Stone Age Distributed Computing (Extended Abstract)

Yuval Emek
Distributed Computing Group
ETH Zurich, Switzerland
yemek@ethz.ch

Roger Wattenhofer[*]
Distributed Computing Group
ETH Zurich, Switzerland
wattenhofer@ethz.ch

ABSTRACT

A new model that depicts a network of randomized finite state machines operating in an asynchronous environment is introduced. This model, that can be viewed as a hybrid of the message passing model and cellular automata is suitable for applying the distributed computing lens to the study of networks of sub-microprocessor devices, e.g., biological cellular networks and man-made nano-networks. Although the computation and communication capabilities of each individual device in the new model are, by design, much weaker than those of an abstract computer, we show that some of the most important and extensively studied distributed computing problems can still be solved efficiently.

Categories and Subject Descriptors

F.1.1 [**Computation by Abstract Devices**]: Models of computation

General Terms

Theory

Keywords

Cellular automata, efficient algorithms, finite state machines, message passing

1. INTRODUCTION

Due to the major role that the Internet plays today, models targeted at understanding the fundamental properties of networks focus mainly on "Internet-capable" devices. Indeed, the standard network model in distributed computing is the so called *message passing* model, where nodes may exchange large messages with their neighbors, and perform arbitrary local computations. Recently, there is a trend to study distributed computing aspects in networks of sub-microprocessor devices, e.g., networks of biological cells [3, 16] or nano-scale mechanical devices [4]. However, the suitability of the message passing model to these types of networks is far from being certain: Do tiny bio/nano nodes "compute" and/or "communicate" essentially the same as a computer? Since such nodes will be fundamentally more limited than silicon-based devices, we believe that there is a need for a network model, where nodes are by design below the computation and communication capabilities of Turing machines.

Networked finite state machines. In this paper, we introduce a new model, referred to as *networked finite state machines (nFSM)*, that depicts a network of randomized finite state machines (a.k.a. automata) progressing in asynchronous steps (refer to Section 2 for a formal description). Under the nFSM model, nodes communicate by transmitting messages belonging to some finite communication alphabet Σ such that a message $\sigma \in \Sigma$ transmitted by node u is delivered to its neighbors (the same σ to all neighbors) in an asynchronous fashion. Each neighbor v of u has a port corresponding to u in which the last message delivered from u is stored.

The access of node v to its ports is limited: In each step of v's execution, the next state and the message transmitted by v at this step are determined by v's current state and by the current number $\sharp(\sigma)$ of appearances of each letter $\sigma \in \Sigma$ in v's ports. The crux of the model is that $\sharp(\sigma)$ is calculated according to the *one-two-many*[1] principle: the node can only count up to some predetermined *bounding parameter* $b \in \mathbb{Z}_{>0}$ and any value of $\sharp(\sigma)$ larger than b cannot be distinguished from b.

The nFSM model satisfies the following *model requirements*, that we believe, make it more applicable to the study of general networks consisting of weaker devices such as those mentioned above:

(M1) The model is applicable to arbitrary network topologies.

(M2) All nodes run the same protocol executed by a (randomized) FSM.

(M3) The network operates in a fully asynchronous environment with adversarial node activation and message delivery delays.

[*]Part of this work was done while the author was in Microsoft Research, Redmond, WA.

[1]The one-two-many theory states that some small isolated cultures (e.g., the Piraha tribe of the Amazon [26]) did not develop a counting system that goes beyond 2. This is reflected in their languages that include words for "1", "2", and "many" that stands for any number larger than 2.

(M4) All features of the FSM (specifically, the state set Q, message alphabet Σ, and bounding parameter b) are of constant size independent of any parameter of the network (including the degree of the node executing the FSM).

The last requirement is perhaps the most interesting one as it implies that a node cannot perform any calculation that involves variables beyond some predetermined constant. This comes in contrast to many distributed algorithms operating under the message passing model that strongly rely on the ability of a node to perform such calculations (e.g., count up to some parameter of the network or a function thereof).

Results. Our investigation of the new nFSM model begins by observing that the computational power of a network operating under this model is not stronger than (and in some sense equivalent to) that of a randomized Turing machine with linear space bound (due to space limitations, this part is deferred to the full version). Since the computational power of a network operating under the message passing model is trivially equivalent to that of a (general) Turing machine, there exist distributed problems that can be solved by a message passing algorithm in constant time but cannot be solved by an nFSM algorithm at all. Nevertheless, as the main technical contribution of this paper, we show that some of the most important problems in distributed computing admit efficient (namely, with polylogarithmic run-time) nFSM algorithms. Specifically, problems such as maximal independent set, node coloring, and maximal matching that have been extensively studied since the early 1980s, always assuming a network of some sort of abstract computers, can in fact be solved, and fast, when each device is nothing but a FSM.

Applicability to biological cellular networks. Regardless of the theoretical interest in implementing efficient algorithms using weaker assumptions, we believe that our new model and results should be appealing to anyone interested in understanding the computational aspects of biological cellular networks. A basic dogma in biology (see, e.g., [41]) states that all cells communicate and that they do so by emitting special kinds of ligand molecules (e.g. the DSL family of proteins) that bind in a reversible, non-covalent, fashion to designated receptors (e.g. the NOTCH family of transmembrane receptors) in neighboring cells. These, in turn, release some intracellular signaling proteins that penetrate the nucleus to modify gene expression (determining the cell's actions).

Translated to the language of the nFSM model, the different types of ligand molecules correspond to the different letters in the communication alphabet, where an emission of a ligand corresponds to transmitting a letter. The effect that the ligands-receptors binding has on the concentration level of the signaling proteins in the nucleus corresponds to the manner in which a node in our model interprets the content of its ports. (The one-two-many counting is the discrete analogue of the ability to distinguish between different concentration levels, considering the fact that once the concentration exceeds some threshold, a further increase cannot be detected.) Using an FSM as the underlying computational model of the individual node seems to be the right choice especially in the biological setting as demonstrated by Benenson et al. [17] who showed that essentially all FSMs can be implemented by enzymes found in cells' nuclei. One may wonder if the specific problems studied in the current paper have any relevance to biology. Indeed, Afek et al. [3] discovered that a biological process that occurs during the development of the nervous system of the Drosophila melanogaster is in fact equivalent to solving the MIS problem.

Related work and comparison to other models. As mentioned above, the message passing model is the gold standard when it comes to understanding distributed network algorithms [32, 40]. Several variants exist for this model, differing mainly in the bounds imposed on the message size (e.g., the *congest* and *local* models [44]) and the level of synchronization. Indeed, most theoretical literature dealing with distributed network algorithms relies on one of these variants. Our nFSM model adopts the concept of an asynchronous message-based communication scheme from the message passing literature.

As the traditional message passing model allows for sending different messages to different neighbors in each round of the execution, it was too powerful for many settings. In particular, with the proliferation of wireless networks, more restrictive message passing models appeared such as the *radio network* model [20]. Over the years, several variants of the radio network model were introduced, the most extreme one in terms of its weak communication capabilities is the *beeping* model [24, 22], where in each round a node can either beep or stay silent, and can only distinguish between the case in which no node in its neighborhood beeps and the case in which at least one node beeps. Efficient algorithms and lower bounds for the MIS problem under the beeping model were developed by Afek et al. [3, 2]. Note that the beeping model resembles our nFSM model in the sense that the "beeping rule" can be viewed as counting under the one-two-many principle with bounding parameter $b = 1$. However, it is much stronger in other perspectives: (i) the beeping model assumes synchronous communication and does not seem to have a natural asynchronous variant; and (ii) the local computation is performed by a Turing machine whose memory is allowed to grow with time and with the network size (this is crucial for the algorithms of Afek et al. [3, 2]). In that regard, the beeping model is still too strong to capture the behavior of biological cellular networks.

Our nFSM model is also closely related to (and inspired by) the extensively studied *cellular automaton* model [47, 25, 48] that captures a network of FSMs, arranged in a grid topology (some other highly regular topologies were also considered), where the transition of each node depends on its current state and the states of its neighbors. Still, the nFSM model differs from the cellular automaton model in many aspects. In particular, the latter model is not applicable to non-regular network topologies and although a small fraction of the cellular automata literature is dedicated to cellular automata with asynchronous node activation [38, 39] (see also [1]), these do not support asynchronous message delivery. As such, cellular automata do not seem to provide a good abstraction for the sub-microprocessor networks we would like to focus on. Moreover, the main goal of our work is to study to what extent can such networks compute solutions quickly, a goal that lies outside the typical interest of the cellular automata community.

Another model that resembles the nFSM model is that of *communicating automata* [18]. This model also assumes that each node in the network operates an FSM in an asynchronous manner, however the steps of the FSMs are message driven: for each state q of node v and for each message

m that node v may receive from an adjacent node u while residing in state q, the transition function of v should have an entry characterized by the 3-tuple (q, u, m) that determines its next move. Consequently, different nodes would typically operate different FSMs and the size of the FSM operated by node v inherently depends on the degree of v and thus, the communicating automata model is still too strong to faithfully represent biological cellular networks.

The *population protocols* model, introduced by Angluin et al. [6] (see also [8, 37]), depicts a network of finite state machines communicating through pairwise rendezvous controlled by a fair adversarial scheduler. While in the full version we reveal some interesting connections between the nFSM model and population protocols, there are two conceptual differences between these models: First, population protocols are only required to *eventually converge* to a correct output and are allowed to return arbitrary (wrong) outputs beforehand. This provides population protocols with the power to solve, e.g., the consensus problem in arbitrary networks, in contrast to nFSM protocols that should irrevocably return a correct output (see Section 2) and therefore, cannot solve this problem (cf. [23]). Second, while the focus of the current paper is mainly on run-time complexity, there is an inherent problem with establishing run-time bounds for population protocols due to the nature of the adversarial scheduler that can delay the execution for arbitrarily long periods. Indeed, the population protocols literature is typically concerned with what can be computed, rather than how fast. An exception is the probabilistic variant of the model [7, 19], where interactions are selected randomly, rather than adversarially, but this variant is no longer fully asynchronous (in the adversarial sense). On top of that, the rendezvous based communication does not seem to fit the communication mechanisms in most biological cellular networks.

2. MODEL

Throughout, we assume a network represented by a finite undirected graph $G = (V, E)$. Under the *networked finite state machines (nFSM)* model, each node $v \in V$ runs a protocol depicted by the 8-tuple

$$\Pi = \langle Q, Q_I, Q_O, \Sigma, \sigma_0, b, \lambda, \delta \rangle \ , \tag{1}$$

where

- Q is a finite set of *states*;
- $Q_I \subseteq Q$ is the subset of *input states*;
- $Q_O \subseteq Q$ is the subset of *output states*;
- Σ is a finite *communication alphabet*;
- $\sigma_0 \in \Sigma$ is the *initial letter*;
- $b \in \mathbb{Z}_{>0}$ is a *bounding parameter*; let $B = \{0, 1, \dots, b-1, {}^{\geq}b\}$ be a set of $b+1$ distinguishable symbols;
- $\lambda : Q \to \Sigma$ assigns a *query letter* $\sigma \in \Sigma$ to every state $q \in Q$; and
- $\delta : Q \times B \to 2^{Q \times (\Sigma \cup \{\varepsilon\})}$ is the *transition function*.

It is important to point out that protocol Π is oblivious to the graph G. In fact, the number of states in Q, the size of the alphabet Σ, and the bounding parameter b are all assumed to be universal constants, independent of any parameter of the graph G. In particular, the protocol executed by node $v \in V$ does not depend on the degree of v in G. We now turn to describe the semantics of the nFSM model.

Communication. Node v communicates with its adjacent nodes in G by *transmitting* messages. A transmitted message consists of a single letter $\sigma \in \Sigma$ and it is assumed that this letter is delivered to all neighbors u of v. Each neighbor u has a *port* $\psi_u(v)$ (a different port for every adjacent node v) in which the last message σ received from v is stored. At the beginning of the execution, all ports store the initial letter σ_0. It will be convenient to consider the case in which v does not transmit any message (and hence does not affect the corresponding ports at the adjacent nodes) as a transmission of the special *empty symbol* ε.
Execution. The execution of node v progresses in discrete *steps* indexed by the positive integers. In each step $t \in \mathbb{Z}_{>0}$, node v resides in some state $q \in Q$. Let $\lambda(q) = \sigma \in \Sigma$ be the query letter that λ assigns to state q and let $\sharp(\sigma)$ be the number of appearances of σ in v's ports in step t. Then, the pair (q', σ') of state $q' \in Q$ in which v resides in step $t+1$ and message $\sigma' \in \Sigma \cup \{\varepsilon\}$ transmitted by v in step t (recall that ε indicates that no message is transmitted) is chosen *uniformly at random*[2] (and independently of all other random choices) among the pairs in

$$\delta\left(q, \beta_b\left(\sharp(\sigma)\right)\right) \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \ ,$$

where $\beta_b : \mathbb{Z}_{\geq 0} \to B$ is defined as

$$\beta_b(x) = \begin{cases} x & \text{if } 0 \leq x \leq b-1 \ ; \\ {}^{\geq}b & \text{otherwise} \ . \end{cases}$$

Informally, this can be thought of as if v queries its ports for appearances of σ and "observes" the exact value of $\sharp(\sigma)$ as long as it is smaller than the bounding parameter b; otherwise, v merely "observes" that $\sharp(\sigma) \geq b$ which is indicated by the symbol ${}^{\geq}b$.
Input and output. Initially (in step 1), each node resides in one of the input states of Q_I. The choice of the initial state of node $v \in V$ reflects the input passed to v at the beginning of the execution. This allows our model to cope with distributed problems in which different nodes get different input symbols. When dealing with problems in which the nodes do not get any initial input (such as the graph theoretic problems addressed in this paper), we shall assume that Q_I contains a single state referred to as the *initial* state.

The output states Q_O are mapped to the possible output values of the problem. For each possible output value o, it is required that the subset $P_o \subseteq Q_O$ of output states mapped to o form a *sink* with respect to the transition function δ in the sense that a node v that moves to a P_o-state will remain in P_o indefinitely, in which case the output of v is determined (irrevocably) to be o. We say that the (global) execution of the protocol is in an *output configuration* if all nodes reside in output states of Q_O.
Asynchrony. The nodes are assumed to operate in an *asynchronous* environment. This asynchrony has two facets: First, for the sake of convenience, we assume that the actual application of the transition function in each step $t \in \mathbb{Z}_{>0}$ of node $v \in V$ is instantaneous (namely, lasts zero time) and occurs at the end of the step;[3] the length of step t of node v, denoted $L_{v,t}$, is defined as the time difference between the application of the transition function in step $t-1$ and that

[2]The protocol is deterministic if the images under δ are always singleton subsets of $Q \times (\Sigma \cup \{\varepsilon\})$.
[3]This assumption can be lifted at the cost of a more complicated definition of the adversarial policy described soon.

of step t. It is assumed that $L_{v,t}$ is finite, but apart from that, we do not make any other assumptions on this length, that is, the step length $L_{v,t}$ is determined by the adversary independently of all other step lengths $L_{v',t'}$. In particular, we do not assume any synchronization between the steps of different nodes whatsoever.

The second facet of the asynchronous environment is that a message transmitted by node v in step t (if such a message is transmitted) is assumed to reach the port $\psi_u(v)$ of an adjacent node u after a finite time delay, denoted $D_{v,t,u}$. We assume that if v transmits message $\sigma_1 \in \Sigma$ in step t_1 and message $\sigma_2 \in \Sigma$ in step $t_2 > t_1$, then σ_1 reaches u before σ_2 does. Apart from this "FIFO" assumption, we do not make any other assumptions on the delays $D_{v,t,u}$. In particular, this means that under certain circumstances, the adversary may overwrite message σ_1 with message σ_2 in port $\psi_u(v)$ of u so that u will never "know" that message σ_1 was transmitted.[4]

Consequently, a *policy* of the adversary is captured by: (1) the length $L_{v,t}$ of step t of node v for every $v \in V$ and $t \in \mathbb{Z}_{>0}$; and (2) the delay $D_{v,t,u}$ of the delivery of the transmission of node v in step t to an adjacent node u for every $v \in V$, $t \in \mathbb{Z}_{>0}$, and $u \in N(v)$.[5] Assuming that the adversary is oblivious to the random coin tosses of the nodes, an adversarial policy is depicted by infinite sequences of $L_{v,t}$ and $D_{v,t,u}$ parameters.

For further information on asynchronous environments, we refer the reader to one of the standard textbooks [34, 40].

Correctness and run-time measures. A protocol Π for problem P is said to be *correct* under the nFSM model if for every instance of P and for every adversarial policy, Π reaches an output configuration w.p. 1, and for every output configuration reached by Π w.p. > 0, the output of the nodes is a valid solution to P.[6] Given a correct protocol Π, the complexity measure that interests us in the current paper is the *run-time* of Π defined as the (possibly fractional) number of *time units* that pass from the beginning of the execution until an output configuration is reached, where a time unit is defined[7] to be the maximum among all step length parameters $L_{v,t}$ and delivery delay parameters $D_{v,t,u}$ in the adversarial policy (cf. [9, 40]). Following the standard procedure in this regard, we say that the run-time of a correct protocol Π for problem P is $f(n)$ if for every n-node instance of P and for every adversarial policy, the run-time of Π is at most $f(n)$ in expectation and w.h.p. The protocol is said to be *efficient* if its run-time is polylogarithmic in the size of the network (cf. [32]).

2.1 Multi-letter queries

[4]Often, much stronger assumptions are made in the literature. For example, a common assumption for asynchronous environments is that the port of node u corresponding to the adjacent node v is implemented by a buffer so that messages cannot be "lost". We do not make any such assumption for our nFSM model.

[5]We use the standard notation $N(v)$ for the *neighborhood* of node v in G, namely, the subset of nodes adjacent to v.

[6]Throughout, w.p. and w.h.p. abbreviate "with probability" and "with high probability", respectively.

[7]Note that time units are defined solely for the purpose of the analysis. Under an asynchronous environment, the nodes have no notion of time and in particular, they cannot measure a single time unit.

According to the model as presented thus far, each state $q \in Q$ is associated with a single query letter $\sigma = \lambda(q)$ and the application of the transition function when node v resides in state q is determined by $\beta_b(\sharp(\sigma))$, namely the number of appearances of the letter σ in the ports of v counted up to the bounding parameter b. However, in many applications, the transition of node v residing in state q depends on multiple-letters. This motivates the introduction of the *multi-letter query* feature that replaces an nFSM protocol as described in (1) by the 7-tuple

$$\Pi = \langle Q, Q_I, Q_O, \Sigma, \sigma_0, b, \delta \rangle \ ,$$

where Q, Q_I, Q_O, Σ, σ_0, and b are defined (and play the same role) as in (1), and the domain of the transition function δ is extended so that

$$\delta : Q \times B^{\Sigma} \to 2^{Q \times (\Sigma \cup \{\varepsilon\})} \ .$$

The semantics of the nFSM model when augmented with the multi-letter query feature is as follows. Suppose that in step $t \in \mathbb{Z}_{>0}$, node v resides in state $q \in Q$ and the number of appearances of σ in v's ports in step t is $\sharp(\sigma)$ for every letter $\sigma \in \Sigma$. Then, the pair (q', σ') of state $q' \in Q$ in which v resides in step $t+1$ and message $\sigma' \in \Sigma \cup \{\varepsilon\}$ transmitted by v in step t is chosen uniformly at random among the pairs in

$$\delta\left(q, \langle \beta_b(\sharp(\sigma)) \rangle_{\sigma \in \Sigma}\right) \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \ ,$$

where $\langle \beta_b(\sharp(\sigma)) \rangle_{\sigma \in \Sigma}$ denotes the vector mapping $\beta_b(\sharp(\sigma))$ to each $\sigma \in \Sigma$. One may wonder if the nFSM model augmented with the multi-letter query feature is strictly stronger than the nFSM model without that feature; in Section 3, we show that this is not the case.

3. CONVENIENT TRANSFORMATIONS

In this section, we show that the nFSM protocol designer may, in fact, assume a slightly more "user-friendly" environment than the one described in Section 2. This is based on the design of black-box *compilers* transforming a protocol that makes strong assumptions on the environment into one that does not make any such assumptions.

As described in Section 2, the nFSM model assumes an asynchronous environment. Nevertheless, it will be convenient to extend the nFSM model to *synchronous* environments. One natural such extension augments the model described in Section 2 with the following two *synchronization properties* that should hold for every two adjacent nodes $u, v \in V$ and for every $t \in \mathbb{Z}_{>0}$:

(S1) when node u is in step t, node v is in step $t-1$, t, or $t+1$; and

(S2) at the end of step $t+1$ of u, port $\psi_u(v)$ stores the message transmitted by v in step t of v's execution (or the last message transmitted by v prior to step t if v does not transmit any message in step t).

An environment in which properties (S1) and (S2) are guaranteed to hold is called a *locally synchronous* environment. Local-only communication can never achieve global synchrony, however, research in the message passing model has shown that local synchrony is often sufficient to provide efficient algorithms [9, 11, 10]. To distinguish a protocol assumed to operate in a locally synchronous environment from those making no such assumptions, we shall often refer to the execution steps of the former as *rounds* (cf. fully synchronized protocols).

THEOREM 1. *Every nFSM protocol $\Pi = \langle Q, Q_I, Q_O, \Sigma, \sigma_0, b, \lambda, \delta \rangle$ designed to operate in a locally synchronous environment can be simulated in an asynchronous environment by a protocol $\widehat{\Pi}$ with the same bounding parameter b at the cost of a constant multiplicative run-time overhead.*

The procedure in charge of the simulation promised in Theorem 1 (whose proof is deferred to the full version) is referred to as a *synchronizer* [9].

Now that we may assume a synchronous environment, it is easy to show that by augmenting the nFSM model with the multi-letter query feature introduced in Section 2.1, one does not (asymptotically) enhance the power of the model. Indeed, at the cost of increasing the number of states and the run-time by constant factors, we can subdivide each round into $|\Sigma|$ subrounds, dedicating each subround to a different letter in Σ, so that at the end of the round, the state of v reflects $\beta_b(\sharp(\sigma))$ for every $\sigma \in \Sigma$.

THEOREM 2. *Every nFSM protocol with the multi-letter query feature can be simulated by an nFSM protocol without this feature and the same bounding parameter b at the cost of a constant multiplicative run-time overhead.*

4. EFFICIENT ALGORITHMS

As stated earlier, the main technical contribution of this paper is cast in the development of efficient nFSM algorithms for some of the most important and extensively studied problems in distributed computing. These problems include maximal independent set, maximal 2-hop independent set, node coloring of bounded degree graphs with $\Delta + 1$ colors, node 2-hop coloring of bounded degree graphs with $\Delta^2 + 1$ colors, node coloring of (undirected) trees with 3 colors, and maximal matching (where we use a small unavoidable modification of the model). The maximal independent set problem is treated in Section 4.1; due to space limitation, the treatment of all other problems is deferred to the full version.

4.1 Maximal independent set

Given a graph $G = (V, E)$, the *maximal independent set (MIS)* problem asks for a node subset $U \subseteq V$ which is independent in the sense that $(U \times U) \cap E = \emptyset$, and maximal in the sense that $U' \subseteq V$ is not independent for every $U' \supset U$. The challenge of designing a fast distributed MIS algorithm was first posed by Valiant in the early 1980s [46]. Distributed MIS algorithms with logarithmic run-time operating in the message passing model were subsequently presented by Luby [33] and independently, by Alon et al. [5].[8] Luby's algorithm has since become a specimen of distributed algorithms; in the last 25 years, researchers have tried to improve it, if only e.g., with an improved bit complexity [36], on special graph classes [42, 31, 14], or in a weaker communication model [2]. An $\Omega(\sqrt{\log n})$ lower bound on the run-time of any distributed MIS algorithm operating in the message passing model was established by Kuhn et al. [30]. Our goal in this section is to establish the following theorem.

[8]The focus of [33] and [5] was actually on the PRAM model, but their algorithms can be adapted to the message passing model.

THEOREM 3. *There exists an nFSM protocol with bounding parameter $b = 1$ that computes an MIS for any n-node graph in time $O(\log^2 n)$.*

Outline of the key technical ideas. The protocol promised by Theorem 3 is inspired by the existing message passing MIS algorithms. Common to all these algorithms is that they are based on the concept of grouping consecutive rounds into *phases*, where in each phase, nodes compete against their neighbors over the right to join the MIS. Existing implementations of such competitions require at least one of the following three capabilities: (1) performing calculations that involve super-constant variables; (2) communicating with each neighbor independently; or (3) sending messages of a logarithmic size. The first two capabilities are clearly out of the question for an nFSM protocol. The third one is also not supported by the nFSM model, but perhaps one can divide a message with a logarithmic number of bits over logarithmically many rounds, sending $O(1)$ bits per round (cf. Algorithm B in [36])?

This naive attempt results in super-constant long phases, while no FSM can count the rounds in such phases — a task essential for deciding if the current phase is over and the next one should begin. Furthermore, to guarantee fair competition, the phases must be aligned across the network, thus ruling out the possibility to start node v's phase i before phase $i-1$ of some node $u \neq v$ is finished. In fact, an efficient algorithm that requires $\omega(1)$ long aligned phases cannot be implemented under the nFSM model. So, how can we decide if node v joins the MIS using constant size messages without the ability to maintain long aligned phases?

This issue is resolved by relaxing the requirements that the phases are aligned and of a predetermined length, introducing a feature referred to as a *tournament*. Our tournaments are only "softly" aligned and their lengths are determined probabilistically, in a manner that can be maintained under the nFSM model. Nevertheless, they enable a fair competition between neighboring nodes, as desired.

The protocol. Employing Theorems 1 and 2, we assume a locally synchronous environment and use multiple-letter queries. The state set of the protocol is $Q = \{\texttt{WIN}, \texttt{LOSE}, \texttt{DOWN}_1, \texttt{DOWN}_2, \texttt{UP}_0, \texttt{UP}_1, \texttt{UP}_2\}$, with $Q_I = \{\texttt{DOWN}_1\}$ (the initial state of all nodes) and $Q_O = \{\texttt{WIN}, \texttt{LOSE}\}$, where WIN (respectively, LOSE) indicates membership (resp., non-membership) in the MIS output by the protocol. The states in $Q_A = Q - Q_O$ are called the *active* states and a node in an active state is referred to as an *active* node. We take the communication alphabet Σ to be identical to the state set Q, where the letter transmissions are designed so that node v transmits letter q whenever it moves to state q from some state $q' \neq q$; no letter is transmitted in a round at which v remains in the same state. Letter $\texttt{DOWN}_1$ is the initial letter stored in all ports at the beginning of the execution. The bounding parameter is set to $b = 1$.

A schematic description of the transition function is provided in Figure 1; its logic is as follows. Each state $q \in Q_A$ has a subset $D(q) \subseteq Q_A - \{q\}$ of *delaying states*: node v remains in the current state q if and only if (at least) one of its neighbors is in some state in $D(q)$. This is implemented by querying on the letters (corresponding to the states) in $D(q)$, staying in state q as long as at least one of these letters is found in the ports. Specifically, state $\texttt{DOWN}_1$ is delayed by state $\texttt{DOWN}_2$, which is delayed by all three UP states. State $\texttt{UP}_j$, $j = 0, 1, 2$, is delayed by state $\texttt{UP}_{j-1 \bmod 3}$, where state

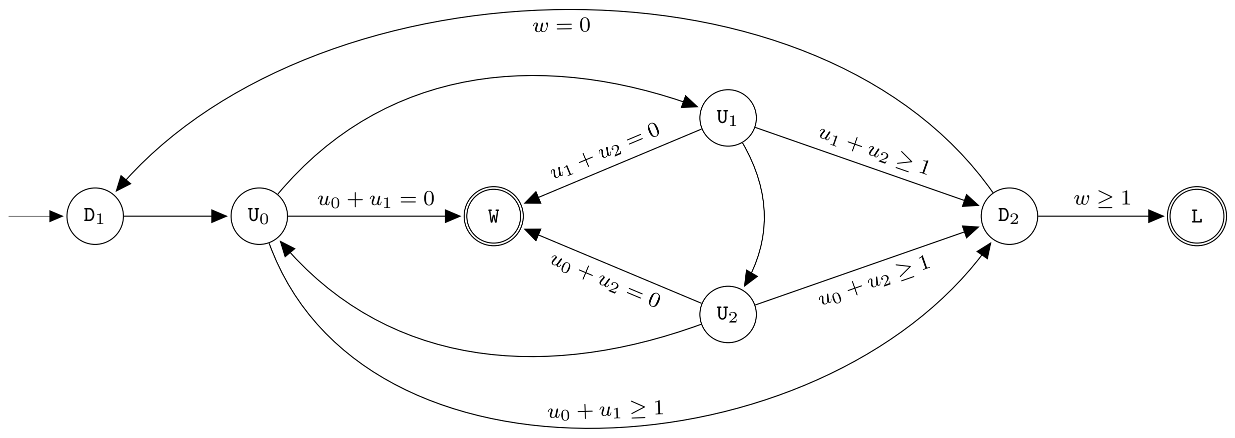

Figure 1: The transition function of the MIS protocol with state names abbreviated by their first (capital) letter. The node stays in state q (a.k.a. *delayed*) as long as letter q' appears (at least once) in its ports for any state q' such that a $q' \to q$ transition is defined (for clarity, this is omitted from the figure). Assuming that the node is not delayed, each transition specified in the figure is associated with a condition on the number of appearances of the query letters in the ports (depicted by the corresponding lower-case letter) so that the transition is followed only if the condition is satisfied (an empty condition is satisfied by all port configurations); if some port configuration satisfies several transition conditions, then one of them is chosen uniformly at random.

$\texttt{UP}_0$ is also delayed by state $\texttt{DOWN}_1$.

States WIN and LOSE are sinks. Assuming that an active node v does not find any delaying letter in its ports, the logic of the UP and DOWN states is as follows. From state $\texttt{DOWN}_1$, v moves to state $\texttt{UP}_0$. From state $\texttt{DOWN}_2$, v moves to state $\texttt{DOWN}_1$ if $\sharp(\texttt{WIN}) = 0$, that is, if it does not find any WIN letter in its ports; otherwise, it moves to state LOSE. When in state $\texttt{UP}_j$, v tosses a fair coin and proceeds as follows: if the coin turns head, then v moves to state $\texttt{UP}_{j+1 \bmod 3}$; if the coin turns tail, then v moves to state WIN if $\sharp(\texttt{UP}_j) = \sharp(\texttt{UP}_{j+1 \bmod 3}) = 0$ (note that $\sharp(\texttt{UP}_{j-1 \bmod 3})$ must be 0 as $\texttt{UP}_{j-1 \bmod 3} \in D(\texttt{UP}_j)$); and to state $\texttt{DOWN}_2$ otherwise. This completes the description of our nFSM protocol for the MIS problem.

Turns and tournaments. Our protocol is designed so that an active node v traverses the DOWN and UP states in a (double-)circular fashion: an inner loop of the UP states (moving from state $\texttt{UP}_j$ to state $\texttt{UP}_{j+1 \bmod 3}$) nested within an outer loop consisting of the DOWN states and the inner loop. Of course, v may spend more than one round at each state $q \in Q_A$ (delayed by adjacent nodes in states $D(q)$); we refer to a maximal contiguous sequence of rounds that v spends in the same state $q \in Q_A$ as a *q-turn*, or simply as a *turn* if the actual state q is irrelevant. A maximal contiguous sequence of turns that starts at a $\texttt{DOWN}_1$-turn and does not include any other $\texttt{DOWN}_1$-turn (i.e., a single iteration of the outer loop) is referred to as a *tournament*. We index the tournaments and the turns within a tournament by the positive integers. Note that by definition, every tournament i of v starts with a $\texttt{DOWN}_1$-turn, followed by a non-empty sequence of UP-turns; tournament i can end with an UP-turn from which v moves to state WIN, with a $\texttt{DOWN}_2$-turn from which v moves to state LOSE, or with a $\texttt{DOWN}_2$-turn from which v moves to state $\texttt{DOWN}_1$ starting tournament $i + 1$. The following observation is established by induction on the rounds.

Observation 1. *Consider some active node $v \in V$ in turn $j \in \mathbb{Z}_{>0}$ of tournament $i \in \mathbb{Z}_{>0}$ and some active node $u \in N(v)$.*

- *If this is a $\texttt{DOWN}_1$-turn of v ($j = 1$), then u is in either (a) the last ($\texttt{DOWN}_2$-)turn of tournament $i - 1$; (b) turn 1 of tournament i; or (c) turn 2 of tournament i.*
- *If this is an UP-turn of v ($j \geq 2$), then u is in either (a) turn $j - 1$ of tournament i; (b) turn j of tournament i; (c) turn $j + 1$ of tournament i; or (d) the last ($\texttt{DOWN}_2$-)turn $j' \leq j + 1$ of tournament i.*
- *If this is a $\texttt{DOWN}_2$-turn of v (the last turn of this tournament), then u is in either (a) an UP-turn $j' \geq j - 1$ of tournament i; (b) the last ($\texttt{DOWN}_2$-)turn of tournament i; or (c) turn 1 of tournament $i + 1$.*

Given some $U \subseteq V$ and $i, j \in \mathbb{Z}_{>0}$, let $T^U(i, j)$ denote the first time at which every node $v \in U$ satisfies either
(1) v is inactive;
(2) v is in tournament $i' > i$;
(3) v is in the last ($\texttt{DOWN}_2$-)turn of tournament i; or
(4) v is in turn $j' \geq j$ of tournament i.
Note that $T^U(i, j)$ is well defined even if some node $v \in U$ does not reach turn j of tournament i. Employing Observation 1, the delaying states feature guarantees that

$$T^{\{v\}}(i, j + 1) \;\leq\; T^{N(v) \cup \{v\}}(i, j) + 1 \qquad (2)$$

for every $v \in V$ and $i, j \in \mathbb{Z}_{>0}$. Since $T^U(i,j) \leq T^V(i,j)$ for every $U \subseteq V$, we can apply inequality (2) to each node $v \in V$, concluding that

$$T^V(i, j+1) \;\leq\; T^V(i,j) + 1\,,$$

which immediately implies that

$$T^V(i, k+1) \;\leq\; T^V(i,1) + k\,. \tag{3}$$

Moreover, if no node in V goes beyond turn j of tournament i, then

$$T^V(i+1, 1) \;=\; T^V(i, j+1) \;\leq\; T^V(i,j) + 1\,. \tag{4}$$

The virtual graph G_i. Let V_i be the set of nodes for which tournament i exists and let $G_i = (V_i, E_i)$ be the subgraph induced on G by V_i, where $E_i = E \cap (V_i \times V_i)$. Given some node $v \in V_i$, let $N_i(v) = \{u \in V_i \mid (u,v) \in E_i\}$ be the neighborhood of node v in G_i and let $d_i(v) = |N_i(v)|$ be its degree. Note that the graph G_i is virtual in the sense that it is defined solely for the sake of the analysis: we do not assume that there exists some time at which the graph induced by any meaningful subset of the nodes (say, the nodes in tournament i) agrees with G_i.

Given some node $v \in V_i$, let $X^v(i)$ denote the number of `UP`-turns in tournament i of v and recall that the total number of turns in this tournament is at most $X^v(i) + 2$, accounting for the $\texttt{DOWN}_1$ turn in the beginning of the tournament and the $\texttt{DOWN}_2$-turn in its end. The logic of the `UP` states implies that $X^v(i)$ is a Geom(1/2)-random variable, namely, it obeys the geometric distribution with parameter 1/2. The key observation now is that conditioned on G_i, the random variables $X^v(i)$, $v \in V_i$, are independent. Moreover, the graph G_{i+1} is fully determined by the random variables $X^v(i)$, $v \in V_i$. Since the maximum of (at most) n independent Geom(1/2)-random variables is $O(\log n)$ w.h.p., inequalities (3) and (4) yield the following observation.

Observation 2. *For every $i \in \mathbb{Z}_{>0}$, $T^V(i,1)$ is finite w.p. 1 and*

$$T^V(i+1, 1) \;\leq\; T^V(i,1) + O(\log n)$$

w.h.p.

Our protocol is designed so that node v moves to an output state (`WIN` or `LOSE`) in the end of each tournament w.p. > 0. Moreover, the logic of state $\texttt{DOWN}_2$ guarantees that if node v moves to state `WIN` in the end of tournament i, then all its active neighbors move to state `LOSE` in the end of their respective tournaments i. The correctness of our protocol now follows from Observation 2: the protocol reaches an output configuration w.p. 1 and every output configuration reflects an MIS. It remains to bound the run-time of our protocol; the following lemma plays a major role in this task.

Lemma 1. *There exist two constants $0 < p, c < 1$ such that $|E_{i+1}| \leq c|E_i|$ w.p. $\geq p$.*

We will soon prove Lemma 1, but first, let us explain why it suffices for the completion of our analysis. Define the random variable $Y = \min\{i \in \mathbb{Z}_{>0} : |E_i| = 0\}$. Lemma 1 implies that Y is stochastically dominated by a random variable that obeys distribution $O(\log n) + \mathrm{NB}(O(\log n), 1-p)$, namely, a fixed term of $O(\log n)$ plus the negative binomial distribution with parameters $O(\log n)$ and $1-p$, hence $Y = O(\log n)$ in expectation and w.h.p. Since the nodes in $V - V_i$ are all in an output state (and will remain in that state), and since the logic of the `UP` states implies that a degree-0 node in G_i will move to state `WIN` in the end of tournament i (w.p. 1) and thus, will not be included in V_{i+1}, we can employ Observation 2 to conclude that the run-time of our protocol is indeed $O(\log^2 n)$.

The remainder of this section is dedicated to establishing Lemma 1. The proof technique we use for that purpose resembles (a hybrid of) the techniques used in [5] and [36] for the analysis of their MIS algorithms. We say that node $v \in V_i$ is *good* in G_i if

$$|\{u \in N_i(v) \mid d_i(u) \leq d_i(v)\}| \geq d_i(v)/3\,,$$

i.e., if at least third of v's neighbors in G_i have degrees smaller or equal to that of v. The following lemma is established in [5].

Lemma 2 ([5]). *More than half of the edges in E_i are incident on good nodes in G_i.*

Disjoint winning events. Consider some good node v in G_i with $d = d_i(v) > 0$ and let $\widehat{N}_i(v) = \{u \in N_i(v) \mid d_i(u) \leq d\}$. Recall that the definition of a good node implies that $|\widehat{N}_i(v)| \geq d/3$. We say that node $u \in \widehat{N}_i(v)$ *wins* v in tournament i if

$$X^u(i) > \max\left\{X^w(i) \mid w \in N_i(u) \cup \widehat{N}_i(v) - \{u\}\right\}$$

and denote this event by $A_i(u,v)$. The main observation now is that if u wins v in tournament i, then in the end of their respective tournaments i, u moves to state `WIN` and v moves to state `LOSE`. Moreover, the events $A_i(u,v)$ and $A_i(w,v)$ are disjoint for every $u, w \in \widehat{N}_i(v)$, $u \neq w$.

Fix some node $u \in \widehat{N}_i(v)$. Let $u_1, \ldots, u_k$ be the nodes in $N_i(u) \cup \widehat{N}_i(v)$, where $0 < k \leq 2\,d$ by the definition of a good node. Let $B_i(u,v)$ denote the event that the maximum of $\{X^{u_\ell}(i) \mid 1 \leq \ell \leq k\}$ is attained at a single $1 \leq \ell \leq k$. Since $X^{u_1}(i), \ldots, X^{u_k}(i)$ are independent random variables that obey distribution Geom(1/2), it follows that $\mathbb{P}(B_i(u,v)) \geq 2/3$ and therefore,

$$\mathbb{P}\left(A_i(u,v)\right) = \mathbb{P}\left(A_i(u,v) \mid B_i(u,v)\right) \cdot \mathbb{P}\left(B_i(u,v)\right) \geq \frac{1}{k} \cdot \frac{2}{3}\,.$$

Given that v is good in G_i and recalling the disjointness of the $A_i(u,v)$ events, the last inequality implies that

$$\begin{aligned}\mathbb{P}\left(v \notin V_{i+1}\right) \geq\; & \mathbb{P}\left(\bigvee_{u \in \widehat{N}_i(v)} A_i(u,v)\right) \\ =\; & \sum_{u \in \widehat{N}_i(v)} \mathbb{P}\left(A_i(u,v)\right) \geq \frac{d}{3} \cdot \frac{1}{2\,d} \cdot \frac{2}{3} = \frac{1}{9}\,.\end{aligned}$$

Combined with Lemma 2, we conclude that $\mathbb{E}[|E_{i+1}|] < \frac{17}{18}|E_i|$. Lemma 1 now follows by Markov's inequality, thus establishing Theorem 3.

4.2 Node coloring

Given a graph $G = (V, E)$, the *coloring* problem asks for an assignment of colors to the nodes such that no two neighboring nodes have the same color. A coloring using at most k colors is called a *k-coloring.* The smallest number of colors needed to color graph G is referred to as the *chromatic* *number* of G, denoted by $\chi(G)$. In general, $\chi(G)$ is difficult to compute even in a centralized model [15]. As such, the distributed computing community is generally satisfied already with a $(\Delta+1)$-, $O(\Delta)$- or even $\Delta^{O(1)}$-coloring, where $\Delta = \Delta(G)$ is the largest degree in the graph G, with possibly $\Delta(G) \gg \chi(G)$ [21, 32, 45, 12, 29, 35, 13, 43]. As the output of each node under the nFSM model is taken from a predetermined constant size set, we cannot hope to solve these problems for general graphs. Instead, we observe that the following simple nFSM protocol colors any *bounded degree* graph in logarithmic time: As long as node v is still uncolored, it picks an available color c uniformly at random, where initially, the set of available colors is $\{1, \dots, d+1\}$ for some constant $d \geq \Delta$. Then, v proposes color c to its neighbors and colors itself (irrevocably) with c if none of its neighbors proposed c in the previous round.

THEOREM 4. *Given some constant d, there exists an nFSM protocol with bounding parameter $b = 1$ that $(d+1)$-colors any n-node graph satisfying $\Delta \leq d$ in time $O(\log n)$.*

As Δ may grow quickly with n even for relatively simple graph classes, we turn our attention to a natural graph class that features a small chromatic number regardless of Δ: trees. Any tree T has a chromatic number $\chi(T) = 2$. Unfortunately, it is easy to show that in general, the task of 2-coloring trees requires run-time proportional to the diameter of the tree even under the message passing model, and hence cannot be achieved by an efficient distributed algorithm. The situation improves dramatically once 3 colors are allowed; indeed, Cole and Vishkin [21] presented a distributed algorithm that 3-colors directed paths, and in fact, any directed tree (directed in the sense that each node knows the port leading to its unique parent), in time $O(\log^* n)$. Linial [32] showed that this is asymptotically optimal.

Since it is not clear how to represent directed trees in the nFSM model, we focus on undirected trees. A lower bound result of Kothapalli et al. [28] shows that under the *anonymous* (namely, the nodes are not assumed to have unique identifiers) message passing model, 3-coloring undirected trees requires $\Omega(\log n)$ time as long as the size of each message is $O(1)$. We show that this lower bound is tight under the nFSM model (proof deferred to the full version).

THEOREM 5. *There exists an nFSM protocol with bounding parameter $b = 3$ that 3-colors any n-node (undirected) tree in time $O(\log n)$.*

4.3 The square graph

Consider some graph $G = (V, E)$, node $v \in V$, and positive integer k. We define the *k-hop neighborhood* of v in G, denoted by $N^k(v)$, as the set of all nodes $u \in V$, $u \neq v$, at distance at most k from v. The *k^{th} power* of G, denoted by G^k, is the graph obtained from G by extending its edge set so that v is adjacent to all nodes in $N^k(v)$ for every $v \in V$. The second power G^2 of G is called the *square* of G. The proof of Lemma 3 is deferred to the full version.

LEMMA 3. *For every nFSM protocol Π with bounding parameter $b = 1$, there exists an nFSM protocol Π^2 with bounding parameter $b = 2$ such that for every graph G, the execution of Π^2 on G simulates the execution of Π on G^2 with a constant multiplicative run-time overhead.*

A node subset $U \subseteq V$ is a *k-hop independent set* of the graph $G = (V, E)$ if $v \in U$ implies that $u \notin U$ for every $u \in N^k(v) - \{v\}$; the *maximal k-hop independent set (MkIS)* problem asks for a k-hop independent set $U \subseteq V$ which is maximal in the sense that $U' \subseteq V$ is not a k-hop independent set for any $U' \supset U$. Likewise, the *k-hop coloring* problem asks for an assignment of colors to the nodes such that the color of v differs from the color of u for every $u \in N^k(v) - \{v\}$. Cast in this terminology, the MIS and coloring problems are special cases of the MkIS set and k-hop coloring problems, respectively, for $k = 1$. It is shown in [23] that the MkIS and k-hop coloring problems cannot be solved for $k \geq 3$ by a distributed algorithm even under the much stronger anonymous message passing model. In contrast, the $k = 2$ case is resolved positively by plugging Theorems 3 and 4 into Lemma 3.

COROLLARY 1. *Given some constant d, there exists an nFSM protocol with bounding parameter $b = 2$ that computes a 2-hop coloring with (d^2+1) colors for any n-node graph satisfying $\Delta \leq d$ in time $O(\log n)$.*

COROLLARY 2. *There exists an nFSM protocol with bounding parameter $b = 2$ that computes an M2IS for any n-node graph in time $O(\log^2 n)$.*

Corollary 1 essentially provides us with the power to implement independent communication along each edge in bounded degree graphs: Given a 2-hop coloring c, we can append the pair $(c(u), c(v))$ to a message originated at node u whose destination is node $v \in N(u)$. This way, node v can detect that the message was sent by u, whereas any node $w \in N(u) - \{v\}$ can ignore it.

Simulating population protocols. Corollary 2 also has an interesting implication that takes us back to the comparison between the nFSM model and the population protocols model (see Section 1) as it allows us to simulate the rendezvous based communication of any population protocol (in arbitrary interaction graphs). To explain how this is done, we need to introduce the notion of an *oriented matching* consisting of a set $M = \{(x_1, y_1), \dots, (x_k, y_k)\}$ of ordered node pairs satisfying (1) $(x_i, y_i) \in E$ for every $1 \leq i \leq k$; and (2) $(x_i, y_j) \notin E$ for every $1 \leq i, j \leq k$, $i \neq j$. We refer to the nodes $x_1, \dots, x_k$ as *leaders* and to the nodes $y_1, \dots, y_k$ as *subordinates.*

Based on the M2IS protocol promised in Corollary 2, we present in the full version an nFSM protocol that computes an oriented matching M; specifically, upon termination of this protocol, each node knows if it is a leader, a subordinate, or neither. Moreover, for every edge $e \in E$ in each one of its two possible orientations, it holds that M is a singleton $M = \{e\}$ w.p. > 0. By the definition of an oriented matching, each leader x_i (respectively, subordinate y_i) can now safely communicate with its subordinate y_i (resp., leader x_i) without risking interference from other leaders (resp., subordinates). Therefore, we can apply the rule of the simulated population protocol to each (x_i, y_i) pair and update x_i and y_i's states under this simulated protocol accordingly. With an appropriate node delaying mechanism (see, e.g., Section 4.1), this process can be repeated indefinitely, thus yielding a schedule that must be fair (cf. [8]) due to the probabilistic guarantees of M.

LEMMA 4. *The model obtained from nFSM by relaxing the correctness requirement to an eventually converging correctness is at least as strong, in terms of its computational power, as the population protocols model (with the same interaction graph).*

4.4 Maximal matching

Given a graph $G = (V, E)$, an edge subset $M \subseteq E$ is called a *matching* if every node $v \in V$ is incident on at most one edge in M. The *maximal matching (MM)* problem asks for a matching which is maximal in the sense that $M' \subseteq E$ is not a matching for every $M' \supset M$. A message passing MM algorithm with logarithmic run-time was developed by Israeli and Itai [27], whereas the $\Omega(\sqrt{\log n})$ lower bound of [30] applies to the MM problem as well.

We would like to design an nFSM protocol for the MM problem. However, the nFSM model as defined in Section 2 is not expressive enough for this problem: in a complete bipartite graph with n nodes in each side for example, the number of maximal matchings is $n!$, while an nFSM protocol with c output states can only specify $c^{2n} \ll n!$ different (global) outputs. In other words, the nFSM model is geared towards problems whose output is specified by labeling the graph's nodes, whereas the output in the MM problem requires assigning (binary) labels to the graph's edges.

This obstacle is tackled by slightly extending the nFSM model in a manner that enhances it with the power to cope with edge labeling problems such as MM without violating model requirements (M1)–(M4) (see Section 1). To that end, we augment the 8-tuple $\Pi = \langle Q, Q_I, Q_O, \Sigma, \sigma_0, b, \lambda, \delta \rangle$ introduced in Section 2 with a finite *internal alphabet* Γ and with a *port transition function*

$$\eta : Q \times \Gamma \times \Sigma \to 2^{\Gamma} .$$

We also change the function $\lambda : Q \to \Sigma$ to $\lambda : Q \to \Gamma$ and the initial letter $\sigma_0 \in \Sigma$ to $\gamma_0 \in \Gamma$.

The new semantics is as follows. While the communication alphabet Σ is still used for message transmission, each port now contains some letter of the internal alphabet Γ, hence the function λ now maps Q to a query letter in Γ. Given some node $v \in V$ and port $\psi_v(u)$ corresponding to neighbor u of v, the port transition function η takes the current state $q \in Q$ of v, the letter $\gamma \in \Gamma$ currently stored in $\psi_v(u)$, and the new letter $\sigma \in \Sigma$ delivered to v from u, and returns some letter $\gamma' \in \Gamma$ chosen uniformly at random from $\eta(q, \gamma, \sigma) \subseteq \Gamma$; the letter γ' is then stored in $\psi_v(u)$ (replacing γ). This can be viewed as a FSM that controls the letters stored in each one of v's ports (the same FSM for all ports).

Using this *extended nFSM* model, we can now specify edge labels through the ports of the nodes residing in output states. In particular, an MM protocol can specify its output matching M as follows. The protocol designer designates some letter $\gamma_{\text{out}} \in \Gamma$ for the purpose of outputting M. Node $v \in V$ residing in an output state may have at most one port $\Psi_v(u)$ storing the letter γ_{out} in which case it is guaranteed that port $\Psi_u(v)$ also stores γ_{out}, thus marking that $(u, v) \in M$. Note that this cannot be achieved under the (non extended) nFSM model, where, by definition, if $\psi_v(u)$ stores the letter $\sigma \in \Sigma$ at the end of the execution, then $\psi_w(u)$ also stores σ for every $w \in N(u)$. The proof of the following theorem is deferred to the full version.

THEOREM 6. *There exists an extended nFSM protocol with bounding parameter $b = 2$ that computes an MM for any n-node graph in time $O(\log^2 n)$.*

5. CONCLUSIONS

Motivated by networks of sub-microprocessor devices, we introduce the new nFSM model that depicts a network of randomized finite state machines whose communication relies on aggregating the messages from all neighbors according to the one-two-many scheme. Although each individual node in this model is, by design, much weaker than the the nodes under the standard message passing model, we show that the collaborative power of the network's nodes is still sufficiently strong to allow efficient algorithms for some of the most important distributed computing problems.

Given the dynamic nature of many biological cellular networks, it would be very interesting to extend the nFSM model to networks that may undergo failures and/or dynamic insertions. Among the open questions that fascinate us in that regard are: What problems can be solved efficiently in such dynamic scenarios? Is is still possible to locally synchronize the nFSM model in the face of dynamic changes?

6. ACKNOWLEDGMENTS

We would like to thank Jasmin Smula for her help with various parts of this paper.

7. REFERENCES

[1] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. F. Knight, Jr., R. Nagpal, E. Rauch, G. J. Sussman, and R. Weiss. Amorphous computing. *Commun. ACM*, 43(5):74–82, May 2000.

[2] Y. Afek, N. Alon, Z. Bar-Joseph, A. Cornejo, B. Haeupler, and F. Kuhn. Beeping a maximal independent set. In *DISC*, pages 32–50, 2011.

[3] Y. Afek, N. Alon, O. Barad, E. Hornstein, N. Barkai, and Z. Bar-Joseph. A Biological Solution to a Fundamental Distributed Computing Problem. *Science*, 331(6014):183–185, Jan. 2011.

[4] I. F. Akyildiz, J. M. Jornet, and M. Pierobon. Nanonetworks: a new frontier in communications. *Commun. ACM*, 54(11):84–89, Nov. 2011.

[5] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7:567–583, December 1986.

[6] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, pages 235–253, mar 2006.

[7] D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, sep 2008.

[8] J. Aspnes and E. Ruppert. An introduction to population protocols. In B. Garbinato, H. Miranda, and L. Rodrigues, editors, *Middleware for Network Eccentric and Mobile Applications*, pages 97–120. Springer-Verlag, 2009.

[9] B. Awerbuch. Complexity of network synchronization. *J. ACM*, 32(4):804–823, 1985.

[10] B. Awerbuch, B. Patt-Shamir, D. Peleg, and M. E. Saks. Adapting to asynchronous dynamic networks (extended abstract). In *STOC*, pages 557–570, 1992.

[11] B. Awerbuch and D. Peleg. Network synchronization with polylogarithmic overhead. In *FOCS*, pages 514–522, 1990.

[12] L. Barenboim and M. Elkin. Distributed (delta+1)-coloring in linear (in delta) time. In *STOC*, pages 111–120, 2009.

[13] L. Barenboim and M. Elkin. Deterministic distributed vertex coloring in polylogarithmic time. *J. ACM*, 58(5):23, 2011.

[14] L. Barenboim, M. Elkin, S. Pettie, and J. Schneider. The locality of distributed symmetry breaking. *CoRR*, abs/1202.1983, 2012.

[15] M. Bellare, O. Goldreich, and M. Sudan. Free bits, pcps, and nonapproximability-towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998.

[16] Y. Benenson. Biomolecular computing systems: principles, progress and potential. *Nat Rev Genet*, 13(7):455–468, July 2012.

[17] Y. Benenson, T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh, and E. Shapiro. Programmable and autonomous computing machine made of biomolecules. *Nature*, 414(6862):430–434, Nov. 2001.

[18] D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30:323–342, April 1983.

[19] I. Chatzigiannakis and P. G. Spirakis. The dynamics of probabilistic population protocols. In *DISC*, DISC '08, pages 498–499, Berlin, Heidelberg, 2008. Springer-Verlag.

[20] I. Chlamtac and S. Kutten. On Broadcasting in Radio Networks–Problem Analysis and Protocol Design. *Communications, IEEE Transactions on [legacy, pre - 1988]*, 33(12):1240–1246, 1985.

[21] R. Cole and U. Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Inf. Control*, 70(1):32–53, July 1986.

[22] A. Cornejo and F. Kuhn. Deploying wireless networks with beeps. In *DISC*, pages 148–162, 2010.

[23] Y. Emek, J. Seidel, and R. Wattenhofer. Distributed computability: Anonymity, revocability, and randomization. A manuscript.

[24] R. Flury and R. Wattenhofer. Slotted Programming for Sensor Networks. In *IPSN*, April 2010.

[25] M. Gardner. The fantastic combinations of John Conway's new solitaire game 'life'. *Scientific American*, 223(4):120–123, 1970.

[26] P. Gordon. Numerical Cognition Without Words: Evidence from Amazonia. *Science*, 306(5695):496–499, Oct. 2004.

[27] A. Israeli and A. Itai. A fast and simple randomized parallel algorithm for maximal matching. *Inf. Process. Lett.*, 22(2):77–80, 1986.

[28] K. Kothapalli, C. Scheideler, M. Onus, and C. Schindelhauer. Distributed Coloring in $\tilde{O}(\sqrt{\log n})$ Bit Rounds. In *IPDPS*, 2006.

[29] F. Kuhn. Weak graph colorings: distributed algorithms and applications. In *SPAA*, pages 138–144, New York, NY, USA, 2009. ACM.

[30] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! In *PODC*, pages 300–309, 2004.

[31] C. Lenzen and R. Wattenhofer. MIS on trees. In *PODC*, pages 41–48, New York, NY, USA, 2011.

[32] N. Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21:193–201, Feb. 1992.

[33] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15:1036–1055, November 1986.

[34] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1st edition, 1996.

[35] Y. Métivier, J. M. Robson, N. Saheb-Djahromi, and A. Zemmari. About randomised distributed graph colouring and graph partition algorithms. *Inf. Comput.*, 208(11):1296–1304, Nov. 2010.

[36] Y. Métivier, J. M. Robson, N. Saheb-Djahromi, and A. Zemmari. An optimal bit complexity randomised distributed MIS algorithm. *Distributed Computing*, 23(5-6):331–340, Jan. 2011.

[37] O. Michail, I. Chatzigiannakis, and P. G. Spirakis. *New Models for Population Protocols*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2011.

[38] K. Nakamura. Asynchronous cellular automata and their computational ability. *Syst Comput Controls*, 5(5):58–66, 1974.

[39] C. L. Nehaniv. Asynchronous automata networks can emulate any synchronous automata network. *Journal of Algebra*, pages 1–21, Dec. 2003.

[40] D. Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[41] D. Sadava. *Life: The Science of Biology*. Sinauer Associates, 2011.

[42] J. Schneider and R. Wattenhofer. An optimal maximal independent set algorithm for bounded-independence graphs. *Distributed Computing*, 22(5-6):349–361, 2010.

[43] J. Schneider and R. Wattenhofer. Distributed Coloring Depending on the Chromatic Number or the Neighborhood Growth. In *SIROCCO*, June 2011.

[44] J. Suomela. Survey of local algorithms. *To appear in ACM Computing Surveys*, 2012. http://www.cs.helsinki.fi/u/josuomel/doc/local-survey.pdf.

[45] M. Szegedy and S. Vishwanathan. Locality based graph coloring. In *STOC*, pages 201–207, 1993.

[46] L. G. Valiant. Parallel computation. In *7th IBM Symp. on Math. Foundations of Computer Science*, 1982.

[47] J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, IL, USA, 1966.

[48] S. Wolfram. *A new kind of science*. Wolfram Media, Champaign, Illinois, 2002.