A Clean Slate Approach to Secure *Ad Hoc* Wireless Networking-Open Unsynchronized Networks

Jonathan Ponniah, *Member, IEEE*, Yih-Chun Hu, *Senior Member, IEEE*, and P. R. Kumar, *Fellow, IEEE*

***Abstract*—Distributed cyberphysical systems depend on secure wireless *ad hoc* networks to ensure that the sensors, controllers, and actuators (or nodes) in the system can reliably communicate. Such networks are difficult to design because, being inherently complex, they are vulnerable to attack. As a result, the current process of designing secure protocols for wireless *ad hoc* networks is effectively an arms race between discovering attacks and creating fixes. At no point in the process is it possible to make provable performance and security guarantees. This paper proposes a system-theoretic framework for the design of secure *open* wireless *ad hoc* networks, that provides precisely such guarantees. The nodes are initially unsynchronized, and join the network at any stage of the operation. The framework consists of a zero-sum game between all protocols and adversarial strategies, in which the protocol is announced before the adversarial strategy. Each choice of protocol and adversarial strategy results in a payoff. The design imperative is to choose the protocol that achieves the optimal payoff. We propose an "edge-tally supervised" merge protocol that is theoretically significant in three ways. First, the protocol achieves the max-min payoff; the highest possible payoff since the adversarial strategy always knows the protocol *a priori*. Second, the protocol actually does better and achieves the min-max payoff; it is a Nash equilibrium in the space of protocols and adversarial strategies. The adversarial nodes gain no advantage from knowing the protocol *a priori*. Third, the adversarial nodes are effectively limited to either jamming or conforming to the protocol; more complicated behaviors yield no strategic benefit.**

***Index Terms*—*Ad hoc* wireless networks, security.**

IN cyberphysical systems, sensors, actuators, and controllers interact through some communication medium; a wired network, or a shared wireless medium, or a combination of both. The content of this communication, the data, is generated by the outside physical world as interpreted by sensors and subsystems within the system itself. The data must be continuously and reliably transmitted between endpoints in order for the whole system to function as intended. Timely transmission and reception of this data is also essential to enable the autonomous agents to coordinate their activities. A key distinction in emphasis between cyberphysical systems and traditional embedded systems is that the former is more explicitly dependent on this ongoing communication, whereas the latter tends to be more defined by the computational requirements.

Secure cyberphysical systems retain their functionality even when individual subsystems fail or actively undermine the higher system. One adversarial tactic, for which there are no universal countermeasures, is the dissemination of false information to disrupt the operation of legitimate subsystems.

This paper considers the design of secure wireless *ad hoc* networks, an essential component of secure distributed cyberphysical systems. Wireless *ad hoc* networks consist of a collection of distributed "sources" and "destinations" which in this context, correspond to sensors or subsystems connected to the physical world. These sources and destinations (or nodes) attempt to exchange data, through a wireless medium, without the aid of a centralized controller; being distributed, the nodes themselves are responsible for determining their neighbors, learning the network topology [19], discovering routes [3], [17], [18], scheduling transmissions and receptions [8], [10], [13], selecting transmission power levels and modulation schemes [16], and verifying compliance [15]. The set of rules and instructions that allows these distributed nodes to participate in a functioning network is referred to as a *protocol*. A *legitimate* node, by definition, follows the protocol exactly whereas an *adversarial* node chooses to subvert the protocol in any way it deems fit. Wireless *ad hoc* networks are *secure* when they are able to continue to support communication between legitimate source-destination node pairs, even when the adversarial nodes behave arbitrarily. A comprehensive survey of game-theoretic approaches to network security is provided in [14].

The major challenge in designing secure wireless *ad hoc* networks, and secure complex systems in general, is that it is nearly impossible to anticipate all of the possible ways in which an adversarial node can undermine a protocol. For example, an adversarial node could choose to selectively drop a packet, advertise a wrong hop count, lie about its topology, cause routing loops in which messages are endlessly circulated, create artificial links known as wormholes, or blackholes into which messages are routed and disappear [20]. The SYBIL attack occurs when adversarial nodes adopt multiple forged identities to influence the network operation [5]. For some of these actions, there are specific types of defenses; temporal and geographical packet leashes [2] to counter wormholes, queue regulation at the access point to counter blackholes [1], routing chains to counter routing loops, encryption to counter control packet manipulation [7] and a link certificates issued by a trusted authority to counter the SYBIL atack. However, some vulnerabilities are structural in nature, and hence more difficult

Manuscript received September 19, 2015; revised April 5, 2016; accepted May 7, 2016. Date of publication May 24, 2016; date of current version March 16, 2017. This work was supported in party by NSF, AFOSR, and USARO under Contracts CNS-1302182 and CCF-0939370 under Contract FA-9550-13-1-0008 under Contract W911NF-15-1-0279. Recommended by Associate Editor P. Cheng.

J. Ponniah and P. R. Kumar are with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843-3259 USA (e-mail: jonathan_ponniah@tamu.edu; prk@tamu.edu).

Y.-C. Hu is with the Computer Science Laboratory and Electrical and Computer Engineering Department, University of Illinois, Urbana, IL 61801 USA (e-mail: yihchun@illinois.edu).

Digital Object Identifier 10.1109/TCNS.2016.2572398

to detect. For instance, the "rushing attack" [6] exploits a hidden vulnerability in dynamic source routing, whereby a node innocently forwards the first route-discovery packet it receives, allowing an adversarial node to manipulate the topological view of a legitimate node by "rushing" the transmission of selected route-discovery packets.

The design challenge is that one can never know whether or not there are other types of vulnerabilities that have yet to be detected. Instead, the overall design process effectively becomes an arms race between more sophisticated attacks and protocol "patches." At no point in the process is it possible to offer comprehensive security guarantees.

This paper continues the development of the game-theoretic, "clean-slate" framework, first proposed in [22] and [9], that provides precisely such guarantees for wireless *ad hoc* networks, contingent on some underlying model assumptions which describe the physical properties of the wireless channel, the computational capabilities of the nodes, and the properties of the local clocks at each node. In this framework, the security problem is modeled as a zero-sum game between the protocols and the adversarial strategies, where the payoff $J(p, q_p)$ for a specific choice of protocol p and adversarial strategy q_p, corresponds to the functionality "retained by the network;" the ability of the network to support data transfer between the legitimate-source destination pairs despite the actions of the adversarial nodes. The protocol p is selected before the adversarial strategy q_p which can be tailored specifically to p. Therefore, the best payoff that any protocol can achieve is the max-min payoff

$$\max_{\text{protocols } p} \min_{\text{attacker strategies } q_p} J(p, q_p). \tag{1}$$

The max-min payoff is optimal because the adversarial nodes, knowing the protocol *a priori*, can always choose the strategy that minimizes the payoff for a specific protocol. At best, a protocol can maximize its "worst case" payoff.

In [22], the proposed protocol is theoretically significant in three ways. First, the protocol described achieves the payoff (1) to within any $\epsilon > 0$. Second, it actually achieves the min-max payoff (again to within any $\epsilon > 0$)

$$\min_{\text{attacker strategies } q} \max_{\text{protocols } p} J(p, q). \tag{2}$$

Since the max-min payoff is generally less than the min-max payoff, the protocol is a saddle-point in the strategy space between protocols and adversarial actions; the adversarial nodes gain no advantage from knowing the protocol *a priori*.

Finally, the protocol achieves the min-max payoff over an adversarial strategy space confined to either jamming or conforming to the protocol at each time instant. Let $\mathcal{Q}$ denote the set of all such adversarial strategies. Then the protocol in [22] achieves to within any $\epsilon > 0$

$$\min_{\text{attacker strategies } q \in \mathcal{Q}} \max_{\text{protocols } p} J(p, q). \tag{3}$$

It is important to note that no protocol can prevent an adversarial node from jamming or conforming (the reasons for why an adversarial node may choose to conform to the protocol are not immediately obvious, but there are some situations described in [22] when such "cooperation" more effectively reduces the payoff than jamming).

As a result, this framework captures the full spectrum of adversarial activity that could be deployed to counter any protocol, for the specific payoff function $J(p, q_p)$ in [22], and a specific set of model assumptions. To improve this framework, this paper will consider more general model assumptions than those in [22] that introduce new kinds of adversarial behavior.

The assumptions in [22] require that the legitimate and adversarial nodes turn on simultaneously, thus eliminating, *a priori*, any scenarios in which networks must adjust their operation to accommodate nodes born later during the operating lifetime. Networks that cannot make this adjustment are by definition, closed. However, in many applications of wireless *ad hoc* networks, the legitimate nodes are not born simultaneously; they may appear at any time, for instance when they enter the same vicinity as other nodes in the case of autonomous automotive systems, or when they turn on after a prolonged period of hibernation in the case of low-energy sensor systems.

This paper considers open networks which, by definition, receive nodes born at any stage of the operating lifetime. Allowing unbounded birth-times, however, creates new possibilities for the adversarial nodes to exploit. In order to account for this activity, the protocol will be upgraded accordingly so that the results in [22], namely the payoffs defined in (1)–(3) are still achievable.

This process, in which the model assumptions are modified to more accurately capture the dynamics of *ad hoc* networks, is a more systematic and sound approach to secure protocol design than an arms race between attacks and protocol fixes.

The rest of the paper is organized as follows: Section I describes the challenges of operating open networks and the additional protocol features needed to address them, Section II defines the model assumptions, Section III defines the game between protocols and adversarial strategies and the corresponding payoff function, Section IV presents the main results, Section V describes the protocol, Section VI proves the main results, and finally Section VII concludes with possible directions for future work.

I. The Challenges of Open Networks

There are several fundamental challenges when confronting the security issues faced by open networks. The first challenge is in detecting the newcomers as they appear. Networks operate according to a common set of agreed upon rules, conditioned on the network topology, that stipulate when and to whom an individual node should transmit, and when and to whom the same node should listen. The set of rules particular to a specific network is referred to as a schedule.

The legitimate nodes, by assumption are half-duplex; they cannot transmit and receive messages simultaneously. The legitimate nodes are also distributed; what is heard or known by one node is not necessarily heard or known by the others. In addition, the legitimate nodes are unsynchronized and their local clocks do not tick at the same rate. An external node, party to an external network, is subject to its own schedule and reference clock and completely ignorant of the schedules and reference clocks that define the networks in which it does not participate. Therefore, without any other mechanism, an adjacent external node may never be detected.

To guarantee detection, the protocol must first set aside a periodic interval of time (to be referred to as the "recurrent neighbor discovery phase") in which all nodes broadcast probe packets to advertise the existence of the network. Conversely, the protocol must also set aside a periodic interval of time (to be referred to as the "sentinel phase") in which all nodes cease transmitting so that the broadcasts of an external node can be detected by an adjacent neighbor in the network. This activity will be referred to as sentinel activity, and the corresponding nodes as sentinels. Finally, the protocol must set aside another periodic interval of time in which the sentinels, upon completing this activity, can inform the other nodes if they detected any external neighbors (to be referred to as the "recurrent network discovery phase").

The second challenge is in merging two subnetworks after one has detected the other. Since the legitimate nodes are half-duplex, distributed, and unsynchronized, a network does not (necessarily) know whether the probe packets it broadcasts were actually received or the identities of the external nodes that may or may not have received them. Furthermore, the network does not know whether the external nodes it detects, are in turn aware of the fact they have been detected or that the network (that has detected them) even exists. Therefore the protocol must set aside a periodic interval of time (to occur during the recurrent neighbor and network discovery phases) in which the network can create mutually authenticated links with all the external nodes that have detected it. Conversely, an external node must in turn receive enough information in a probe packet to align its schedule with the network from which the probe packet originated.

The third challenge is in controlling the overhead of these processes when they are subverted or attacked by the adversarial nodes. Due to the first two challenges, there is necessarily a delay from when a node in network A detects an external neighbor, until the moment the other nodes in A are aware that an external node has been detected. There is also a delay from this moment until the external node reaches a periodic interval in which it can join other networks (the recurrent neighbor discovery phase).

During this delay, the external node, unaware that it is the target of a merge attempt, may independently detect and align its schedule with another network B, so that its schedule is no longer aligned with A. The attempt by A to merge with the external node then fails. This challenge introduces a fundamental ambiguity in which A is unable to determine after the fact, whether or not the external node was malicious, or was innocently absorbed by a third party. This ambiguity can be exploited by an adversarial node to repeatedly lure networks into making spurious merge attempts. Since each merge attempt is expensive with respect to the expended overhead (the aforementioned delay during which no throughput is transferred and no other merges are attempted), this adversarial strategy can significantly reduce the network functionality. Therefore an additional mechanism is required that removes the ambiguity from the merge process, and ensures that the overhead is an arbitrarily small fraction of the operating lifetime.

The final challenge, more technical in nature, is encountered when operating in unbounded infinite time; namely, the size of any packet bearing a timestamp grows to infinity. Since the recurrent neighbor discovery and sentinel phases, which occur periodically, must necessarily be of bounded length in their first iterations, the timestamps and the packets that bear them, must always be confined to a bounded size. In unbounded time, this means that the clocks must periodically reset themselves. However to avoid the ambiguity and consequent vulnerability (e.g., to a replay attack) that arises from reusing old timestamps, the network must also change encryption after each clock reset. Assuming only a finite number of such changes can be supported, the operating lifetime must be bounded and the network must limit encryption changes to occasions in which new nodes join. During the intervals in which the operating lifetime has expired and no new nodes have emerged, the network must freeze its clock and activity. These intervals will be referred to as the "coma phase."

To summarize, allowing the birth times of the legitimate nodes to be unbounded introduces unique challenges that require careful treatment.

II. The Model

There are n wireless nodes which model the nodes in a cyberphysical system. These nodes generate data which must be shared, not necessarily with all other nodes, but between specific pairs of nodes. The legitimate nodes, by definition, follow the communication protocol exactly. The adversarial nodes, on the other hand, behave arbitrarily so as to undermine the exchange of data. The legitimate nodes are half-duplex; they cannot simultaneously transmit and receive. In addition, the legitimate nodes do not know which nodes are legitimate or adversarial *a priori*. However, the adversarial nodes not only know which nodes are adversarial, but can also communicate with these nodes via backchannels of infinite bandwidth. All nodes are subject to a power constraint.

A legitimate *modulation scheme* $m \in \mathcal{M}_i$ for node i is a mode of transmission that specifies a joint selection of transmit power level, modulation symbol, symbol rate, and encoding scheme by node i. In other words, the modulation scheme m describes the physical action of node i in the wireless medium. The effective intended transmission rate of a legitimate modulation scheme is denoted by $r(m)$. The actual transmission rate of any modulation scheme may be zero if the intended destination does not receive the transmitted message. An adversarial node has the option of using the legitimate modulation schemes, but can also choose to jam by emitting random noise at the maximum power level.

A *concurrent transmission vector* $c := \{m_1, \ldots, m_{n(n-1)}\}$, is a vector of $n(n-1)$ modulation schemes, where the index of each element corresponds to one of the $n(n-1)$ possible one-hop source destination pairs, and the element itself corresponds to the modulation scheme used by the source. Let $\mathcal{C}$ denote the set of possible concurrent transmission vectors.

A concurrent transmission vector is *feasible* by definition if the message transmitted by each source is received at the corresponding destination. In a network composed entirely of legitimate nodes, the set of feasible concurrent transmission vectors is determined exclusively by the physical channel. The addition of adversarial nodes augments the feasible set; some of the concurrent transmission vectors that would otherwise be non-feasible can be made artificially feasible if the adversarial nodes use the backchannel of infinite bandwidth to exchange messages while simulating the modulation schemes in the wireless channel. Let $\mathcal{F}$ denote the augmented feasible set.

A feasible concurrent transmission vector is *disabled* if a destination fails to receive a message due to the action of any adversarial node. The feasible concurrent transmission vectors that can be disabled fall into one of two categories. The first consists of vectors that can be *directly* disabled; adversarial nodes refuse to comply with the modulation scheme prescribed by the vector, electing to jam instead, and affecting the wireless medium in a way that makes the reception of at least one legitimate message impossible. An artificially feasible concurrent transmission vector can also be directly disabled if the adversarial nodes choose not to forward legitimate messages through the backchannel. Let $\mathcal{D}$ denote the set of concurrent transmission vectors that can be directly disabled.

The second category consists of vectors that can be *indirectly* disabled. A concurrent transmission vector, to be active, requires a prior level of coordination in the network so that each source-destination pair in the vector can simultaneously select the prescribed modulation scheme. In a network with adversarial nodes, the process by which this prior level of coordination is reached can also be attacked, particularly if the process is poorly or naively constructed. A concurrent transmission vector is indirectly disabled if the adversarial nodes prevent its execution by attacking this process. Since the success of the attack depends on the robustness of the process itself, the set of concurrent transmission vectors that can be indirectly disabled is protocol dependent. By contrast, the set of vectors that can be directly disabled is independent of the protocol that establishes the prior level of coordination.

The results in this paper are contingent on the following four model assumptions.

(**M1**) The set of legitimate modulation schemes (and concurrent transmission vectors $\mathcal{C}$ by implication) is finite. This assumption is true in practice; all communication systems operate on the basis of a finite number of modulation schemes.

(**M2**) The legitimate nodes are connected. More precisely, there exists a graph where every edge in the graph corresponds to a modulation scheme and the legitimate nodes in the graph are connected. Furthermore, every edge in the graph belongs to a concurrent transmission vector c in which the legitimate source-destination pairs in c are assigned non-zero rate and c cannot be directly disabled. That is, $c \in \mathcal{F} \setminus \mathcal{D}$. This assumption is necessary for any kind of security guarantees; if the legitimate nodes are not connected, then an adversarial node can always disconnect a legitimate node from the network.

(**M3**) An encryption scheme assigns each node a unique private key with which to encrypt transmitted messages. The encrypted message can be deciphered with a corresponding public key, but cannot be altered or forged without the private key. A different encryption scheme is assigned to every possible subnetwork; each node uses the unique private key assigned to it by the encryption scheme of the corresponding subnetwork. In addition, each node also has an identity certificate from a trusted authority that verifies the node's identity. These assumptions, though computationally expensive, are quite feasible to implement with existing technology.

(**M4**) Each legitimate node has a local clock where $\tau^i(t)$ denotes the local clock at node i with respect to some global reference clock t. The local clocks are relatively affine. Let $\tau^i_j(t)$ denote the local clock of node i with respect to t, the local clock at node j. Then $\tau^i_j(t) := a_{ij}t + b_{ij}$, where a_{ij} and b_{ij} are constants that denote the relative skew and offset respectively of node i with respect to node j. The relative skew is bounded so that $0 < 1/a_{\max} \leq a_{ij} \leq a_{\max}$. Finally, let $d^{(j)}_{ij}$ denote the one-way message delay between i and j as measured by the local clock at node j. The message delay satisfies the following constraint $d^{(j)}_{ij} \leq d_{\max}$. The legitimate nodes know $a_{\max}$ and $d_{\max}$ *a priori*, but not the relative skews, offsets, and one-way delays. This assumption, or variants of it, is standard in the clock synchronization literature [4], [11].

III. Game Between Protocols and Adversarial Strategies

A protocol p, created with the intent of enabling the nodes to form a functioning network, is given to each node. The adversarial nodes then decide on a strategy q_p to subvert p. At time $t = 0$, with respect to the global reference clock, the first legitimate node turns on. Subsequently, at arbitrary unbounded times, other nodes, legitimate or adversarial, turn on and begin to execute p and q_p respectively. Some legitimate or adversarial nodes may never turn on.

Let $\mathcal{D}_{p,q}(t)$ denote the set of concurrent transmission vectors that have been disabled (directly or indirectly) by the adversarial nodes at time t. Let $G(t)$ denote the graph that satisfies the following three conditions at time t: first, all legitimate nodes are included in $G(t)$; second, every edge in $G(t)$ corresponds to a non-zero rate modulation scheme in some non-disabled concurrent transmission vector c (that is, $c \in \mathcal{F} \setminus \mathcal{D}_{p,q}(t)$); finally, every edge in $G(t)$ belongs to a connected component that includes at least one legitimate node. Thus the graph $G(t)$ includes all legitimate nodes and those adversarial nodes that conceal their adversarial identity by conforming with the protocol (in the sense of enabling sufficiently many concurrent transmission vectors to remain in the connected component of at least one legitimate node).

Now let t_l denote the unbounded birth time of the last legitimate node with respect to the global reference clock, and let T denote the fixed predetermined length of the operating lifetime. Consider the execution of p and q_p during the interval $[t_l, t_l + T)$. Since there are a finite number of concurrent transmission vectors (assumption M1), by implication there are a finite number of distinct graphs $\{G_i, i = 1, \ldots, N\}$ that appear during the interval $[t_l, t_l + T)$, where $G_i := G_{t_i}$ for some $t_i \in [t_l, t_l + T)$. Let α_i denote the fraction of the interval $[t_l, t_l + T)$ in which the graph G_i is active. Let x_{G_i} denote the effective throughput accrued to all source-destination pairs in the graph G_i. For any utility function $U(x)$ of throughput vector x, continuous in the components of x, the payoff to the network $J(p, q_p)$ for the choice of protocol p and adversarial strategy q_p is

$$J(p, q_p) := \sum_{i=1}^{N} \alpha_i U(x_{G_i}). \tag{4}$$

The payoff in (4) also appears in [22], where the birth times are within a known bound. The advantage of bounded birth times is that the operating lifetime can be made large enough to amortize the throughput loss during the time period in which some legitimate nodes have yet to be born. In this paper, the birth times are unbounded, which implies that an operating lifetime of fixed size can never be made large enough to amortize this throughput loss. In order for the payoff to be meaningful (nonzero), the operating lifetime over which the payoff is evaluated must begin from the birth of the last legitimate node.

If the utility $U(x)$ is the benefit or functionality of a network supporting a throughput x through the protocol p, the payoff in (4) is one way of defining the functionality retained by this network when the adversarial nodes choose strategy q_p. The most suitable such definition is still an open question; the notion of the network functionality is itself ambiguous if the adversarial nodes strategically cooperate with the protocol. We will leave this question for future work.

IV. Main Results

The main contribution of this paper is an "edge-tally supervised" merge protocol (hereafter referred to simply as the merge protocol) that achieves the payoff in (3). Let $\mathcal{Q}$ denote the set of adversarial strategies in which the adversarial nodes are confined to directly disabling concurrent transmission vectors at each time instant. That is, $\mathcal{D}_{p,q}(t) \subset \mathcal{D}$ for all p and $q_p \in \mathcal{Q}$. We have the following theorem.

Theorem IV.1: For any $\epsilon > 0$, the merge protocol achieves the payoff $(1 - \epsilon) \min_{\text{attacks } q \in \mathcal{Q}} \max_{\text{protocols } p} J(p, q)$.

The proof is constructive. This section describes three theorems that together imply Theorem IV.1. Section V describes the protocol and summarizes the proof of the first theorem (which appears in [22]). The other two theorems are proved in Section VI. To state the theorems, some definitions are needed.

Let G denote the connected graph of legitimate nodes in model assumption (M2). A pair of connected subcomponents of G are *adjacent*, by definition, if they are connected by an edge in G. A node is *unborn* if it has yet to turn on, *active*, if it has turned on and its local clock has not expired, and *dormant* its local clock has expired. During the execution of the merge protocol, each active legitimate node l belongs to some subnetwork S (which may consist of l alone) and a *maximal* active legitimate subcomponent $G_l(S)$; every edge in $G_l(S)$ is in S and G, and any other legitimate node $l' \in S$ is either also in $G_l(S)$ or part of a disjoint subcomponent $G_{l'}(S)$ where $G_l(S)$ and $G_{l'}(S)$ are connected by adversarial nodes in S. Two independent subnetworks are *adjacent* by definition, if some pair of their respective maximal legitimate subcomponents are adjacent.

The first theorem states that the legitimate nodes in a connected subcomponent of G, with birth times within a known bound, form a subnetwork that achieves a "locally" optimal payoff if the throughput loss ϵ from the protocol overhead is small. The payoff is locally optimal in the sense that (3) is limited to the nodes in the subnetwork.

Theorem IV.2: For a sufficiently small $\epsilon > 0$ the merge protocol enables each subnetwork to achieve the payoff $(1 - \epsilon) \min_{\text{attacks } q \in \mathcal{Q}} \max_{\text{protocols } p} J(p, q)$ on the local topology.

An *undesirable state* in the execution of the merge protocol, is one in which there are independently operating adjacent subnetworks; the network is "artificially" segregated by the relative birth times of the legitimate nodes and not by any fundamental property of the physical channel. A *desirable state* is one in which there are no adjacent subnetworks. The next theorem states that any execution of the merge protocol converges to a desirable state within a bounded time T' of the birth of the last legitimate node t_l.

Theorem IV.3: Every execution of the merge protocol reaches a desirable state by $t_l + T'$.

The final theorem shows that the throughput loss due to the protocol overhead can be made arbitrarily small.

Theorem IV.4: The throughput loss ϵ is arbitrarily small.

Theorem IV.1 follows from Theorems IV.2, IV.3, and IV.4.

V. The Protocol

The edge-tally supervised merge protocol serves two functions: first, it enables each legitimate node to form a rudimentary subnetwork that converges to a "locally" optimal throughput (Theorem IV.2); second, it forces adjacent subnetworks to detect each other and merge (Theorem IV.3), so that the throughput to which any subnetwork converges is optimal for the global topology (Theorem IV.4). The protocol phases that serve the first function are the initial neighbor discovery phase, the initial network discovery phase, the scheduling phase, the data transfer phase, and the verification phase. These phases appear in [22] and will be summarized in Section V-A. The second protocol function is served by the recurrent neighbor discovery phase, the recurrent network discovery phase, the sentinel phase, and the coma phase. These phases will be examined more closely in Section V-B. An example that demonstrates the protocol operation in closed synchronized networks is provided in Section V-C. Fig. 1 shows how each of these phases interact.

A. Locally Optimal Operation

Consider an active legitimate node A immediately after birth. In the *initial neighbor discovery phase*, node A broadcasts probe packets that are received by any adjacent nodes born within a known bound of node A.

Lemma V.1: Every pair of adjacent legitimate nodes whose birth times are within a known bound, complete a handshake.

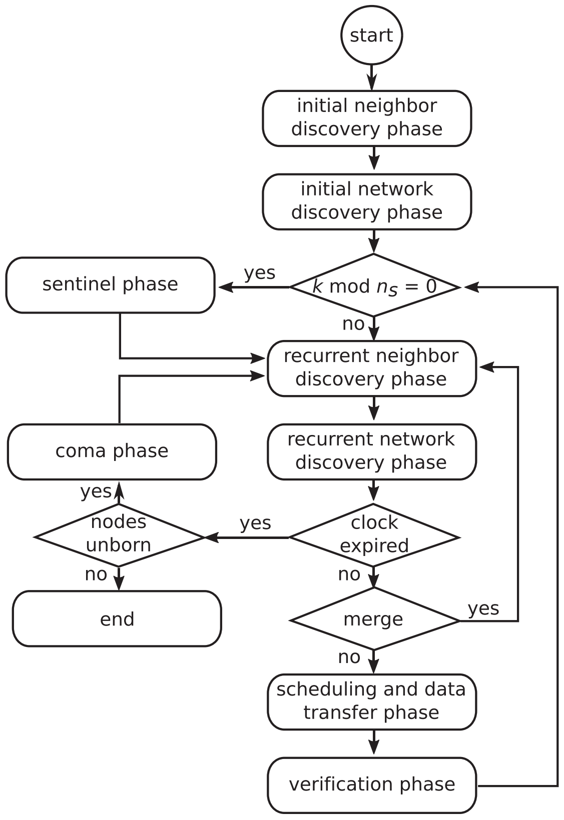

start
initial neighbor discovery phase
initial network discovery phase
sentinel phase
yes
k mod $n_S = 0$
no
recurrent neighbor discovery phase
recurrent network discovery phase
coma phase
yes
nodes unborn
yes
clock expired
no
no
end
merge
yes
no
scheduling and data transfer phase
verification phase

Fig. 1. Protocol phase diagram shows the interaction between the protocol phases. The initial neighbor and network discovery phases, the scheduling and data transfer phases, and the verification phase drive each subnetwork to a "locally" optimal payoff whereas the recurrent neighbor and network discovery phases, the coma phase and the sentinel phase enable adjacent subnetworks to merge.

Proof: The orthogonal MAC code [21] guarantees that any pair of adjacent nodes can exchange a message within a bounded time despite the half-duplex constraints, provided their relative skews and birth times are bounded. □

With each neighbor, node A completes a series of handshakes and exchanges a pair of timing packets. Both nodes measure their relative skews, and relative offsets to within $d_{\max}$ from the time-stamps generated during the exchange, and include these packets, the probe packets and the acknowledgements in a mutually authenticated link certificate.

In the *initial network discovery phase*, node A and the other legitimate nodes use the Byzantine General's algorithm [12] to share their list of neighbors and link certificates.

Lemma V.2: The legitimate nodes in a connected subcomponent of G with birth times within a known bound, obtain a common topological view.

Proof: The Byzantine General's algorithm guarantees that a connected, synchronized network of legitimate nodes with encrypted communication will reach consensus, even if the adversarial nodes provide false information. The legitimate nodes are not initially synchronized, but [22] describes how to attain a sufficient level of coordination when the relative clocks are affine (M4) and the birth times are within a known bound, to execute the algorithm. □

Given a common topological view, each legitimate node estimates the relative skew and offset of the local clock of a designated node, using the relative skews and offsets measured between adjacent nodes in the initial neighbor discovery phase.

Lemma V.3: The legitimate nodes obtain consistent estimates of the designated reference clock.

Proof: The paths connecting each legitimate node to the designated node may include adversarial nodes. The relative clock parameters estimated along one path may differ from those estimated along another path if the adversarial nodes generate false time-stamps during the initial neighbor discovery phase. The consistency check algorithm described in [22] guarantees the legitimate nodes will identify the adversarial path if the error in the relative clock skew exceeds a specified error. □

It follows from Lemmas V.2 and V.3 that the legitimate nodes in the resulting subnetwork, denoted by S_A, share a common topological view and consistent estimates of a common reference clock. Hence, S_A is effectively synchronized.

The subnetwork S_A enters the scheduling phase, and derives a utility-optimal schedule based on an estimate of the feasible non-disabled concurrent transmission vectors $\mathcal{F}^{(1)}$, where $\mathcal{F}^{(1)} := \mathcal{C}$ (the initial estimate is the entire set of concurrent transmission vectors $\mathcal{C}$).

Lemma V.4: The legitimate nodes derive the same schedule.

Proof: Since the legitimate nodes share a common topological view, and know $\mathcal{C}$ *a priori*, they can independently determine the schedule that maximizes the utility function $U(x)$. □

This schedule is executed in the data transfer phase. In the verification phase, the legitimate nodes in S_A, using the Byzantine General's algorithm, attempt to infer the set of concurrent transmission vectors $\mathcal{D}^{(1)}$ that failed to deliver all scheduled packets.

Lemma V.5: The legitimate nodes decide on the same $\mathcal{D}^{(1)}$.

Proof: Each legitimate destination knows the packets it is scheduled to receive and recognizes when these packets fail to arrive. The Byzantine General's algorithm guarantees that the legitimate nodes, being connected, synchronized and able to communicate securely via encryption, will reach a consensus on the scheduled packets missed by each legitimate destination. The legitimate nodes also know the schedule of concurrent transmission vectors that failed to deliver the missing packets, so each legitimate node can independently infer which concurrent transmission vector was responsible for each missing packet. Therefore, the legitimate nodes will decide on the same set $\mathcal{D}^{(1)}$. □

These disabled or non-feasible concurrent transmission vectors are pruned from the estimated feasible set so that $\mathcal{F}^{(2)} := \mathcal{F}^{(1)} \setminus \mathcal{D}^{(1)}$.

At this stage, S_A enters the protocol phases that enable the subnetwork to detect and merge with any adjacent subnetworks: the sentinel phase, the recurrent neighbor discovery phase, and the recurrent network discovery phase. These phases will be discussed in Section V-B.

Suppose S_A does not acquire any new nodes or detect any adjacent subnetworks. Then S_A completes another iteration of the scheduling, data transfer, and verification phases. After the kth iteration, the estimate of feasible concurrent transmission vectors is whittled down or pruned according to the rule $\mathcal{F}^{(k+1)} := \mathcal{F}^{(k)} \setminus \mathcal{D}^{(k)}$. Now Theorem IV.2 can be proved.

Proof of Theorem IV.2: Since there are a finite number of possible concurrent transmission vectors (M1), the estimate $\mathcal{F}^{(k)}$ will be correct for an arbitrarily large fraction of the total number of iterations, provided the number of iterations is sufficiently large. If the protocol overhead can be made arbitrarily small (Theorem IV.4), it follows that S_A operates at an effective throughput that is payoff-optimal for the "local" topology (the nodes in S_A).

B. Globally Optimal Operation

The protocol phases in the previous section force every subnetwork to transition to a state in which its operation is optimal with respect to its particular "local" topology. This section describes the protocol phases that drive the entire network to a state in which there are no adjacent subnetworks; the operation of a particular subnetwork is optimal with respect to the topology of the entire network.

There is a fundamental delay from the moment a node in subnetwork detects an external node until the moment the subnetwork is ready to merge with the node that was detected. Moreover, this delay can prevent a successful merge between an adjacent pair of legitimate subnetworks since the target subnetwork can be absorbed by a third party before the merge attempt is complete. Hence, the cause of every failed merge attempt is ambiguous. The edge-tally supervised merge protocol assigns a recursively determined bound on the number of merge attempts that can be instigated by an edge that connects two adjacent subnetworks. By maintaining a "tally" of failed merge attempts for each edge, we can show that the set of adversarial actions is finite and can be whittled down after sufficiently many protocol iterations. The protocol phases will now be examined more closely.

1) The Sentinel Phase: The sentinel phase is a periodic interval in which all nodes in a subnetwork cease transmission and listen for any probe packets broadcast by external subnetworks. This activity is necessary because the external subnetworks operate according to a different schedule known only to the nodes within that subnetwork. A *sentinel* is a node in the sentinel phase.

Lemma V.6: The sentinels are guaranteed to detect any adjacent subnetworks.

Proof: A *protocol iteration* consists of the recurrent neighbor discovery phase, the recurrent network discovery phase, the scheduling phase, the data transfer phase, and the verification phase. During the recurrent neighbor discovery phase (to be described next), the nodes in the subnetwork broadcast probe packets. The sentinel phase, which runs the duration of two protocol iterations (stretched by a factor of $a_{\max}$), is guaranteed to fully overlap with the recurrent neighbor discovery phase of the external subnetwork, regardless of the relative skew between the reference clocks of both subnetworks. □

Let n_s denote the number of protocol iterations between successive sentinel phases; the kth protocol iteration is substituted for the sentinel phase if $k \bmod n_s = 0$. The medium-term throughput loss ϵ_m is the loss incurred when the protocol periodically detours into the sentinel phase.

Lemma V.7: The loss ϵ_m can be made arbitrarily small.

Proof: The sentinel phase lasts for $2a_{\max}$ protocol iterations. For any desired ϵ_m choose n_s so that $\epsilon_m < 2\lceil a_{\max}\rceil / n_s$. □

Upon completing the sentinel phase, the subnetwork proceeds to the recurrent neighbor discovery phase.

2) The Recurrent Neighbor Discovery Phase: The recurrent neighbor discovery phase is a periodic interval in which the nodes in a subnetwork broadcast probe packets to any external nodes.

Lemma V.8: The topology and reference clock of a subnetwork can be inferred from a received pair of probe packets.

Proof: The probe packets are timestamped according to the estimate of the reference clock of the subnetwork (which by Lemma V.3 is accurate to within a desired error). Hence, the external node can measure the relative skew and offset to within $d_{\max}$ from the probe packet pair. Each probe packet includes the identity certificate of the broadcasting node, the link certificates of the subnetwork, and the packets received by the broadcasting node during the Byzantine General's consensus algorithm that prove a consensus was reached; sufficient information to show the nodes in the subnetwork share a common topological view. Hence an external node can verify the subnetwork exists from the probe packet pair. □

Each node creates a mutually authenticated link certificate with any external nodes that respond to the probe packets, using the same steps described in the initial neighbor discovery phase in Section V-A. Upon completing the recurrent neighbor discovery phase, the subnetwork proceeds to the recurrent network discovery phase.

3) The Recurrent Network Discovery Phase: The recurrent network discovery phase is an interval in which the legitimate nodes attempt to arrive at a consensus on the link certificates generated during the recurrent neighbor discovery phase and the probe packets received during the preceding sentinel phase; the information necessary for the subnetwork to expand and operate in a coordinated manner. The legitimate nodes use the Byzantine General's algorithm to share the link certificates and probe packets received from each individual node during the preceding phases. Hence, each legitimate node independently infers the same view of the network topology and the external subnetworks (Lemma V.2). The consistency check described in Section V-A ensures the legitimate nodes obtain a reliable estimate of a common reference clock (Lemma V.3).

The legitimate nodes follow a set of merge rules to determine whether the topology has expanded to include new nodes, whether external subnetworks are present in the vicinity, whether the number of previous failed attempts (if any) to merge with these external subnetworks exceeds a predetermined bound, whether there are any external subnetworks for which this bound is not exceeded, and whether there is sufficient time in the operating lifetime to make any further merge attempts. Based on this information, the subnetwork will either return to the scheduling phase, or attempt another merge, or freeze operation and enter the coma phase.

Before stating the merge rules, it is necessary to clarify some terminology. A subnetwork S *detects* S' if legitimate nodes in the former receive probe packets from legitimate nodes in the latter; the legitimate nodes in S are able to accurately predict when S', barring any changes to its schedule, will next enter the recurrent neighbor discovery phase (Lemma V.8). Moreover, S *attempts to merge* with S' if the legitimate nodes in the former change their schedule and reference clock to align the next recurrent neighbor discovery phase of S with that of S', based on the timestamps of the received probe packets. Finally, attempted merge is *successful* if the recurrent neighbor discovery phases of both subnetworks align; S' does not change its schedule to merge with some other subnetwork S'' in the interim. The outcome of a successful merge is a new subnetwork that includes the pair of adjacent maximal legitimate subcomponents from the original subnetworks. Since a legitimate node does not know which nodes (apart from itself) are legitimate, a merge attempt is a *failure* if the new subnetwork includes none of the nodes from S'.

Every possible subnetwork that can be formed from a collection of n nodes is assigned, *a priori*, an encryption scheme [see model assumption (M3)], and an *index* where the assigned index is greater than the number of nodes in the subnetwork. Let i_S denote the index of subnetwork S.

(**MR1**) Let n_{AB} denote the number of previous failed attempts by S_A to merge with an external subnetwork S_B, from probe packets transmitted and received by node B and A respectively. The subnetwork S_B is designated *eligible* for a merge attempt by S_A if $i_{S_A} > i_{S_B}$, and $n_{AB} \leq (i_{S_B}!)^3$.

(**MR2**) Each legitimate node in S_A increments the counter n_{AB} by one if S_A made a failed attempt to merge with S_B during the previous recurrent neighbor discovery phase.

(**MR3**) If the topology of S_A includes a new edge, each legitimate node resets the reference clock and switches to the encryption scheme assigned to S_A.

(**MR4**) If the reference clock of S_A reaches the specified threshold time T_r that signals the clock is about to expire, each legitimate node in S_A freezes its local clock and enters the coma phase.

Lemma V.9: The index of a subnetwork, once discarded, is never reused.

Proof: The legitimate nodes in a subnetwork reset the reference clock and switch encryption keys only if the topology includes new edges (MR3). The loss of a node does not trigger a clock reset and change in encryption. □

Lemma V.10: The maximum number of merge attempts made by subnetwork S_i of index i is fewer than $(i!)^3$

Proof: Given any $k < i$, subnetwork S_i makes at most $(k!)^3$ attempts to merge with S_k for each edge that connects S_i and S_k (MR1). There are at most ik such edges; by construction the index of a subnetwork is larger than the number of nodes in the subnetwork. Therefore S_i makes at most $ik(k!)^3$ attempts to merge with S_k. The total number of merge attempts made by S_i is $\sum_{1\leq k<i} ik(k!)^3 < i^2(i-1)((i-1)!)^3 < (i!)^3$. □

If the reference clock does not trigger the coma phase, and there are eligible external subnetworks, then S_A designates as the target of a merge, the eligible subnetwork with the lowest index. If there are no eligible external subnetworks, then S_A proceeds to the scheduling phase.

4) The Coma Phase: The coma phase, is an interval in which a legitimate node freezes its local clock, ceases all transmission, and listens for any probe packets transmitted by an external node. The node remains in this phase until such probe packets are received.

Suppose node A, in the coma phase, receives probe packets from node B in subnetwork S_B. By definition, S_B is an eligible target if the following two conditions are satisfied. First, A has never previously been in a subnetwork that included S_B and the edge between node A and B. Second, $n_{AB} \leq (i_{S_B}!)^3$ where i_{S_B} is the index of subnetwork S_B. If both conditions are satisfied, node A unfreezes its clock, and attempts to merge with S_B. Otherwise node A remains in the coma phase until detecting a subnetwork that satisfies both conditions.

Lemma V.11: The clock freeze ensures a legitimate node can successively attempt $(i!)^3$ merges with S_i for every edge between the node and S_i for all $1 \leq i \leq i_{\max}$.

Proof: Let T_I denote the length of a protocol iteration, and $i_{\max}$ the largest index that can be assigned to a subnetwork. The total runtime allowed by the reference clock is T, the operating lifetime. Each merge attempt runs the clock for $2a_{\max}T_I$; the maximum wait time until the next recurrent neighbor discovery phase of the targeted subnetwork, plus an additional iteration to reach the recurrent network discovery phase in which the merge is completed (all stretched by a factor of $a_{\max}$). Choose the threshold time T_r in (MR4) so that $T - T_r > ((i_{\max}+1)!)^3 2a_{\max}T_I$. The lemma follows since $((i_{\max}+1)!)^3 > \sum_{j\leq i_{\max}} j(j!)^3$. □

5) Steady State: The subnetwork S_A repeatedly iterates through the recurrent neighbor discovery phase, recurrent network discovery phase, scheduling phase, data transfer phase, and verification phase. At lower periodicity S_A enters the sentinel phase to detect adjacent external subnetworks. An eligible subnetwork once detected, becomes the target of a merge attempt by S_A. A merge attempt can fail for benign reasons or because of adversarial nodes within S_A or the external subnetwork. In addition, the reference clock may trigger the coma phase before S_A can complete a merge with a legitimate neighbor. However, some merges are also guaranteed to succeed because the legitimate nodes are connected [model assumption (M2)]. Each successful merge yields a new subnetwork that independently converges to a locally optimal throughput vector.

In unbounded time, a scenario unfolds in which nodes are born, organize into subnetworks, independently converge to the utility-optimal throughput, detect other subnetworks and merge, converge again to the utility-optimal throughput, disband (possibly) in the coma phase, but reconstitute and merge again. In the next section, we prove that the overall operation yields a min-max optimal payoff.

C. The Protocol in a Closed Synchronized Network

The following example will illustrate the protocol operation in a simple closed synchronized network of three nodes A, B, and C. The key idea behind the operation is the iterative pruning of disabled/non-feasible concurrent transmission vectors from a finite set. A similar approach is used in the more general setting of open unsynchronized networks to enable the merging of adjacent subnetworks.

In this example, nodes A and B are legitimate while C is adversarial. There are two rates at which the nodes can communicate: 1 bit/s and 2 bit/s. If C cooperates (refrains from jamming), any pair of nodes can communicate at 2 bit/s. With C jamming, Nodes A and B can communicate at 1 bit/s. Let (x_{AB}, x_{BC}, x_{AC}) be a concurrent transmission vector where x_{AB}, x_{BC}, and x_{AC} denote the link rates on the links AB, BC, and AC respectively. The set of possible (but not necessarily feasible) concurrent transmission vectors $\mathcal{C}$ is the set of all triples $\bar{x} := (x_1, x_2, x_3)$ where $x_i \in \{0, 1, 2\}$ for all $i = 1, 2, 3$. In this example, the utility is defined as $U(\bar{x}) := \min_i x_i$. For example, if $\bar{x} = (2, 1, 2)$ then $U(\bar{x}) = 1$.

Define a particular adversarial strategy q^* in which node C chooses to cooperate with the protocol at 1 bit/s for the first quarter of the operating lifetime, and jam for the other three quarters. This strategy creates two distinct network graphs over the operating lifetime: $G_1 = \{AB, BC, AC\}$ in which all nodes participate and $G_2 = \{AB\}$ in which node C excludes itself. By definition $\bar{x}_{G_1} = (x_{AB}, x_{BC}, x_{AC})$ and $\bar{x}_{G_2} = x_{AB}$. The graphs G_1 and G_2 are the largest connected components that include all legitimate nodes [see model assumption (M2)]. The fraction of the operating lifetime in which each graph is active is $\alpha_1 = 0.25$ and $\alpha_2 = 0.75$ respectively. The feasible concurrent transmission vectors $(2, 1, 1)$ and $(1, 0, 0)$ maximize $U(\bar{x}_{G_1})$ and $U(\bar{x}_{G_2})$, respectively, where $U(\bar{x}_{G_1}) = 1$ and $U(\bar{x}_{G_2}) = 1$. Therefore, the maximum payoff under adversarial strategy q^*, by definition in (4), is $\max\limits_{\text{protocols } p} J(p, q^*) \leq \alpha_1 U(\bar{x}_{G_1}) + \alpha_2 U(\bar{x}_{G_2}) = 1$.

The protocol described in this paper p^* achieves a payoff close to 1. The legitimate nodes do not know *a priori* which nodes are legitimate or adversarial nor do they know the actions of the adversarial nodes. Instead, the initial estimate of the feasible non-disabled set of concurrent transmission vectors includes all possible concurrent transmission vectors $\mathcal{C}$. Over successive iterations, the protocol verifies whether the concurrent transmission vectors used in the schedule successfully delivered their data packets. Those that did not are permanently pruned from the estimated feasible set.

The concurrent transmission vector in $\mathcal{C}$ that maximizes $U(\bar{x})$ is $(2, 2, 2)$. This vector is scheduled in the first iteration, but since node C only cooperates with the protocol at 1 bit/s, at least one packet will either not be acknowledged or transmitted. At the end of the verification phase the legitimate nodes arrive at a consensus that $(2, 2, 2)$ is either infeasible or disabled and prune this concurrent transmission vector from the estimated feasible set. In the next iteration either $(2, 2, 1)$ or $(2, 1, 2)$ is scheduled. Each of these concurrent transmission vectors is pruned from the estimated feasible set in successive protocol iterations after they fail to deliver all expected packets. At the fourth iteration, the legitimate nodes try out the feasible concurrent transmission vector $(2, 1, 1)$. This concurrent transmission vector remains scheduled during subsequent iterations while node C cooperates with the protocol. A quarter-way through the operating lifetime, node C stops cooperating and starts jamming, at which point $(2, 1, 1)$, $(1, 1, 1)$, $(1, 1, 0)$, and $(1, 0, 1)$ are pruned from the estimated feasible set in successive protocol iterations. The concurrent transmission vector $(1, 0, 0)$ is scheduled for the remaining iterations.

Suppose there are a total of 1000 iterations, and that the throughput loss due to the overhead in each iteration is neglible. During the first quarter of the operating lifetime, three protocol iterations are required for the protocol to arrive at the feasible concurrent transmission vector $(2, 1, 1)$. Four iterations are required to arrive at $(1, 0, 0)$ after node C starts jamming. Hence, the effective payoff over the entire operating lifetime for the protocol is $J(p^*, q^*) = 0.25 * (1 * 247/250) + 0.75 * (1 * 746/750) = 0.993$.

The payoff $J(p^*, q^*) \to 1$ as the number of protocol iterations goes to infinity. In general the number of protocol iterations must be much larger than the set of possible concurrent transmission vectors $|\mathcal{C}|$. As long as the model assumptions (M1)–(M4) hold, the same process of successively verifying and pruning disabled/infeasible concurrent transmission vectors allows the network to recover the max-min payoff to within any desired ϵ for any utility function $U(\bar{x})$, regardless of the actions of the adversarial nodes.

VI. Proof of Theorem IV.3 and Theorem IV.4

The proof of Theorem IV.3 is based on a self-stabilizing, finite-state machine argument; any execution that brings the network to an undesirable state in which there are independent adjacent subnetworks, will eventually reach a desirable state in which there are no adjacent subnetworks. The state machine describes the state of a particular node at every time instant, where the state is determined by the index of the subnetwork in which the node participates, the index of the subnetwork to which the node is adjacent, and the statuses of both subnetworks (active, dormant, unborn). Each state falls into one of five categories. In any given state, each subnetwork carries out the actions specified by the protocol. These actions trigger events that move each node into a different state. Let $\tau_S(t)$ denote the reference clock of subnetwork S at time t with respect to the global reference clock t.

The "Active/Active" Initial Category A: There are two active adjacent subnetworks S_1 and S_j, with indices 1 and j respectively. One of the following reference clock conditions is satisfied: $T_r - \tau_{S_1}(t) > ((i_{\max} + 1)!)^3 (4 a_{\max} T_I)$ or $T_r - \tau_{S_j}(t) > ((i_{\max} + 1)!)^3 (2 a_{\max} T_I)$. The first condition ensures S_1 has enough time on its reference clock to move to a category C state if S_j enters the coma phase before completing one merge attempt for every connecting edge with S_1. The second condition ensures S_j has enough time on its reference clock to move to a category C state if S_1 enters the coma phase before S_j completes one merge attempt for every connecting edge with S_1. The state label is the unordered pair (S_1, S_j).

Actions in Category A: For each edge through which S_j receives probe packets from S_1, S_j makes at most one attempt to merge with S_1 [see (MR1)].

Event A1: Another subnetwork S merges with S_1 or with S_j to form S', before S_j can successfully merge with S_1. The reference clock of subnetwork S' starts at zero and the nodes in S' switch over to the encryption scheme assigned to S' [see (MR3)]. The nodes in S_1, S_j, and S' move to state (S_1, S') in category A or state (S', S_j) in category B.

Event A2: A merge attempt succeeds; S_j merges with S_1 to form a new subnetwork S. The reference clock of S starts at zero, and the nodes in S switch over to the encryption scheme assigned to S. The category of the new state of the nodes in S is determined by the status of the subnetworks adjacent to S (if any). If there is an active or dormant subnetwork S' adjacent to S then the nodes in S move to state (S', S) in category B or C, respectively (the reference clock condition is satisfied since the reference clock of S is starts at zero). If there are adjacent legitimate nodes that have yet to turn on *and* there are no active or dormant subnetworks adjacent to S, then S moves to state (S) in category D. If there are no active or dormant adjacent subnetworks and no adjacent legitimate nodes that will turn on then S moves to the terminal state E.

Event A3: The reference clock of either S_1 or S_j triggers the coma phase [see (MR4)] before S_j can successfully merge with S_1. The nodes in S_1 and S_j move to state (S_1, S_j) in category C.

The "Active/Active" General Category B: There are two adjacent active subnetworks S_i and S_j, with indices i and j, respectively where $1 < i < j$. One of the reference clock conditions in category A is satisfied. The state label is the unordered pair (S_i, S_j).

Actions in Category B: For each edge through which S_j receives probe packets from S_i, S_j makes at most $(i!)^3$ attempts to merge with S_i.

Event B1: Another subnetwork S merges with S_i or with S_j to form S' before S_j can successfully merge with S_i. The reference clock of subnetwork S' starts at zero and the nodes in S' switch over over to the encryption scheme assigned to S'. The nodes in S_i, S_j, and S' move to either state (S', S_j) or (S_i, S') in category B.

Event B2: A merge attempt succeeds; S_j merges with S_i to form a new subnetwork S. The reference clock of S starts at zero and the nodes in S switch over to the encryption scheme assigned to S. For the same reasons as those described in event A2, the nodes in S move to states in category B, C, D, or E.

Event B3: The reference clock of either S_i or S_j triggers the coma phase before S_j can successfully merge with S_i. The nodes in S_i and S_j move to state (S_i, S_j) in category C.

The "Active/Dormant" Category C: There are two adjacent subnetworks S_i and S_j, each of which are dormant and active respectively. The reference clock of S_j satisfies the following condition: $T_r - \tau_{S_j}(t) > ((i_{\max}+1)!)^3(2a_{\max})T_I$; S_j will not enter the coma phase before S_i successively completes $(k!)^3$ merge attempts with S_k for every edge that connects S_i and S_k and all $1 \leq k \leq i$ (see proof of Lemma V.11). The state label is the unordered pair (S_i, S_j).

Actions in Category C: For each edge through which a legitimate node l in S_i receives probe packets from S_j, node l makes at most $(j!)^3$ attempts to merge with S_j (see the coma phase merge rule).

Event C1: Another subnetwork S merges with S_j to form S' before node l can merge with S_j. The reference clock of S' starts at zero and the nodes in S' switch over to the encryption scheme assigned to S'. The nodes in S_i and S' move to state (S_i, S') in category C.

Event C2: A merge attempt succeeds; node l merges with S_j to form a new subnetwork S. The reference clock of S starts at zero and the nodes in S change over to the encryption scheme assigned to S. If $S_i \setminus l$ is non-empty then the nodes in subnetworks $S_i \setminus l$ and S move to state $(S_i \setminus l, S)$ in category C. If $S_i \setminus l$ is empty, then the nodes in S move to states in category B, C, D, or E for the same reasons as those described in event A2.

The "Active/Dormant/Unborn" Category D: There is a subnetwork S, either active or dormant, adjacent to an unborn legitimate node. The label of the state is (S).

Actions in Category D: The nodes in subnetwork S listen for any probe packets transmitted by an adjacent node.

Event D1: The unborn legitimate node l is born. The local clock of node l starts at zero. The nodes in S and node l move to state (l, S) in category A, B, or C.

Terminal State E: There are no adjacent subnetworks and no unborn legitimate nodes that will ever turn on.

The proof of Theorem IV.3 requires *closure* and *convergence*: first, that every outcome in any execution corresponds to some state in the state machine; second, that there are a finite number of states, that a node once leaving a state never returns to the same state, and that at least one of the events that force a node to leave a state is guaranteed to occur if the node carries out the actions prescribed by the protocol.

Lemma VI.1: There are a finite number of states.

Proof: The number of subnetworks is finite. □

Lemma VI.2: A node never returns to the same state.

Proof: A legitimate node never reuses a discarded subnetwork index (Lemma V.9). Moreover, a transition occurs when one of the subnetwork indices in the state label changes. □

Lemma VI.3: The actions in every category (except E) trigger a state transition.

Proof: Consider a state in category A. The events A1, A2, and A3, in changing S_1 and/or S_j, move the nodes in each subnetwork to a new state. We show that in the absence of A1 and A3, A2 is guaranteed to occur. Since S_1 and S_j are adjacent, S_j is guaranteed to detect S_1 (Lemma V.6). Hence, S_j will eventually attempt to merge with S_1. Since S_1, the subnetwork with the lowest index, never changes its schedule to attempt any merges of its own, the merge attempt is guaranteed to succeed (A2), provided S_1 or S_j are not first absorbed by an external subnetwork (A1) and their reference clocks do not expire (A3).

Now consider a state in category B with adjacent subnetworks S_i and S_j where $i < j$. We show that in the absence of B1 and B3, B2 is guaranteed to occur. The proof is by induction. The initial case where $i = 1$ is included in category A. Assume the statement is true for $1 < k < i$. From Lemma V.10 the number of instances in which S_i changes its schedule to attempt a merge is smaller than $(i!)^3$. Since S_j makes $(i!)^3$ merge attempts for each connecting edge between S_i and S_j, and S_i is adjacent to S_j some attempt must succeed, provided S_i or S_j are not first absorbed by an external subnetwork (B1) and their reference clocks do not expire (B3).

Next consider a state in category C with adjacent subnetworks S_j and S_i. We show that in the absence of C1, C2 is guaranteed to occur. A legitimate node l in S_i adjacent to S_j makes at most $(j!)^3$ attempts to merge with S_j for every connecting edge between l and S_j. By the same argument as before, S_j unilaterally changes its schedule fewer than $(j!)^3$ times. Therefore one attempt by node l is guaranteed to succeed, provided S_j is not absorbed by an external subnetwork (C1). Lemma V.11 ensures that node l has sufficient time left on its reference clock to make a merge attempt with any eligible subnetwork, and the reference clock condition ensures that S_j does not enter the coma phase before node l has completed all possible merge attempts.

Finally consider a state in category D. A state transition occurs when an adjacent legitimate node is born. By the category assumption, an unborn adjacent legitimate node exists. □

Lemma VI.4: The state machine is closed.

Proof: The proof is by contradiction. Suppose at some time t with respect to the global reference clock, a pair of adjacent subnetworks does not satisfy one of the reference clock conditions in A or B; $T_r - \tau_{S_i}(t) < ((i_{\max}+1)!)^3(4a_{\max})T_I$ and $T_r - \tau_{S_j}(t) < ((i_{\max}+1)!)^3(2a_{\max})T_I$. Since T_r is larger than $((i_{\max}+1)!)^3(4a_{\max})T_I$ (by choice) there must be some $t' < t$ where one reference clock condition is satisfied. Therefore S_i and S_j are either in a category A or B state at time t'. Moreover, the events in each category lead to a state transition, and the actions within a category are guaranteed to trigger an event (Lemma VI.3). Hence, a scenario in which S_i and S_j violate both reference clock conditions at time t is impossible. A similar argument also applies to the reference clock condition of category C. □

It follows from Lemmas VI.1, VI.2, VI.3, and VI.4, that every node will eventually reach the terminal state E. Since each subnetwork S_i makes at most $(i!)^3$ merge attempts (Lemma V.10) and there are at most $i_{\max}$ subnetworks, the total number of merge attempts required to reach the terminal state is no more than $((i_{\max}+1)!)^3 > \sum_{i<i_{\max}}(i!)^3$. To complete the proof of Theorem IV.3, set $T' := ((i_{\max}+1)!)^3(2a_{\max}T_I)$ since each merge attempt lasts for $2a_{\max}$ protocol iterations.

A. *Proof of Theorem IV.4*

To show that the throughput loss due to the protocol overhead ϵ can be made arbitrarily small, we will decompose ϵ into ϵ_l, ϵ_m, and ϵ_s; the long-run, medium-term, and short-term throughput loss respectively, where $(1-\epsilon) := (1-\epsilon_l)(1-\epsilon_m)(1-\epsilon_s)$.

The long-run throughput loss ϵ_l is the fraction of the operating lifetime, evaluated from the birth of the last legitimate node, in which an active legitimate node is not in the terminal state *and* operates according to a schedule that includes disabled concurrent transmission vectors. Both factors contributing to ϵ_l persist for only a finite number of iterations; in the first case each node reaches the terminal state within a bounded number of protocol iterations and in the second case, all disabled concurrent transmission vectors are eventually pruned from the estimated feasible set (Theorems IV.2) and IV.3). A *block* is defined as n_s successive protocol iterations, where n_s is the number of iterations that occur between sentinel phases. Let n_b denote the number of blocks that occur during the operating lifetime. The total number of protocol iterations n_r that occur during the operating lifetime is $n_r = n_b n_s$. Since the maximum number of merge attempts made by any legitimate node is $((i_{\max}+1)!)^3$ and each attempt runs for $2a_{\max}$ iterations, choose n_b so that $\epsilon_l = \max(|\mathcal{C}|, ((i_{\max}+1)!)^3 2a_{\max})/n_b$ for any desired ϵ_l, where $\mathcal{C}$ denotes the set of all possible concurrent transmission vectors. Thus, ϵ_l can be made arbitrarily small.

The medium-term throughput loss ϵ_m, the loss incurred from periodically entering the sentinel phase, can also be made arbitrarily small (Lemma V.7).

A protocol iteration consists of the recurrent neighbor and network discovery phases, the scheduling phase, the data transfer phase, and the verification phase. The short-term throughput loss ϵ_s is the loss incurred during a protocol iteration from the phases in which no data that directly contributes to the throughput is transferred: the recurrent neighbor and network discovery phases, and the verification phase. Given any n_r, it is possible to choose the protocol operating lifetime T, the neighbor and network discovery phases, the data transfer phase, and the verification phase to make ϵ_s arbitrarily small (see [22] and [9]). Thus Theorem IV.4 is proved.

VII. Concluding Remarks

An important challenge in designing secure distributed cyberphysical systems is guaranteeing the integrity of all internal communication, despite the actions of adversarial agents within the system itself. In this paper, we have considered a system in which agents can join at anytime, forcing the communication protocol to adapt accordingly. We have provided a constructive proof that it is possible to not only guarantee communication between all legitimate agents, but also to achieve optimal payoff. To this end, we have described a comprehensive protocol that does so.

The edge-tally supervised merge protocol is based on the architecture of its simpler counterpart in [22] for closed networks. Both protocols rely on a system of verification and adjustment for every possible failure mode, a scheme that is exponentially complex in the number of nodes, and therefore impractical to implement in its current form. However, the fundamental idea of confining adversarial behavior to a finite set of outcomes and sequentially eliminating each element of the set, can serve as a basis for more advanced schemes that converge more rapidly with respect to the number of nodes. We leave this very important problem open, in the hope that the work presented so far will stimulate further research in this direction.

References

[1] J. Choi, J. T. Chiang, D. Kim, and Y.-C. Hu, *Partial Deafness: A Novel Denial-of-Service Attack in 802.11 Networks*. New York, NY, USA: Springer-Verlag, 2010, vol. 50, pp. 235–252.

[2] Y. chun Hu, A. Perrig, and D. B. Johnson, "Wormhole attacks in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, pp. 370–380, 2006.

[3] T. Clausen and P. Jacquet, Optimized Link State Routing Protocol (OLSR), RFC 3626, Oct. 2003.

[4] D. Dolev, J. Halpern, and H. R. Strong, "On the possibility and impossibility of achieving clock synchronization," in *Proc. 16th Annu. ACM Symp. Theory of Computing, STOC*, New York, NY, USA, 1984, pp. 504–511.

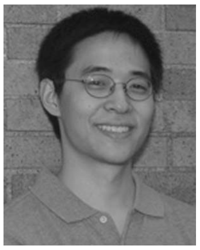
[5] J. R. Douceur, "The SYBIL attack," in *Proc. IPTPS*, 2002, pp. 251–260.

[6] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Rushing attacks and defense in wireless *ad hoc* network routing protocols," in *Proc. WiSec*, 2003, pp. 30–40.

[7] Y.-C. Hu, A. Perrig, and D. B. Johnson, “Ariadne: A secure on-demand routing protocol for *ad hoc* networks,” *Wireless Netw.*, vol. 11, no. 1/2, pp. 21–38, 2005.
[8] *IEEE Protocol 802.11. Draft Standard for Wireless LAN: Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Jul. 1996.
[9] Y.-C. H. Jonathan Ponniah and P. R. Kumar, “A clean slate approach to secure wireless networking,” *Found. Network.*, vol. 9, no. 1, pp. 1–105, 2014.
[10] F. Kelly, “Charging and rate control for elastic traffic,” *Eur. Trans. Telecommun.*, vol. 8, pp. 33–37, 1997.
[11] L. Lamport and P. M. Melliar-Smith, “Byzantine clock synchronization,” in *Proc. 3rd Annu. ACM Symp. PODC*, New York, NY, USA, 1984, pp. 68–74.
[12] L. Lamport, R. Shostak, and M. Pease, “The byzantine generals problem,” *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.
[13] X. Lin, N. Shroff, and R. Srikant, “A tutorial on cross-layer optimization in wireless networks,” *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.
[14] M. H. Manshaei, Q. Zhu, T. Alpcan, T. Bacşar, and J.-P. Hubaux, “Game theory meets network security and privacy,” *ACM Comput. Surv.*, vol. 45, no. 3, pp. 25:1–25:39, Jul. 2013.
[15] S. Marti, T. J. Giuli, K. Lai, and M. Baker, “Mitigating routing misbehavior in mobile *ad hoc* networks,” in *Proc. 6th Annu. Int. Conf. MobiCom*, New York, NY, USA, 2000, pp. 255–265.
[16] S. Narayanaswamy, V. Kawadia, R. S. Sreenivas, and P. R. Kumar, “Power control in ad-hoc networks: Theory, architecture, algorithm and implementation of the COMPOW protocol,” in *Proc. Eur. Wireless Conf.*, 2002, pp. 156–162.
[17] C. Perkins, E. Belding-Royer, and S. Das, Ad Hoc on Demand Distance Vector (aodv) Routing, RFC 3561, Jul. 2003.
[18] C. E. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing,” in *Proc. SIGCOMM*, London, U.K., Aug. 1994, pp. 234–244.
[19] R. Perlman, “Network layer protocols with Byzantine robustness,” Ph.D. dissertation, Dept. Electr. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 1988.
[20] A. Perrig, J. Stankovic, and D. Wagner, “Security in wireless sensor networks,” *Commun. ACM*, vol. 47, no. 6, pp. 53–57, Jun. 2004.
[21] J. Ponniah, Y.-C. Hu, and P. Kumar, “An orthogonal multiple access coding scheme,” *Commun. Inform. Syst.*, vol. 12, no. 1, pp. 41–76, 2012.
[22] J. Ponniah, Y.-C. Hu, and P. Kumar, “A system-theoretic clean slate approach to provably secure *ad hoc* wireless networking,” *IEEE Trans. Control Netw. Syst.*, to be published, doi: 10.1109/TCNS.2015.2428309.

Jonathan Ponniah (M’15) received the B.A.Sc. and M.A.Sc. degrees in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 2006 and 2008, respectively, and the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign, in 2013.

He is currently a Post-doctoral Researcher at Texas A&M University, College Station. His research interests include distributed systems, game theory, network optimization, and network information theory.



Yih-Chun Hu (SM’08) received the B.S. degree in computer science and pure mathematics from the University of Washington, Seattle, USA, in 1997, and the Ph.D. degree in computer science from Carnegie Mellon University, Pittsburgh, PA, in 2003, with a thesis focused on security and performance in wireless *ad hoc* networks.

He was a Post-doctoral Researcher with the University of California, Berkeley, conducting research in the area of network security. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana. His research interests include systems and network security.

P. R. Kumar (F’88) received the B. Tech. degree in electrical engineering (electronics) from I.I.T. Madras, Madras, India, in 1973, and the M.S. and D.Sc. degrees in systems science and mathematics from Washington University, St. Louis, in 1975 and 1977, respectively.

From 1977 to 1984, he was a Faculty Member in the Department of Mathematics, University of Maryland Baltimore County. From 1985 to 2011, he was a faculty member in the Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois. Currently, he is with Texas A&M University, College Station, where he holds the College of Engineering Chair in Computer Engineering. He has worked on problems in game theory, adaptive control, stochastic systems, simulated annealing, neural networks, machine learning, queueing networks, manufacturing systems, scheduling, wafer fabrication plants, and information theory. His research is currently focused on energy systems, wireless networks, secure networking, automated transportation, and cyberphysical systems.

Dr. Kumar is a member of the National Academy of Engineering of the USA, and a Fellow of the Academy of Sciences of the Developing World. He was awarded an honorary doctorate by the Swiss Federal Institute of Technology (Eidgenossische Technische Hochschule) in Zurich. He received the Outstanding Contribution Award of ACM SIGMOBILE, the IEEE Field Award for Control Systems, the Donald P. Eckman Award of the American Automatic Control Council, and the Fred W. Ellersick Prize of the IEEE Communications Society. He is an ACM Fellow. He was a Guest Chair Professor and Leader of the Guest Chair Professor Group on Wireless Communication and Networking at Tsinghua University, Beijing, China. He is an Honorary Professor at IIT Hyderabad and a D. J. Gandhi Distinguished Visiting Professor at IIT Bombay. He was awarded the Distinguished Alumnus Award from IIT Madras, the Alumni Achievement Award from Washington University, St. Louis, and the Daniel C. Drucker Eminent Faculty Award from the College of Engineering at the University of Illinois.