\hat{e}_{23} = -e$. Therefore, if we substitute Eq. (9) into the above subequation system, there is no conflict.*
   (c) If the edge corresponding to $\mathcal{E}$ contains both an actual synchronization fault and an estimated synchronization fault, $\mathcal{E}$ has the form of $\hat{\delta}_{w0} - \hat{\delta}_{v0} + \hat{e}_{wv} = \widetilde{\delta}_{wv} = \delta_{w0} - \delta_{v0} + e_{wv}$, where $n_w \in \mathcal{V}(G_i)$ and $n_v \in \mathcal{V}(G_j)$. Note that in this case $\hat{e}_{wv} = 0$ in Eq. (8). By substituting Eq. (8) to the left-hand side of $\mathcal{E}$, we have $\hat{\delta}_{w0} - \hat{\delta}_{v0} + \hat{e}_{wv} = \delta_{w0} - (\delta_{v0} - e) + 0 = \delta_{w0} - \delta_{v0} + e$, which is equal to the right-hand side of $\mathcal{E}$. Thus, there is no conflict when substituting Eq. (8) to the subequation system associated with C in this case. *For example, in Fig. 1(b), the subequation system associated with C in this case is $\hat{\delta}_{10} - \hat{\delta}_{30} + \hat{e}_{13} = \delta_{10} - \delta_{30} + e$. Note that in Eq. (9), $\hat{e}_{13} = 0$. Therefore, substituting Eq. (9) into the above subequation system results in no conflict.*

   Recall that in our counterexamples, each edge of the edge cut C is associated with an estimated fault, or an actual fault, or both of them. Therefore, there is no conflict when substituting Eq. (8) to the subequation system associated with C.

In summary, we have substituted Eq. (8) to the three disjoint subequation systems and there is no conflict. Thus, Eq. (8) is a solution of Eq. (1). Now, we prove that in this case, G is not K-resilient. If the distribution of the estimated

Algorithm 2 Compute the tight bound of maximum resilience

Given: NCS equation system $\mathbf{Ax} = \mathbf{b}$ and the corresponding NCS graph $G = (V, E)$
Output: Tight bound of maximum resilience of G

```
K ← 0
while K ≤ |E| do
  for each combination of 2K edges selected from all the |E| edges in G do
    remove the selected 2K edges to generate a subgraph G'
    if G' is not connected then
      return K − 1
    end if
  end for
  K ← K + 1
end while
```

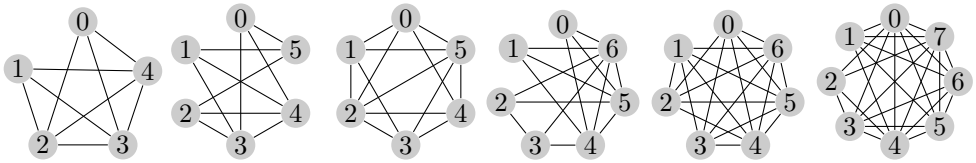
p2p faults is identical to the distribution of the actual faults, Eq. (1) has at least two solutions including Eq. (8) and the solution $\{\hat{\boldsymbol{\delta}} = \boldsymbol{\delta}, \hat{\boldsymbol{e}} = \boldsymbol{e}\}$, which violates the condition (2)-(a) of the K-resilience condition defined in Definition 2. If the two distributions are different, Eq. (1) has a solution of Eq. (8), which violates the condition (2)-(b) of the K-resilience condition defined in Definition 2. Therefore, for any NCS graph G that is not $(2K+1)$-edge-connected, G is not K-resilient, i.e., $\neg C_1 \Leftarrow \neg C_2$. Therefore, $C_1 \Rightarrow C_2$. □

6.2. **Algorithm to Compute Tight Bound of Maximum Resilience.** Based on Theorem 3, Algorithm 2 computes the tight bound of the maximum resilience for NCS graph G. Specifically, starting with $K = 0$, Algorithm 2 increases K by one in each step of the outer loop to check whether the NCS graph is K-resilient by checking the connectivity of the subgraphs after removing $2K$ edges from G. If any subgraph is not connected, the sufficient and necessary condition given by Theorem 3 is not satisfied for the current K value. Thus, the algorithm returns $K - 1$ as the tight bound.

Now, we analyze the time complexity of Algorithm 2. For each K value, Algorithm 2 needs to check the connectivity of totally $\binom{|E|}{2K}$ subgraphs. Existing graph-theoretic algorithms can be used to check the connectivity of a graph, such as depth-first search (DFS) and breadth-first search (BFS) [3]. The DFS and BFS algorithms have the same time complexity of $O(|V| + |E|)$. In particular, for complete NCS graphs, the complexity of the two algorithms is $O(|V|^2)$. Thus, the time complexity of the Kth step of Algorithm 2 is $O\left(|V|^2 \binom{\frac{|V||V-1|}{2}}{2K}\right)$. Therefore, determining the tight bound of maximum resilience for any graph incurs a high computation overhead for large-scale NCS graphs. Nevertheless, Algorithm 2 is a method to exactly compute the tight bound of maximum resilience for incomplete NCS graphs. Figure 2 shows several incomplete NCS graphs and their tight bounds of maximum resilience computed by Algorithm 2.

7. Fast NCS Algorithm with Fault Correction

Algorithm 1 enumerates all possible distributions of the faults, leading to the exponential time complexity in the worst case. From the proof of Theorem 3, if we



(A) $N = 5$, $f_t(G) = 1$. (B) $N = 6$, $f_t(G) = 1$. (C) $N = 6$, $f_t(G) = 1$. (D) $N = 7$, $f_t(G) = 1$. (E) $N = 7$, $f_t(G) = 2$. (F) $N = 8$, $f_t(G) = 2$.

FIGURE 2. The tight bounds of maximum resilience for several incomplete NCS graphs with $N = 5, 6, 7, 8$.

can find a fault-free path connecting n_i and n_0, we can obtain the correct estimate of clock offset between n_i and n_0. This observation sheds light on a new NCS algorithm that can correct the faults without enumerating all possible distributions of the faults. In this section, we present such a new NCS algorithm. Then, we show that the new NCS algorithm achieves the same fault correction capability as Algorithm 1 and analyze the time complexity of the new NCS algorithm.

7.1. **Fast NCS Algorithm.** First, we prove that we can correctly estimate the p2p clock synchronization offsets on the fault-free path. We have the following lemma.

Lemma 3. *Any fault-free path between the reference node n_0 and node n_i leads to the correct estimate of n_i's clock offset, i.e., $\hat{\delta}_i = \delta_i$.*

Proof. Denote the path by $\langle n_0, n_{w_1}, n_{w_2}, \ldots, n_{w_p}, n_i \rangle$. We can formulate a system of equations along the path. Since all edges on the path are associated with neither estimated nor actual synchronization fault, the equation system consists of a chain $\hat{\delta}_{w_1 0} = \delta_{w_1 0}$, $\hat{\delta}_{w_1 0} - \hat{\delta}_{w_2 0} = \delta_{w_1 0} - \delta_{w_2 0}$, $\hat{\delta}_{w_2 0} - \hat{\delta}_{w_3 0} = \delta_{w_2 0} - \delta_{w_3 0}$, …, $\hat{\delta}_{w_{p-1} 0} - \hat{\delta}_{w_p 0} = \delta_{w_{p-1} 0} - \delta_{w_p 0}$, $\hat{\delta}_{w_p 0} - \hat{\delta}_{w_i 0} = \delta_{w_p 0} - \delta_{w_i 0}$. By substituting the solution of the previous equation to the next equation in the above chain of equations, we have a solution that $\hat{\delta}_i = \delta_i$. □

From Lemma 3, if for every node n_i in G we can find at least one fault-free path connecting n_i and n_0, we can obtain all the correct clock offset estimates, i.e., $\hat{\boldsymbol{\delta}} = \boldsymbol{\delta}$. Then, we can use the solution $\hat{\boldsymbol{\delta}} = \boldsymbol{\delta}$ to pinpoint the faulty p2p synchronization sessions. Specifically, if $\widetilde{\delta}_{ij} \neq \hat{\delta}_i - \hat{\delta}_j$, where $\hat{\delta}_i$ and $\hat{\delta}_j$ are from $\hat{\boldsymbol{\delta}}$, the p2p synchronization session between n_i and n_j is faulty. The fault is given by $e_{ij} = \widetilde{\delta}_{ij} - (\hat{\delta}_i - \hat{\delta}_j)$.

Thus, the NCS problem becomes how to find a fault-free path between any node n_i and the reference node n_0. This is challenging because the system has no knowledge of the number of faults and their distribution among the $|E|$ sessions. We address this challenge using a voting scheme. The details are as follows. We let Z_i denote the maximum number of pairwise edge-disjoint paths connecting n_i and n_0 and let S_i denote a set of such paths. Existing algorithms can be used to compute Z_i and S_i, such as those presented in [12], [31] and [24]. The worst-case time complexity of these algorithms is $O(|V|^2)$. Assuming every path in S_i is fault-free, we apply the approach described in Lemma 3 to compute $\hat{\delta}_i$ for every path in S_i. Note that there might be multiple different sets of pairwise edge-disjoint

Algorithm 3 Fast NCS algorithm with fault correction.

Given: $\{\widetilde{\delta}_{ij}|\forall n_i \leftrightarrow n_j \in E\}$
Output: $\{\hat{\delta}_{j0}|\forall j \in [1, N-1]\}$ and $\{\hat{e}_{ij}|\forall n_i \leftrightarrow n_j \in E\}$
1: **for** each node n_i in V where $i \neq 0$ **do**
2: compute the maximum number of pairwise edge-disjoint paths Z_i and find a corresponding set of such paths S_i
3: **for** each path $P_k \in S_i$ **do**
4: compute the corresponding value of the estimated clock offset $\hat{\delta}^k_{i0}$
5: **end for**
6: $\hat{\delta}_{i0} \leftarrow$ the most frequent value in $\hat{\delta}^k_{i0}$, where $k \in \{1, 2, 3, ..., Z_i\}$
7: **end for**
8: **for** each synchronization session $n_i \leftrightarrow n_j$ **do**
9: **if** $\widetilde{\delta}_{ij} - (\hat{\delta}_i - \hat{\delta}_j) \neq 0$ **then**
10: $\hat{e}_{ij} = \widetilde{\delta}_{ij} - (\hat{\delta}_i - \hat{\delta}_j)$
11: **end if**
12: **end for**
13: **return** $\{\hat{\delta}_{j0}|\forall j \in [1, N-1]\}$ and $\{\hat{e}_{ij}|\forall n_i \leftrightarrow n_j \in E\}$

paths with the identical set cardinality. The S_i used in the following discussion can be any one of them. If all the paths in the set S_i are really fault-free, their corresponding estimated clock offsets should be the same. Otherwise, they will be different. We use the most frequent value among all the clock offset estimates as the voting result, which is yielded as the final clock offset estimate $\hat{\delta}_i$. We repeat the above process for every node n_i to generate the voting result $\hat{\delta}_i$. After that, we can correct the faults by following the procedure described in last paragraph. Algorithm 3 shows the pseudocode of the new NCS algorithm. Algorithm 3 has the same practicality as Algorithm 1 in that it requires neither the actual number nor the actual distribution of the p2p synchronization faults.

7.2. **Tight Bound of Maximum Resilience of Fast NCS with Fault Correction.** In this section, we show that Algorithms 1 and 3 have the same fault correction capability. Therefore, the networks with Algorithms 1 and 3 as the NCS algorithm respectively have the same tight bound of maximum resilience.

Theorem 4. *For any NCS graph G, Algorithms 1 and 3 achieve the same tight bound of maximum resilience.*

Proof. First, we prove that, if an NCS graph is K-resilient under Algorithm 1, it is also K-resilient under Algorithm 3. From Theorem 3, the K-resilience of G under Algorithm 1 is equivalent to that G is $(2K+1)$-edge-connected. From the Menger's theorem, the $(2K+1)$-edge-connectivity means that for any pair of nodes n_i and n_j, there exist at least $2K+1$ edge-disjoint paths connecting them. Since the number of actual faults is no greater than K, there are at least $K+1$ fault-free paths between n_i and n_0. Thus, the majority voting in Algorithm 3 must give the correct result and Algorithm 3 can correct the faults. Therefore, the NCS graph is also K-resilient under Algorithm 3.

Then, we prove that, if an NCS graph G is not K'-resilient under Algorithm 1, it is also not K'-resilient under Algorithm 3. We assume G with Algorithm 1 can

correct at most K faults, where $K' \geq K+1$. From Theorem 3, G is $(2K+1)$-edge-connected. Thus, there exists at least one node pair n_i and n_j that have $2K+1$ edge-disjoint paths and no more connecting them. Note that the system's resilience is independent from the choice of reference node, i.e., any node can be designated as the reference node n_0. Without loss of generality, we designate n_j as n_0. Now, we consider the cases where there are K' faults, i.e., there are at least $K+1$ faults since $K' \geq K+1$. For the case where all the $K+1$ faults occur on the paths among the $2K+1$ edge-disjoint paths, the remaining fault-free edge-disjoint paths do not form the majority of Algorithm 3's voting. As a result, Algorithm 3 cannot correctly estimate the clock offset of n_i and cannot correct the faults. Thus, the G with Algorithm 3 is not K'-resilient. □

Now, we analyze the time complexity of Algorithms 1 and 3. If the algorithm in [24] is used to compute Z_i and S_1, Line 2 of Algorithm 3 has a time complexity of $O(|V|^2)$. The loop from Line 3 to Line 5 has a time complexity of $O(|V|)$, because from Theorems 1 and 3, the maximum number of pairwise edge-disjoint paths is less than $\left\lfloor \frac{|V|}{2} \right\rfloor - 1$. Line 6 has a time complexity of $O(|V|)$ [23]. Thus, the loop from Line 1 to Line 7 has a time complexity of $O(|V|^3)$. The loop from Line 8 to Line 12 has a time complexity $O(|V|^2)$. Therefore, the time complexity of Algorithm 3 is $O(|V|^3 + |V|^2) = O(|V|^3)$.

In Section 3.1, we have shown that the time complexity upper bound of Algorithm 1 is $O(2^{|E|})$. Now, we derive the time complexity lower bound of Algorithm 1 for complete NCS graphs that are K-resilient. When there are K faults, the time complexity of Algorithm 1 is $O\left(\sum_{k=0}^{K} \binom{|E|}{k}\right)$. From Theorem 1, Algorithm 1 can correct at most $K = \lfloor \frac{N}{2} \rfloor - 1$ faults. Thus, the time complexity of Algorithm 1 is $O\left(\sum_{k=0}^{\lfloor \frac{N}{2} \rfloor - 1} \binom{\frac{N(N-1)}{2}}{k}\right)$. When $N > 4$, we have the following inequality:

$$\sum_{k=0}^{\lfloor \frac{N}{2} \rfloor - 1} \binom{\frac{N(N-1)}{2}}{k} > \sum_{k=0}^{N/2} \binom{N/2}{k} = \sqrt{2}^N. \tag{10}$$

Thus, Algorithm 1 has an exponential complexity.

From the above analysis, Algorithm 3 achieves the same fault correction capability as Algorithm 1 with a cubic time complexity. In practice, Algorithm 3 should be used. Note that as Algorithm 1 is intuitive, the definition of fault resilience based on Algorithm 1 is also intuitive. Differently, the development of Algorithm 3 is based on our further analysis on the fault resilience. As a result, the fault resilience notion behind Algorithm 3 is not direct. From this sense, Algorithm 1, though not scalable to the network size, helps achieve a clear definition of fault resilience and is still a basis of this paper.

8. Minimum NCS Graph for K-Resilience

Section 5 and Section 6 analyzed the bounds of the number of faults that Algorithm 1 and Algorithm 3 can correct. Differently, in this section, we aim at minimizing the number of p2p synchronization sessions while maintaining the K-resilience of a network under Algorithms 1 and 3. In other words, we aim at looking for the *minimum NCS graph* for K-resilience, which is formally defined as follows.

Algorithm 4 Compute minimum NCS graphs for any N-node network with K-resilience

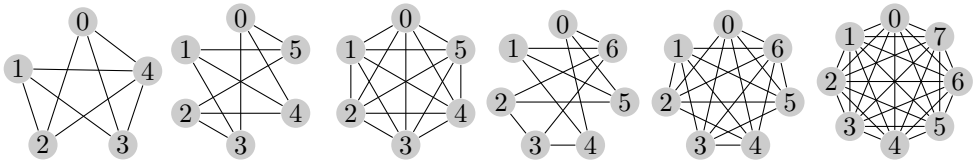
Given: The number of nodes N, resilience value K
Output: A set of minimum NCS graphs $\mathcal{G}$

$m \leftarrow 0$, $\mathcal{G} \leftarrow \emptyset$
construct the N-node complete NCS graph $G_c = (V, E_c)$
while $m \leq \frac{N(N-1)}{2}$ **do**
  $\mathcal{G}_{\text{current}} \leftarrow \emptyset$
  for each combination of m sessions among E_c **do**
    remove the m sessions from G_c to generate an NCS subgraph $G' = (V, E')$
    resilient $\leftarrow$ **true**
    for each combination of $2K$ equations selected from all the $|E'|$ sessions **do**
      remove the selected $2K$ edges to generate an NCS subgraph G''
      if G'' is unconnected **then**
        resilient $\leftarrow$ **false** // G' is not K-resilient
        break
      end if
    end for
    if *resilient* = **true then**
      $\mathcal{G}_{\text{current}} \leftarrow \mathcal{G}_{\text{current}} \cup \{G'\}$
    end if
  end for
  if $\mathcal{G}_{\text{current}} = \emptyset$ **then**
    return $\mathcal{G}$ // each G' is not K-resilient for current m
  else
    $\mathcal{G} \leftarrow \mathcal{G}_{\text{current}}$
  end if
  $m \leftarrow m + 1$
end while

Definition 6 (Minimum NCS graph for K-resilience)**.** Denote by V a set of N nodes. An NCS graph $G = (V, E)$ is a minimum NCS graph for K-resilience if the network with G is K-resilient and any network with the NCS graph $G' = (V, E')$ where $|E'| < |E|$ is not K-resilient.

With minimum NCS graphs, we can minimize the communication cost of NCS without compromising fault correction capability. In Section 8.1, we develop an algorithm based on the K-resilience's sufficient and necessary condition given by Theorem 3 to compute the minimum NCS subgraphs and show several examples. In Section 8.2, we derive the theoretic lower bound of the number of edges in the NCS graph that provides K-resilience. The theoretic lower bound can be used to understand the order of magnitude of the number of edges in a computed minimum NCS graph. In particular, the number of edges of a computed minimum NCS graph is identical to the theoretic lower bound in our computed examples. This implies that the theoretic lower bound is tight.

8.1. **The Algorithm to Compute Minimum NCS Graphs.** Based on Theorem 3, Algorithm 4 finds the minimum NCS graphs for any N-node network to



(A) $N = 5$, $K = 1$. (B) $N = 6$, $K = 1$. (C) $N = 6$, $K = 2$. (D) $N = 7$, $K = 1$. (E) $N = 7$, $K = 2$. (F) $N = 8$, $K = 3$.

FIGURE 3. Minimum NCS graphs providing K-resilience computed by Algorithm 4 under different K and N settings.

ensure K-resilience. Note that a network may have multiple different minimum NCS graphs. Algorithm 4 returns a set of minimum NCS graphs that have the same number of edges. We now explain Algorithm 4. The algorithm uses an N-node complete NCS graph $G_c = (V, E_c)$ as the basis to look for the minimum NCS graphs. In each iteration of the while loop (from Line 3 to Line 25), the m is increased by one from zero, where the m represents the number of edges removed from the E_c of G_c to generate a candidate minimum NCS graph G' (Line 6). For each possible G', the snippet from Line 7 to Line 14 uses Theorem 3 to check whether G' is K-resilient. If G' is K-resilient, G' is included into a set $\mathcal{G}_{\text{current}}$ (Line 16) that is reset to an empty set for the next m value (Line 4). If all possible G' graphs with the current m value cannot be confirmed K-resilient (i.e., $\mathcal{G}_{\text{current}} = \emptyset$), the algorithm returns the non-empty $\mathcal{G}_{\text{current}}$ in the previous iteration of the while loop. This mechanism is implemented by Line 19 to Line 24. The algorithm gives the minimum NCS graphs, because the snippet from Line 7 to Line 14 can confirm the K-resilience of the candidate G'.

Fig. 3 shows several minimum NCS graphs computed by Algorithm 4 under different settings of N and K. In Section 8.2, we will show that the number of edges in all these graphs are equal to the theoretic lower bound.

8.2. **Lower Bound of the Number of Edges for K-Resilience.** In this section, for any NCS graph $G = (V, E)$ that provides K-resilience, we derive a lower bound of $|E|$. We develop the following lemma that will be used to derive the bound.

Lemma 4. *A necessary condition for an NCS graph $G = (V, E)$ to give K-resilience is* $\text{mindeg}(G) \geq 2K+1$ *where* $\text{mindeg}(G)$ *denotes the minimum degree of all vertexes of G.*

Proof. Theorem 3 shows that an NCS graph G is K-resilient if and only if it is $(2K + 1)$-edge-connected. From Whitney's theorem [33], the minimum number of edges whose deletion results in disconnectivity of G is no greater than the minimum degree $\text{mindeg}(G)$. Thus, a necessary condition for an NCS graph $G = (V, E)$ to give K-resilience is $\text{mindeg}(G) \geq 2K + 1$. □

Theorem 5. *For an NCS graph $G = (V, E)$ providing K-resilience, $|E| \geq \left\lceil \frac{N(2K+1)}{2} \right\rceil$, where $N = |V|$. In other words, a necessary condition for G to be K-resilient is $|E| \geq \left\lceil \frac{N(2K+1)}{2} \right\rceil$.*

Proof. Let C_1 denote the clause that G is K-resilient; let C_2 denote the clause that $\text{mindeg}(G) \geq 2K+1$. Lemma 4 can be represented in logic as: $C_1 \Rightarrow C_2$.

From Lemma 4, to realize K-resilience, each node in G should have a degree of at least $2K+1$. We adopt a greedy algorithm to construct the graph G^* with the minimum number of edges subject to the condition of $\text{mindeg}(G^*) \geq 2K+1$. The algorithm is as follows. Starting from no edges, each step of the algorithm adds an edge to connect two nodes that do not have an edge and the degree of each of them is no greater than any other nodes. The algorithm terminates once the condition $\text{mindeg}(G^*) \geq 2K+1$ is satisfied. The resulting graph of this algorithm is as follows. First, when N is an even number, the degree of every node is $2K+1$. The total number of edges is $\frac{N}{2} \cdot (2K+1)$. Second, when N is an odd number, there are a total of $(N-1)$ nodes each having a degree of $(2K+1)$ and the remaining one node having a degree of $(2K+2)$. The total number of edges is $\left\lceil \frac{N(2K+1)}{2} \right\rceil$. In summary, the minimum number of edges to meet $\text{mindeg}(G^*) \geq 2K+1$ is $\left\lceil \frac{N(2K+1)}{2} \right\rceil$. Denoting by C_3 the clause of $|E| \geq \left\lceil \frac{N(2K+1)}{2} \right\rceil$, the above result can be represented in logic as $C_2 \Rightarrow C_3$.

Since $C_1 \Rightarrow C_2$ and $C_2 \Rightarrow C_3$, we have $C_1 \Rightarrow C_3$, i.e., C_3 is a necessary condition for C_1. □

The lower bound given by Theorem 5 can be used to understand the order of magnitude of the number of edges in a computed minimum NCS graph. Table 4 shows the lower bound values under several settings of N and K as well as the numbers of the edges of the corresponding minimum NCS graphs shown in Fig. 3. We can see that the minimums are identical to the lower bound values, which implies that the theoretic lower bound is tight. Thus, we can see that the relationship between the communication overhead (which is characterized by the number of edges) and the network size N is roughly linear. Therefore, order-wise, the communication overhead for achieving K-resilience is acceptable.

In the traditional synchronization methods that do not provide any fault correction capbility, one slave node synchronizes with only one master node. Thus, the number of edges in the NCS graph of the traditional synchronization methods is $N-1$. From the result given by Theorem 5, the additional communication overhead for K-resilience is at least $\left\lceil \frac{N(2K+1)}{2} \right\rceil - (N-1) = \left\lceil \frac{N(2K-1)}{2} \right\rceil + 1$. For instance, when $K=1$ and $N=8$, the additional communication overhead is at least five p2p synchronization sessions.

9. Implication of Results

This section discusses several important implications of the analytic results obtained in the previous sections.

9.1. **The Most Fault-Resilient Network.**

If every p2p synchronization session has the same fault rate, the *degree of resilience* (DoR) defined as the ratio of the maximum number of correctable faults (i.e., K) and the number edges in an NCS graph (i.e., $|E|$) becomes a meaningful metric that characterizes the allowable percentage of faulty p2p synchronization sessions. We have the following corollary.

Corollary 1. *The 4-node network with complete NCS graph achieves the highest DoR.*

TABLE 4. The lower bound of the number of edges in NCS graph providing K-resilience and the number of edges in the computed minimum NCS graphs shown in Fig. 3, as well as the upper bound of degree of resilience (DoR) that is defined in Section 9.1.

N	K	Lower bound from Theorem 5	Number of edges in Fig. 3	Upper bound of DoR
5	1	8	8	1/8
6	1	9	9	1/9
6	2	15	15	1/7.5
7	1	11	11	1/11
7	2	18	18	1/9
8	3	28	28	$1/9.\dot{3}$

Proof. We use the lower bound given by Theorem 5 to derive an upper bound of DoR when $N \geq 4$:

$$\begin{aligned} \text{DoR} &\leq \frac{K}{\left\lfloor \frac{N(2K+1)}{2} \right\rfloor - 1} \leq \frac{K}{\frac{N(2K+1)}{2} - 2} = \frac{2K}{N(2K+1) - 4} \\ &\leq \frac{2K+1}{N(2K+1)-4} = \frac{1}{N - \frac{4}{2K+1}} \leq \frac{1}{N - \frac{4}{3}}, \end{aligned} \tag{11}$$

where the last inequality follows from $K \geq 1$. Therefore, $\text{DoR} = O\left(\frac{1}{N}\right)$, suggesting that larger networks will have lower degree of resilience when N is large enough.

From Theorem 1, the DoR of the 4-node network is 1/6. To ensure that the DoR upper bound given in Eq. (11) is smaller than 1/6 (i.e., $\frac{1}{N-4/3} < 1/6$), we have $N \geq 8$. In other words, when $N \geq 8$, the network's DoR must be smaller than 1/6. Now, we check the DoRs of the networks when $N \in [5, 7]$. The last column of Table 4 gives the upper bound of DoR that is the ratio of K and the third column (i.e., the lower bound of the number of edges from Theorem 5). From the results, we can see that when $N \in [5, 7]$, the upper bound of DoR is smaller than 1/6. Therefore, the 4-node network achieves the highest DoR of 1/6. □

9.2. **Tiered Clock Synchronization for Fault Resilience.** The result in Section 9.1 suggests that, for a large-scale network, we can group the nodes into 4-node synchronization groups, forming the first tier of the clock synchronization. Each synchronization group with a complete NCS graph will use Algorithm 3 to correct at most one fault. Every four central nodes from four tier-1 synchronization groups form a synchronization group in the second tier of the clock synchronization. Similarly, each tier-2 synchronization group will use Algorithm 3 to correct at most one fault. More tiers are formed until all nodes in the network are connected. The NCS is executed from top to down in the tiered architecture.

We now use an example to illustrate. Suppose a network has 16 nodes. A two-tier clock synchronization with four 4-node tier-1 synchronization groups and one 4-node tier-2 synchronization group can be formed, as illustrated in Fig. 4. The tier-2 synchronization group executes NCS first. Then, each of the tier-1 synchronization group executes its own NCS. As such, all 16 nodes can be synchronized even if each synchronization group has a p2p synchronization faults (i.e., totally five faults). Note that the total number of edges in this two-tier network is 30. Alternatively,

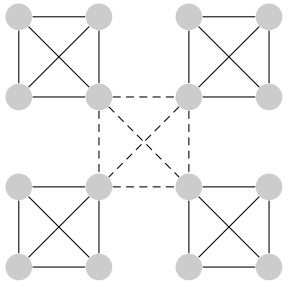


FIGURE 4. A two-tier 16-node clock synchronization architecture consisting of four 4-node tier-1 synchronization groups (solid lines) and one 4-node tier-2 synchronization groups (dashed lines).

we can also use Algorithm 4 to construct the minimum NCS graph for the 16-node network without the tiered architecture. From Theorem 5, the minimum NCS graph that provides 5-resilience will have at least $\left\lceil \frac{16\times(2\times5+1)}{2} \right\rceil = 88$ edges. Therefore, there is a trade-off between the above two solutions. In the minimum NCS graph without the tiered architecture, the five faults can occur on any five edges. However, the number of edges of the minimum NCS graph will be about three times of the tiered architecture shown in Fig. 4. On the other hand, while the tiered architecture uses less edges and thus incurs less communication cost, the five faults that the network can correct need to be distributed among the five synchronization groups.

10. Conclusion

This paper studied the resilience of network clock synchronization based on practical p2p synchronization fault correction algorithms. Our analysis gave the following results:

(1) A closed-form tight bound of the maximum number of faults that can be corrected when every node pair in the network performs p2p synchronization, with respect to the number of nodes N. The tight bound is $\lfloor \frac{N}{2} \rfloor - 1$.
(2) An algorithm to compute the tight bound of the maximum number of faults that can be corrected when not every node pair performs p2p synchronization.
(3) A fast NCS algorithm with a time complexity of $O(N^3)$ that achieves the same fault correction capability as the original NCS algorithm that has an exponential time complexity.
(4) An algorithm that minimizes the number of p2p synchronization sessions while ensuring that a specified number of faults can be corrected.
(5) A theoretic lower bound of the number of p2p synchronization sessions needed to correct K faults. The lower bound is $\left\lceil \frac{N(2K+1)}{2} \right\rceil$.

Lastly, we showed that the 4-node network achieves the highest degree of resilience. Based on this, we discussed a tiered clock synchronization architecture that provides understood resilience and requires reduced p2p synchronization sessions. The results in this paper provide important understanding on the resilience of network

clock synchronization against p2p synchronization faults and useful guidelines for the design of resilient clock synchronization systems.

References

1. Thomas Böhme, Frank Göring, and Jochen Harant, *Menger's theorem*, Journal of Graph Theory **37** (2001), no. 1, 35–36.
2. Yin Chen, Qiang Wang, Marcus Chang, and Andreas Terzis, *Ultra-low power time synchronization using passive radio receivers*, Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), IEEE, 2011, pp. 235–245.
3. Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein, *Introduction to algorithms*, MIT press, Cambridge, MA, 2009.
4. Danny Dolev, Joseph Y Halpern, and H Raymond Strong, *On the possibility and impossibility of achieving clock synchronization*, Journal of Computer and System Sciences **32** (1986), no. 2, 230–250.
5. Adwait Dongare, Patrick Lazik, Niranjini Rajagopal, and Anthony Rowe, *Pulsar: A wireless propagation-aware clock synchronization platform*, Proceedings of The IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), IEEE, 2017, pp. 283–292.
6. Jeremy Elson, Lewis Girod, and Deborah Estrin, *Fine-grained network time synchronization using reference broadcasts*, ACM SIGOPS Operating Systems Review **36** (2002), no. SI, 147–163.
7. Saurabh Ganeriwal, Ram Kumar, and Mani B Srivastava, *Timing-sync protocol for sensor networks*, Proceedings of The 1st International Conference on Embedded Networked Sensor Systems (SenSys), ACM, 2003, pp. 138–149.
8. Chaojie Gu, Linshan Jiang, Rui Tan, Mo Li, and Jun Huang, *Attack-aware data timestamping in low-power synchronization-free lorawan*, Proceedings of The 40th IEEE International Conference on Distributed Computing Systems (ICDCS), IEEE, 2020.
9. Jayant Gupchup, Răzvan Musăloiu-e, Alex Szalay, and Andreas Terzis, *Sundial: Using sunlight to reconstruct global timestamps*, Proceedings of European Conference on Wireless Sensor Networks (EWSN), Springer, 2009, pp. 183–198.
10. T. Hao, R. Zhou, G. Xing, and M. Mutka, *Wizsync: Exploiting wi-fi infrastructure for clock synchronization in wireless sensor networks*, Proceedings of The 32nd IEEE Real-Time Systems Symposium (RTSS), IEEE, 2011, pp. 149–158.
11. IEEE, *Ieee standard for a precision clock synchronization protocol for networked measurement and control systems*, IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002) (2008), 1–300.
12. Michael Kaufmann and Gerhard Klär, *A faster algorithm for edge-disjoint paths in planar graphs*, International Symposium on Algorithms, Springer, 1991, pp. 336–348.
13. Leslie Lamport and P Michael Melliar-Smith, *Synchronizing clocks in the presence of faults*, Journal of the ACM (JACM) **32** (1985), no. 1, 52–78.
14. Liqun Li, Guoliang Xing, Limin Sun, Wei Huangfu, Ruogu Zhou, and Hongsong Zhu, *Exploiting FM radio data system for adaptive clock calibration in sensor networks*, Proceedings of The 9th International Conference on Mobile Systems, Applications, and Services (MobiSys), ACM, 2011, pp. 169–182.
15. Yang Li, Rui Tan, and David KY Yau, *Natural timestamping using powerline electromagnetic radiation*, Proceedings of The 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), ACM, 2017, pp. 55–66.
16. Zhenjiang Li, Wenwei Chen, Cheng Li, Mo Li, Xiang-Yang Li, and Yunhao Liu, *Flight: Clock calibration using fluorescent lighting*, Proceedings of The 18th Annual International Conference on Mobile Computing and Networking (MobiCom), ACM, 2012, pp. 329–340.
17. Martin Lukac, Paul Davis, Robert Clayton, and Deborah Estrin, *Recovering temporal integrity with data driven time synchronization*, Proceedings of The 8th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), IEEE, 2009, pp. 61–72.
18. Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi, *The flooding time synchronization protocol*, Proceedings of The 2nd International Conference on Embedded Networked Sensor Systems (SenSys), ACM, 2004, pp. 39–49.
19. David L Mills, *Internet time synchronization: the network time protocol*, IEEE Trans. Commun. **39** (1991), no. 10, 1482–1493.

20. T. Mizrahi, *Security requirements of time protocols in packet switched networks*, 2014, `https://tools.ietf.org/html/rfc7384`.
21. Tal Mizrahi, *A game theoretic analysis of delay attacks against time synchronization protocols*, International Symposium on Precision Clock Synchronization for Measurement Control and Communication, IEEE, 2012, pp. 1–6.
22. Tyler Nighswander, Brent Ledvina, Jonathan Diamond, Robert Brumley, and David Brumley, *GPS software attacks*, Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS), ACM, 2012, pp. 450–461.
23. Behrooz Parhami, *Voting algorithms*, IEEE transactions on reliability **43** (1994), no. 4, 617–629.
24. Yehoshua Perl and Yossi Shiloach, *Finding two disjoint paths between two pairs of vertices in a graph*, J. Assoc. Comput. Mach **25** (1978), no. 1, 1–9.
25. Dima Rabadi, Rui Tan, David KY Yau, and Sreejaya Viswanathan, *Taming asymmetric network delays for clock synchronization using power grid voltage*, Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (AsiaCCS), ACM, 2017, pp. 874–886.
26. Anthony Rowe, Vikram Gupta, and Ragunathan Raj Rajkumar, *Low-power clock synchronization using electromagnetic energy radiating from ac power lines*, Proceedings of The 7th ACM Conference on Embedded Networked Sensor Systems (SenSys), ACM, 2009, pp. 211–224.
27. Igor R Shafarevich and Alexey Remizov, *Linear algebra and geometry*, Springer Science & Business Media, 2012.
28. Rui Tan, Linshan Jiang, Arvind Easwaran, et al., *Resilience bounds of sensing-based network clock synchronization*, Proceedings of the 24th International Conference on Parallel and Distributed Systems (ICPADS), IEEE, 2018, pp. 894–902.
29. Markus Ullmann and Matthias Vögeler, *Delay attacks – implication on ntp and ptp time synchronization*, Proceedings of International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, IEEE, 2009, pp. 1–6.
30. Sreejaya Viswanathan, Rui Tan, and David KY Yau, *Exploiting electrical grid for accurate and secure clock synchronization*, ACM Trans. Sensor Netw. **14** (2018), no. 2, 12.
31. Dorothea Wagner and Karsten Weihe, *A linear-time algorithm for edge-disjoint paths in planar graphs*, Combinatorica **15** (1995), no. 1, 135–150.
32. Douglas Brent West et al., *Introduction to graph theory*, vol. 2, Prentice hall Upper Saddle River, NJ, 1996.
33. Hassler Whitney, *A theorem on graphs*, Annals of Mathematics (1931), 378–390.
34. Zhenyu Yan, Yang Li, Rui Tan, and Jun Huang, *Application-layer clock synchronization for wearables using skin electric potentials induced by powerline radiation*, Proceedings of The 15th ACM Conference on Embedded Networked Sensor Systems (SenSys), ACM, 2017, pp. 1–14.