New Clocks, Fast Line Formation and Self-Replication
Population Protocols

Leszek Gąsieniec
Department of Computer Science, University of Liverpool, Ashton Street, L69 38X, U.K.

Paul Spirakis
Department of Computer Science, University of Liverpool, Ashton Street, L69 38X, U.K.

Grzegorz Stachowiak
Institute of Computer Science, University of Wrocław, Joliot Curie 15, Wrocław, Poland

February 28, 2022

Abstract

The model of population protocols was developed to study distributed processes based on pairwise interactions between anonymous agents drawn from a large population of size $n$. The interacting pairs of agents are chosen by the random scheduler and their states are adjusted adequately by the predefined transition function which governs the considered process. The state space of agents is fixed (limited to a constant size) and the size $n$ is not known, i.e., not hard-coded in the transition function. We assume that a population protocol starts in the predefined initial configuration of agents’ states representing the input, and it concludes in the final configuration of states representing the output of the considered problem. The sequential time complexity of such protocols refers to the number of interactions required to stabilise the relevant protocol in the final configuration. We also define the parallel time as the sequential time complexity divided by $n$.

In this paper we consider a variant of the standard population protocol model in which agents are allowed to be connected by edges, known as the constructors model. During an interaction between two agents the relevant connecting edge can be formed, maintained or eliminated by the transition function. Since pairs of agents are chosen uniformly at random the status of each edge is updated every $\Theta(n^2)$ interactions on average which coincides with $\Theta(n)$ parallel time. This phenomenon provides a natural lower bound on the time complexity for any non-trivial rigid network construction designed for this variant. This is in contrast with the standard population protocol model in which efficient solutions are expected to operate in $O(poly\log n)$ parallel time.

The contributions of this paper are manifold.

• We propose and analyse a novel type of phase clocks allowing to count parallel time $\Theta(n \log n)$ in the constructors model. This new type of clocks can be also implemented in the standard population protocol model assuming a unique leader is available.

• The new clock enables a nearly optimal $O(n \log n)$ parallel time spanning line construction which improves dramatically on the best previously known $O(n^2)$ parallel time solution.

• We define a probabilistic version of bubble-sort in which random comparisons are allowed only between adjacent numbers in the sequence being sorted. We show that rather surprisingly this probabilistic bubble-sort requires $O(n^2)$ comparisons in expectation, i.e., on the same level as its deterministic counterpart.

• We propose the first self-replication protocol allowing to reproduce a strand (line-segment carrying information) of length $k$ in parallel time $O(n(k + \log n))$. This result is based on the probabilistic bubble-sort argument. This protocol permits also simultaneous replication where $l$ copies of the strand can be obtained in time $O(n(k + \log n) \log l)$.

All protocols in this paper operate with high probability defined as $1 - n^{-\eta}$, for a constant $\eta > 0$.

1 Introduction

The model of population protocols originates from the seminal paper of Angluin et al. [4]. This model provides tools for the formal analysis of pairwise interactions between simple indistinguishable entities referred to as agents. The agents are equipped with limited storage, communication and computation capabilities. When two agents engage in a direct interaction their states are amended according to the predefined transition function which forms an integral part of the population protocol. The weakest possible assumption in population protocols, also adopted in this paper, says that the state space of agents is fixed (limited to a constant size) and the size of the population $n$ is not known, i.e., not hard-coded in the transition function. In the probabilistic variant of population protocols adopted in this paper, in each step the random scheduler selects from the whole population an ordered pair of agents formed...
of the \textit{initiator} and the \textit{responder}, uniformly at random. The lack of symmetry in this pair is a powerful source of random bits often used by population protocols. In this variant, in addition to \textit{state utilisation} one is also interested in the \textit{time complexity} of the proposed solutions. In more recent work on population protocols the focus is on \textit{parallel time} defined as the total number of pairwise interactions (sequential time) leading to the solution divided by the size $n$ of the whole population. For example, a core dissemination tool in population protocols known as \textit{one-way epidemic} \cite{5} distributes simple (e.g., $0/1$) messages to all agents in the population utilising $\Theta(n \log n)$ interactions or equivalently $\Theta(\log n)$ parallel time. The parallel time is meant to reflect on massive parallelism of simultaneous interactions. However, this is not entirely accurate simplification, see \cite{12}. Nevertheless, it provides a good estimation on the locally observed time, i.e., the number of interactions each agent was involved in throughout the computation process.

Unless stated otherwise we assume that any protocol starts in the predefined \textit{initial configuration} with all agents being in the same \textit{initial state}. A population protocol \textit{terminates with success} if the whole population stabilises eventually, i.e., it arrives at and stays indefinitely in the \textit{final configuration} of states representing the desired property of the solution.

### 1.1 Constructors Model

While in the standard population protocol model the population of agents remains unstructured, in the \textit{constructors} model introduced in \cite{22} and adopted in this paper during an interaction between two agents the edge connecting them can be formed, maintained or eliminated by the transition function. In this way the protocol instructs agents how to organize themselves into temporary or more definite network structures. We refer to such distributed and dynamically structured systems based on population protocols as \textit{network constructors} or simply \textit{constructors}, see \cite{22}.

Note that since pairs of agents are chosen uniformly at random the status of each edge is updated every $\Theta(n^2)$ interactions on average which coincides with $\Theta(n)$ parallel time. This phenomenon provides a natural lower bound on the time complexity of non-trivial network construction processes, see \cite{22}. However, some form of relaxation of expectations may allow to surpass this bottleneck as shown in the construction of \textit{almost balanced} trees requiring $o(n^2)$ interactions \cite{11}, where network edges are consistently drawn from a large group of available connections and most of the edges are never examined during the computation process.

**Model specificity** Whenever possible we will be using capital letters to denote states of the agents. In order to \textit{accommodate} edge connections the transition function governs the relation between triplets of the following type:

$$P + Q + S \rightarrow P' + Q' + S'.$$

The first two terms on both sides of the rule refer to the states $P$ and $Q$ of the initiator and the responder (respectively) before and $P'$ and $Q'$ after the interaction. The third term $S$ before and $S'$ after the interaction is a binary flag indicating the status of the connection between the two agents, where the edge availability is indicated by 1 and by 0 the lack of it. Note that the states of agents are often more complex as they are a combination of a (constant) number of attributes. Such states are represented as tuples. In such compound states we use vector representation $<, >$ with acute brackets where individual attributes are separated by commas.

**Probabilistic guarantees** Let $\eta$ be a universal positive constant referring to the reliability of our protocols. We say that an event occurs with \textit{negligible} probability if it occurs with probability at most $n^{-\eta}$, and an event occurs with high probability (whp) if it occurs with probability at least $1 - n^{-\eta}$. This estimate is of an asymptotic nature, i.e., we assume $n$ is large enough to validate the results. Similarly, we say that an algorithm succeeds with high probability if it succeeds with probability at least $1 - n^{-\eta}$. When we refer to the probability of failure $p$ different to $n^{-\eta}$, we say \textit{directly with probability at least} $1 - p$. Our protocols make heavy use of Chernoff bounds and the new tail bounds for sums of geometric random variables derived in \cite{18}. We refer to these new bounds as Chernoff-Janson bounds.

### 1.2 Our results and their significance

The model of population protocols gained considerably in popularity in the last 15 years. We study here several central problems in distributed computing by focusing on the adopted variant of population protocols. These include the concept of \textit{phase clocks}, a distributed synchronisation tool with good space and accuracy guarantees. The first study of leader based $O(1)$ space phase clocks can be found in the seminal paper by Angluin et al. in \cite{5}. Further extensions including the junta based clock and nested clocks counting any $\Theta(\text{poly log } n)$ parallel time were analysed in \cite{16}. In a very recent work \cite{14} Doty et al. study constant resolution phase clocks utilising $O(\log n)$ states as the main engine in the optimal \textit{majority} computation protocols. In this work we propose and analyse a new phase clock based on a matching allowing to count $\Theta(n \log n)$ parallel time. This is the first clock able to confirm the conclusion of the slow leader election protocol based on direct duels between the (remaining) leader candidates. We also propose an alternative edge-less version of this clock based on the computed leader. This edge-less clock supports a significantly improved and nearly optimal $O(n \log n)$ parallel time line formation protocol.

We also consider a probabilistic variant of the classical bubble-sort algorithm, in which comparisons between any two consecutive positions in the sequence are chosen uniformly at random. We show that rather surprisingly
this variant is on par with its deterministic counterpart as it requires only \( \Theta(n^2) \) random comparisons whp. While this new result is of an independent algorithmic interest, together with the new edge-less clock they power a strand (line-segment carrying information) self-replication protocol studied towards the end of this paper.

In a wider context, self-replication is a property of a dynamical system which allows reproduction. Such systems are of increasing interest in biology, e.g., in the context of how life could have begun on Earth \cite{21}, but also in computational chemistry \cite{23}, robotics \cite{19} and other fields. In our case a larger chunk of information (well beyond the limited state capacity) is stored collectively in a strand (line-segment) of agents. Such a strand may represent a pattern in string matching or an instruction forming a part of more complex distributed process. In such cases the replication mechanism facilitates an improved accessibility to this information. We propose the first strand self-replication protocol allowing to reproduce a strand (line-segment containing information) of size \( k \) in parallel time \( O(n(k + \log n)) \). This protocol permits simultaneous replication, where \( l \) copies of a strand can be generated in parallel time \( O(n(k + \log n) \log l) \).

### 1.3 Related work

One of the main tools used in this paper refers to the central problem of leader election, with the final configuration comprising a single agent in the leader state and all other agents in the follower state. The leader election problem received in recent years greater attention in the context of population protocols. In particular, the results in \cite{19} \cite{13} laid down the foundation for the proof that leader election cannot be solved in a sublinear time with agents utilising a fixed number of states \cite{15}. In further work \cite{5}, Alistarh and Gelashvili studied the relevant upper bound, where they proposed a new leader election protocol stabilising in time \( O(\log^3 n) \) assuming \( O(\log^3 n) \) states per agent.

In a more recent work Alistarh et al. \cite{1} considered more general trade-offs between the number of states used by the agents and the time complexity of stabilisation. In particular, the authors delivered a separation argument distinguishing between slowly stabilising population protocols which utilise \( o(\log n) \) states and rapidly stabilising protocols relying on \( O(\log n) \) states per agent. This result coincides with another fundamental result by Chatziigiannakis et al. \cite{9} stating that population protocols utilizing \( o(\log n) \) states are limited to semi-linear predicates, while the availability of \( O(n) \) states (permitting unique identifiers) admits computation of more general symmetric predicates. Further developments include also a protocol which elects the leader in time \( O(\log^2 n) \) w.h.p. and in expectation utilizing \( O(\log^2 n) \) states \cite{8}. The number of states was later reduced to \( O(\log n) \) by Alistarh et al. in \cite{2} and by Berenbrink et al. in \cite{7} through the application of two types of synthetic coins.

In more recent work Gasieniec and Stachowiak reduce memory utilisation to \( O(\log \log n) \) while preserving the time complexity \( O(\log^2 n) \) whp \cite{16}. The high probability can be traded for faster leader election in the expected parallel time \( O(n \log n \log \log n) \), see \cite{17}. This upper bound was recently reduced to the optimal expected time \( O(\log n) \) by Berenbrink et al. in \cite{6}. One of the main open problems in the area is to establish whether one can elect a single leader in time \( o(\log^2 n) \) whp while preserving the optimal number of states \( O(\log \log n) \).

### 2 Two phase clocks and leader election

In order to compute the unique leader and confirm its computation we execute two protocols simultaneously. Namely, the slow leader election protocol which concludes in parallel time \( O(n \log n) \) whp, and the new (introduced in this section) matching based phase clock which counts parallel time \( \Theta(n \log n) \) whp. The conclusion of leader election is confirmed via one-way epidemic when the final state (in this clock) is reached by any agent. Further, this leader is utilised in the leader based (edge-less) clock in nearly optimal computation of the line containing all agents, see Section \cite{4} and in self-replication of strands of information, see Section \cite{5}.

The transition rules for governing the slow leader election and the new clocks follow.

#### 2.1 Slow leader election

In the initial configuration all agents are in state \( L \) and the leader election protocol is driven by a single rule:

\[
L + L \rightarrow L + F,
\]

where \( L \) represents a leader candidate, and \( F \) stands for a follower (or a free) agent. It is well known that this leader election protocol operates in parallel time \( O(n \log n) \) whp.

#### 2.2 Matching based phase clock

The proposed matching based clock assumes the constructors model in which the transition function recognises whether two interacting agents are connected by an edge or not, indicated by 1 or 0, respectively. The agents begin in the predefined state \( \langle \text{start} \rangle \). When two agents in state \( \langle \text{start} \rangle \) interact they get connected and they enter the counting stage with their counters set to \( \langle 0 \rangle \). Eventually these counters reach the maximum (exit) value \( \text{max} \). Note that the values of the counters can either go up or down, depending on the rule used during the relevant
interaction. Note also that the number of agents taking part in the counting process is always even as they enter and leave this process in pairs. The counting stage sub-protocol guarantees that the counters of all agents which enter this stage reach value $\text{max}$ in time $\Theta(n \log n)$, see Theorem 1. And during the next interaction between the two connected agents in state $<\text{max}>$ the connection is dropped and the states are updated to $<\text{end}>$ indicating the exit from the counting stage.

The rules of the transition function used in the counting stage are as follows:

**Initialisation**

$$<\text{start}> + <\text{start}> + 0 \rightarrow <0> + <0> + 1$$

**Timid counting**

- For all connected $i \leq j$ and $i < \text{max}$
  $$<i> + <j> + 1 \rightarrow <i+1> + <i+1> + 1$$

- For all disconnected $i < j$
  $$<i> + <j> + 0 \rightarrow <i> + <i+1> + 0$$

**Maximum level epidemic**

$$<\text{max}> + <i> + 0 \rightarrow <\text{max}> + <\text{max}> + 0$$

**Conclude and disconnect**

$$<\text{max}> + <\text{max}> + 1 \rightarrow <\text{end}> + <\text{end}> + 0$$

$$<\text{start}> + <\text{end}> + 0 \rightarrow <\text{end}> + <\text{end}> + 0$$

In the next subsection we discuss the rules of an alternative phase clock in which instead of a matching the agents use virtual edges "connecting" them with the computed leader.

### 2.3 Leader based (edge-less) phase clock

We allocate separate constant memory to host the states of the leader based clock. This allows to run the two clocks simultaneously and independently. The followers in the leader based clock start with the counters set to $<0>$, and $L$ refers to the leader state. Note that state $<0>$ is initiated for the leader based clock as soon as the agent reaches state $<\text{max}>$ or $<\text{end}>$ in the matching based clock. Below we present the timid counting rules which now refer to the interactions with the leader $L$ along the virtual connections.

**Timid counting**

- Leader interactions, for $i < \text{max}$
  $$<i> + L \rightarrow <i+1> + L$$

- Non-leader interactions, for $i < j$
  $$<i> + <j> \rightarrow <i> + <i+1>$$

One can show that the two clocks have the same asymptotic time performance, see Section 3 for the relevant detail. Note also that the leader based clock can be used independently from any edge dependent process executed in the population simultaneously.

### 2.4 Periodic leader based (edge-less) clock

One can expand functionality of the leader based clock to pace a series of consecutive rounds of a more complex process, with each round operating in parallel time $\Theta(n \log n)$. The extension is assumed to work in rounds formed of three consecutive stages 0, 1 and 2, where each stage is associated with a single execution (full turn) of the leader based clock. The conclusion of each stage is announced with the help of one-way epidemic in parallel time $O(\log n)$ whp. And when this happens all agents which received the announcement proceed to the next stage. This means that after at most $O(\log n)$ time delay (caused by the time of the epidemic) all agents will run the clock in the same stage whp. Note also that while the signal to start the next stage remains in the population throughout the whole stage, it will be wiped out whp by the signal announcing the beginning of the stage that follows. And since we have 3 stages during each round the synchronisation of agents is guarantied whp.
3 The clocks’ analysis

In this section we provide the time and the probabilistic guaranties for the two phase clocks introduced in Section 2. We first analyse the matching based clock and later extend the reasoning to the leader based (edge-less) clock. We prove the following theorem towards the end of this section.

**Theorem 1.** In either of the two clocks state <max> is reached by any agent in parallel time $\Theta(n \log n)$ whp.

When the matching based clock starts working, it forms a matching consisting of unmatched pairs of agents. We first formulate two lemmas describing how fast this is done.

**Lemma 1.** All edges of the matching are formed in the expected parallel time $\Theta(n)$ and whp $O(n \log n)$.

*Proof.* The probability of an interaction forming edge $i + 1$ when $i$ edges are already formed is $\frac{n(n-1)}{n(2i)(n-2i-1)}$. Thus the number of interactions separating formation of edges $i$ and $i + 1$ has geometric distribution with the expected value $\frac{n(n-1)}{n(2i)(n-2i-1)}$. Thus the expected number of interactions to form all edges is $\sum_{i=1}^{n/2-1} \frac{n(n-1)}{n(2i)(n-2i-1)}$, which is $\Theta(n^2)$.

A sufficient condition to form all the edges is that all possible $\binom{n}{2}$ pairs of agents are generated by the random scheduler. The probability of not choosing a fixed pair in first $O(cn \log n)$ interactions is $(1 - 1/\binom{n}{2})^{cn^2 \log n}$, which is negligible for $c$ big enough. Thus all edges are formed after parallel time $O(n \log n)$ whp. □

The following lemma refers to early interactions in the matching based clock.

**Lemma 2.** After time $0.51n$ at least $\frac{n}{2}$ agents contribute to already formed edges whp.

*Proof.* Assume that so far exactly $i$ edges are formed. The probability that during an interaction edge $i + 1$ is formed is $\frac{n(n-1)}{(n-2i)(n-2i-1)}$. So the expected number of interactions $T_i$ of forming edge $i + 1$ is $\frac{n(n-1)}{(n-2i)(n-2i-1)}$, and in turn the expected number of interactions $T$ of forming first $n/4$ edges satisfies

$$T = T_0 + \cdots + T_{n/4} = \sum_{i=0}^{n/4} \frac{n(n-1)}{(n-2i)(n-2i-1)} \sim n \int_0^{1/4} \frac{dx}{(1-2x)^2} = \frac{n}{2}.$$ 

We can estimate the probability that $T$ exceeds $0.51n$ using Chernoff-Janson bound (Thm.2.1) proving that it is negligible. In this case we can substitute (for $n$ large enough)

$$p_* = \frac{1}{4}, \quad M \sim \frac{n}{2}, \quad \text{and} \quad \lambda = \frac{0.51}{0.5} > 1.$$ 

Thus we get that $T \geq 0.51n$ wp less than $e^{-p_*M(\lambda - 1 - \ln(\lambda))} = e^{-\frac{1}{4}n(\lambda - 1 - \ln(\lambda))}$, i.e., with negligible probability. □

As soon as the edges are formed communication along them begins. In order to analyze this process we define the edge collector problem in which one is asked to collect (draw) all edges of a given matching $M$ of cardinality $n' > \frac{n}{4}$. This process concludes when the random scheduler generates interactions along all edges of the matching. Note that for a technical reason we do not assume that a maximal matching is collected but only some matching that is formed before edge collection is complete. As we indicated in Lemma 2, this matching has more than $\frac{n}{4}$ edges whp.

**Lemma 3.** For any cardinality $n' \in [n/4, n/2]$, the parallel time complexity of the edge collector problem is $O(n \log n)$ whp. In addition, the time needed to collect the last $0.05 \cdot n$ edges (of the matching) is at least $0.4 \cdot n \ln n$ whp.

*Proof.* The probability of collecting an edge in an interaction, when $i$ edges are still missing is $p_i = \frac{2i}{n(n-1)}$. The number of interactions needed to collect this edge is a random variable $X_i$ which has a geometric distribution with the average $\frac{n(n-1)}{2i}$. When $k$ edges are still to be collected, the expected number of interactions to collect extra $k - l$ edges is

$$\sum_{i=l}^{k} \frac{n(n-1)}{2i} = \frac{n(n-1)}{2} \left( H_k - H_l \right) \sim \frac{n(n-1)}{2} \ln \frac{k}{7}. $$

Using the upper bound of lower tail (Theorem 3.1) of Chernoff-Janson bounds we show that this number of interactions is at least $0.4n \ln n$ whp, for $k = 0.05n$ and $l = n^{0.1}$. And indeed, for $n$ large enough one can adopt

$$p_* = p_l = \frac{2n^{0.1}}{n(n-1)}, \quad M \sim \frac{n(n-1)}{2} \ln(0.05n^{0.9}) > 0.44n(n-1) \ln n, \quad \text{and} \quad \lambda = \frac{0.4}{0.44} < 1.$$ 

This way we get that the number of interactions smaller than $0.4n \ln n$ with probability smaller than $e^{-p_*M(\lambda - 1 - \ln(\lambda))} \leq e^{-0.88n^{0.72} \ln n(\lambda - 1 - \ln(\lambda))}$, i.e., with negligible probability.

The collection of edges concludes as soon as the end points forming each edge interact with one another. The probability of a missing interaction along some edge in the first $cn \log n$ interactions is $(1 - 1/\binom{n}{2})^{cn^2 \log n}$, which is negligible for $c$ large enough. Thus the edge collection concludes in parallel time $O(n \log n)$ whp. □
In our clock protocol the value of parameter \( d > 0 \) depends on the constant \( \eta \) with respect to the high probability guarantees. We prove the existence of this parameter \( d \), for any \( \eta' = \eta + 3 \).

**Lemma 4.** In a time window of size \( n^a \), for \( 0 < a < 1 \), any edge in the matching is used in at most \( d \) interactions wp.

**Proof.** By Union bound the probability that an edge is a subject to at least \( d \) interactions in time \( n^a \) does not exceed

\[
\left( \frac{n^{1+a}}{d} \right) \left( \frac{2}{n(n-1)} \right)^d \leq \left( \frac{2n^a}{n-1} \right)^d.
\]

and this value is smaller than \( n^{-\eta'} \) is for \( d \) large enough.

**Lemma 5.** In a window of size \( n^a \), for \( 0.1 < a < 1 \), there are \( \leq 2.1n^a \) interactions along edges of the matching wp.

**Proof.** The probability that a given interaction is an edge interaction is \( \frac{2n^a}{n(n-1)} \). Thus in the window of size \( n^a \), there are expected \( 2n^a \frac{n}{n-1} \) edge interactions. By Chernoff bound the number of edge interactions is at most \( 2.1n^a \) wp.

For the clarity of the presentation, depending on the context we will use notions of counters and levels interchangeably.

**Lemma 6.** Let \( k \) be an integer where \( k < \max -d-2 \). There exists a constant \( c \), s.t., during time period \((0.51, cn \log n)\) presence of any agent on level \( l < k \) guarantees wp a linear subpopulation of agents of size at least \( 0.1n \) on levels \( j \leq k \). Also during this period no agent reaches level max wp.

**Proof.** We prove the validity of the lemma whp, i.e., with probability at least \( (wp) \). The proof is done by induction on parallel time \( t \). First we show that in the initial period \([0.51, n^{0.2}]\) the thesis of the lemma holds wp \( 1 - 10n \cdot n^{-\eta} \), where \( t = n^{0.2} \). Later we prove that until time \( t \) the thesis of the lemma holds wp \( 1 - 10tn \cdot n^{-\eta} \). Note that this guaranties that the thesis holds whp, i.e., wp \( 1 - n^{-\eta} \), for \( t = O(n \log n) \). Assume that all events in the theorem of the lemma hold before time \( t \). We prove that if the thesis of the lemma holds before time \( t \), then it also holds in time wp \( 1 - 10n^{-\eta} \). By the inductive hypothesis before time \( t \) or equivalently until time \( t = \frac{1}{n} \) the thesis of the lemma holds wp \( 1 - 10(t - \frac{1}{n})n \cdot n^{-\eta} \). In return we get that until time \( t \) the thesis of the lemma holds wp \( 1 - 10tn \cdot n^{-\eta} \).

We first prove the base step of induction. As we proved in Lemma [5] during the initial time \( 0.51 \) at least \( n/2 \) agents enter the clock with state \((0)\) wp \( 1 - n^{-\eta} \). Some of these agents could also relocate to higher levels. By Lemma [5] applied to the initial time period \( n^{0.2} \) there are at most \( 2.1n \) of the latter wp. \( 1 - n^{-\eta} \). Thus in time interval \([0.51, n^{0.2}]\) level \( 0 \) is the host of at least \( 0.5n - 2.1n^{0.2} \) \( > 0.4n \) agents constantly residing at this level wp \( 1 - 2n^{-\eta} \). Also, by Lemma [4] no agent reaches level max wp \( 1 - n^{-\eta} \), and until time \( t = n^{0.2} \) the lemma holds wp \( 1 - 10tn \cdot n^{-\eta} \).

Now we prove the inductive step. We observe first that during period \([t', t]\), where \( t' = n^{-a}, t = n^{-0.01} \), all agents which entered the clock are at least once on level \( l \leq k + 1 \) wp \( 1 - n^{-\eta} \). And indeed during this period an agent avoids interactions with agents on levels \( j \leq k \) wp at most

\[
(1 - 0.1/n)^{0.1} < e^{0.1n^{0.1}}
\]

Because of this and Lemma [4] during this period, any agent which entered the clock does not elevate to levels higher than \( k + 1 + d \) wp \( 1 - 2n^{-\eta} \). Therefore no agent reaches level max during period \([t', t]\) wp \( 1 - 2n^{-\eta} \).

In order to prove the first thesis of the lemma we consider two cases.

**Case 1:** in this case in time \( t' \) there are at least \( 0.11 \) agents on levels not exceeding \( k \). Since by Lemma [5] in time period \([t', t]\) at most \( 2.1n^{0.1} \) such agents can increase their level wp \( 1 - n^{-\eta} \). And in turn, in time \( t \) there are at least \( 0.01n > 0.11 \) \( - 2.1n^{0.1} \) agents on levels \( j \leq k \).

**Case 2:** in this case in time \( t' \) the number of agents on levels at most \( k \) is between \( 0.1n \) and \( 0.11n \) and the number of agents on levels below \( k \) is at least \( 3n^{0.1} \). Let \( Y \) be the set of agents belonging to the levels above \( k \) in time \( t' \). By Lemma [7] the probability that in time window \([t', t]\) the number of agents below level \( k \) drops below \( 0.9n^{0.1} \) is negligible, i.e., at most \( n^{-\eta} \). Consider any set \( X \) with \( 0.9n^{0.1} \) agents residing at levels smaller than \( k \) and estimate how many agents from set \( Y \) interact with them. For as long as \( 0.38 \) agents from \( Y \) do not interact with \( X \), the probability of interaction between an unused (not in contact with agents in \( X \)) agent in \( Y \) and some agent in \( X \) is at least \( 0.68n^{-0.9} \). Any such interaction increases the number of agents on levels not exceeding \( k \). Consider a sequence of \( n^{1.1} \) zeros and ones in which position \( \iota \) is one \( (1) \) if and only if either

- interaction \( \iota \) is between an unused agent in \( Y \) with an agent in \( X \) if there are more than \( 0.38n \) unused agents in \( Y \),
- if this number is smaller than \( 0.38n \) value 1 is drawn with a fixed probability \( 0.68n^{-0.9} \).
By Chernoff bound the probability that this sequence has less than $0.6n^{0.2}$ ones is negligible, i.e., at most $n^{-\eta'}$. Since $0.12n < 0.11n + 0.6n^{0.2}$ this sequence has less than $0.6n^{0.2}$ ones only when the number of agents elevated to levels not exceeding $k$ is smaller than $0.6n^{0.2}$. Also by Lemma 5 during period $[t',t]$ at most $2.1n^{0.1}$ others may increase their level beyond $k$ wp $1 - n^{-\eta'}$. So in Case 2 the number of agents on levels not exceeding $k$ increases during period $[t',t]$ by at least $0.6n^{0.2} - 2.1n^{0.1}$.

**Case 3:** assume that in time $t'$ the number of agents on levels $j \leq k$ is between $0.1n$ and $0.11n$ and also the number of agents on levels below $k$ is smaller than $3n^{0.1}$. The probability of an interaction between one of such agents and an agent in set $Y$ of agents above level $k$ is at most $6n^{-0.9}$. Any such interaction increases the number of agents on levels not exceeding $k$. By Chernoff bound the probability that this number of interactions exceeds $7n^{0.2}$ in $[t',t]$ is negligible, i.e., at most $n^{-\eta'}$. Thus in Case 3 the probability that the number of agents on levels at most $k$ exceeds $0.12n > 0.11n + 7n^{0.2}$ is negligible, i.e., at most $n^{-\eta'}$.

We now formulate Claim 1 that upperbounds the number of agents leaving levels $j < k$ and Claim 2 that bounds from below the number of agents joining these levels in Case 3. Because wp $1 - 6n^{-\eta'}$ the levels $j \leq k$ gain agents as a result of these two processes. This will conclude the proof.

**Claim 1:** In Case 3 during period $[t',t]$ there are at most $0.26n^{0.1}$ agents located at levels $j \leq k$ which increment their level wp $1 - 4n^{-\eta'}$.

And indeed, for as long as there are at most $0.12n$ agents on levels not greater than $k$, the probability that such agent interacts as the initiator with a clock agent is at most $0.12/n$. Such an interaction increments the level of this clock agent with probability at most $0.12/n$. We prove that the probability of at least $0.13n^{0.1}$ such incrementations is negligible, i.e., at most $4n^{-\eta'}$. Consider a sequence of $n^{1.1}$ zeros and ones in which position $\iota$ is one if and only if either

- interaction $\iota$ increments initiator’s level and there are at most $0.12n$ agents on levels not greater than $k$
- if this number is greater than $0.38n$ value $1$ is drawn with a fixed probability $0.12/n$.

By Chernoff bound this sequence has less than $0.13n^{0.1}$ ones (1s) wp $1 - n^{-\eta'}$. On the other hand we have at most $0.12n$ agents on levels at most $k$ wp $1 - n^{-\eta'}$. Thus wp $1 - 2n^{-\eta'}$ at most $0.13n^{0.1}$ agents on levels not exceeding $k$ can increment their levels in $[t',t]$ acting as initiators. Analogously, we can prove that wp $1 - 2n^{-\eta'}$ at most $0.13n^{0.1}$ agents on levels not exceeding $k$ can increment their levels in $[t',t]$ acting as responders. So altogether at most $0.26n^{0.1}$ agents on levels $j \leq k$ increment their levels during period $[t',t]$ wp $1 - 4n^{-\eta'}$.

**Claim 2:** In Case 3 during period $[t',t]$ there are at least $0.75n^{0.1}$ interactions between agents on levels $i < k$ and those residing on levels higher than $k$ wp $1 - 2n^{-\eta'}$.

For as long as there are at most $0.12n$ agents on levels at most $k$, at least $0.38n = n/2 - 0.12n$ agents are on levels higher than $k$. The probability of interaction between such agents and an agent on level $i < k$ is at least $0.76/n = 2 \cdot 0.38/n$. Any such an interaction increases the number of agents on levels not exceeding $k$. Consider a sequence of $n^{1.1}$ zeros and ones in which position $\iota$ is one if and only if either

- there are at most $0.12n$ agents on levels not greater than $k$ and interaction $\iota$ increases the number of such agents
- the number of agents on levels up to $k$ is greater than $0.12n$ and value $1$ is drawn with a fixed probability $0.76/n$.

By Chernoff bound this sequence has more than $0.75n^{0.1}$ ones (1s) wp $1 - n^{-\eta'}$. On the other hand we have at most $0.12n$ agents on levels at most $k$ wp $1 - n^{-\eta'}$. Thus wp $1 - 2n^{-\eta'}$ at least $0.75n^{0.1}$ agents on levels exceeding $k$ can reduce their levels to at most level $k$ during period $[t',t]$ while acting as initiators.

Because of both Claims 1 and 2 after time period $[t',t]$ there are at least $0.51n^{0.1} = 0.75n^{0.1} - 0.24n^{0.1}$ more agents on levels $j \leq k$ than in time $t'$. This proves that in time $t$ there are at least $0.1n$ agents on levels $j \leq k$.

**Lemma 7.** The time in which the first agent achieves level max is greater than $(\max - d - 2) \cdot 0.4n \ln n$ wp.

**Proof.** Let $t_k$ be the time when for the first time there are no agents available at levels lower than $k$. By Lemma 6 during period $[0.51, t_k]$, there are at least $0.1n$ agents on level $k$ or lower. Let $n_k \in [n/4, n/2]$ be the number of edges in time $t_k$. Thus between time $t_k$ and $t_k + 1$ at least $0.1n$ agents have to increment their levels to $k + 1$. This is done by collecting (interacting via) edges adjacent to them. By Lemma 5 this takes time at least $0.4n \ln n$. This process has to be repeated for $\max - d - 2$ levels when no agents reach state (max) wp.

**Lemma 8.** The first agent moves to level max in time $O(n \log n)$ wp.

**Proof.** The total time to initiate $[n/2]$ edges is $O(n \log n)$ wp by Lemma 1. If the first agent achieves level max earlier the lemma remains true. If this is not the case, the time $O(n \log n)$ is determined by collection of all $[n/2]$ edges which needs to be repeated max times resulting in the total time $O(n \log n)$.
Now we are ready to prove Theorem 1. The thesis for matching based clock follows directly from Lemmas 7 and 8. The thesis for the leader based clock can be proved by a sequence of lemmas almost identical to Lemmas 6, 7 and 8. In the analog of Lemma 6 we can take \( n - 2 \) followers instead of \( n' \) edges. This is because Lemma 1 assures that the time counted by the matching based clock is long enough to form all edges whp. Note that \( n - 2 \) agents are initiated at level 0 of the leader based clock in time \( O(\log n) \) whp by the epidemic resulting in dismantling of the matching based clock. And in turn we can use the initial time \( O(n \log n) \) instead of 0.51 in the analog of Lemma 6.

4 Fast line formation

**Line formation** We define and analyse a new optimal line formation protocol which operates in time \( \Theta(n \log n) \) whp. while utilising a constant number of extra states (not mixed with other protocols including clocks). The protocol is preceded by leader election confirmed by the matching based clock. And when this happens, the periodic leader based clock starts running together with the following line formation protocol based on two main rules defined below.

**Form head and tail**

\[
L + F + 0 \rightarrow H + T + 1
\]

This rule creates the initial head in state \( H \) and the tail in state \( T \) of the newly formed line. Note that since the line formation process uses separate memory the leader in the leader based clock remains in the leadership state, i.e., it is the head state \( H \) is used solely in the line formation protocol.

**Extend the line**

\[
H + F + 0 \rightarrow R + H + 1
\]

This rule extends the current line by addition of an extra agent from the head end of the line. The state \( R \) indicates that the agent is in the line between the head and the tail.

**Theorem 2.** The line formation protocol operates in time \( O(n \log n) \) whp.

**Proof.** The probability of an interaction adding agent \( i + 1 \) to the line when \( i \) agents are already present is \( \frac{2}{n(n-1)} \). So the number of interactions to add agent \( i + 1 \) has geometric distribution with the expected value \( \frac{n(n-1)}{2(n-1)} \). Thus the expected time of forming the line is

\[
\frac{1}{n} \sum_{i=1}^{n} \frac{n(n-1)}{2(n-i)} = \frac{n}{2} \sum_{i=1}^{n} \frac{1}{i} \sim \frac{n \ln n}{2}.
\]

By Chernoff-Janson bound this time is \( O(n \log n) \) whp.

In order to make the line formation protocol always correct we need some backup rules for the unlikely case of desynchronisation when two or more leaders survive to the line formation stage. In such case we need to continue leader elimination.

\[
L + L + 0 \rightarrow L + F + 0
\]

Also when a leader meets already formed head.

\[
L + H + 0 \rightarrow F + H + 0
\]

Finally we have to dismantle excessive lines if two or more lines are formed. This is done using extra state \( D \) which dismantles the line edge by edge starting from the head.

\[
H + H + 0 \rightarrow H + D + 0
\]

\[
D + R + 1 \rightarrow F + D + 0
\]

\[
D + T + 1 \rightarrow F + F + 0
\]

5 Probabilistic bubble-sort

Let array \( A[0..n-1] \) contain an arbitrary sequence of \( n \) numbers. In the probabilistic bubble-sort during each comparison step an index \( j \in \{0,\ldots,n-2\} \) is chosen uniformly at random, and if \( A[j] > A[j+1] \) these two values are swapped in \( A \). We show that the expected number of comparisons required to sort all numbers in \( A \) (in the increasing order) is \( \Theta(n^2) \) whp.

In order to prove this result we first remind the reader that any sorting procedure based on fixing local inversions requires \( \Omega(n^2) \) comparisons. In order to prove the upper bound we utilise the classical zero-one principle stating that if a (probabilistic) sorting network sorts correctly all sequences of zeros and ones, it also sorts an arbitrary sequence
of numbers of the same length. More precisely, if we want to prove that a given sequence \(X\) of \(n\) numbers will be sorted we have to consider only \(n+1\) zero-one sequences obtained by replacing \(k\) largest elements of \(X\) by ones and the remaining elements by zeros, for any \(k = 0, \ldots, n\), see \cite{20}. Thus it is enough to prove that the probabilistic bubble-sort utilises \(O(n^2)\) comparisons to sort whp any zero-one sequence of length \(n\), and later use the union bound to extend this result to any sequence of numbers, also whp.

**Theorem 3.** The probabilistic bubble-sort utilises \(4(n - 1)(n \ln 2 + \eta \ln n)\) comparisons whp to sort any zero-one sequence of size \(n\).

Let \(k\) be the number of ones in a zero-one sequence represented by \(A\). We define a configuration \(C\) as the subset of all positions in \(A\) at which ones are situated, where \(|C| = k\). The probabilistic bubble-sort starts in the initial configuration (based on the original zero-one sequence) and thanks to the conditional swaps progresses through consecutive configurations including the final one in which all zeros precede \(k\) ones. For any configuration \(C\), we define a potential function \(P(C) = \sum_{i \in C} P[i]\) with a non-negative integer value, where

\[
P[i] = 2^{n-k+2l-i} - 2^l, \quad \text{for } l = |C \cap \{0, \ldots, i-1\}|
\]

Note that the value of this potential is zero for all \(i\) if and only if the sequence is sorted. Thus \(P(C) = 0\) for a sorted sequence \(C\). Also, when all ones precede all zeros, the potential \(P(C)\) is the highest possible. One can notice that always \(P(C) < 2^n\).

We prove the following lemma.

**Lemma 9.** Let \(C\) be an arbitrary configuration in \(A\) and \(EP(C')\) be the expected potential of the next configuration \(C'\) in the probabilistic bubble-sort. The following inequality holds.

\[
EP(C') \leq \left(1 - \frac{1}{4(n-1)}\right) P(C).
\]

**Proof.** We split configuration \(C\) into disjoint blocks of indices \(B_1, B_2, \ldots\), each corresponding to a solid run of ones. For any block \(B = \{x, \ldots, y\}\) we define a potential \(P(B) = \sum_{i=x}^{y} P[i]\). In the subsequent configuration \(C'\), let \(EP(B')\) be the expected potential of \(B' \subseteq C'\) based on the ones originating from \(B\) in the preceding configuration \(C\). We show that

\[
EP(B') \leq \left(1 - \frac{1}{4(n-1)}\right) P(B).
\]

Let \(l = |C \cap \{0, \ldots, y-1\}|\). We have

\[
P(B) = \sum_{i=x}^{y} P[i] = \sum_{i=x}^{y} 2^{n-k+2(l+i-y)-i} - \sum_{i=x}^{y} 2^{l+i-y} < 2^{n-k+2l-y+1}
\]

Assume first that \(y = n - 1\). The inequality follows from the fact that \(P(B) = P(B') = 0\) as ones located at positions in \(B\) cannot be moved any further. Thus we can assume that \(y < n - 1\). Now, as either \(B' = B\) or \(B' = \{x, \ldots, y-1\} \cup \{y+1\}\) and the latter happens with probability \(\frac{1}{n-1}\), we get

\[
P(B') = \sum_{i=x}^{y-1} 2^{n-k+2(l+i-y)-i} - \sum_{i=x}^{y} 2^{l+i-y} + 2^{n-k+2l-y-1} = P(B) - 2^{n-k+2l-y-1} \leq \frac{3}{4} P(B).
\]

And in turn

\[
EP(B') \leq \left(1 - \frac{1}{n-1}\right) P(B) + \frac{1}{n-1} \cdot \frac{3}{4} P(B) = \left(1 - \frac{1}{4(n-1)}\right) P(B).
\]

Note that any configuration \(C\) is the union of disjoint blocks \(B_i\) and \(P(C) = \sum_i P(B_i)\), thus also

\[
EP(C') = \sum_i EP(B'_i) \leq \sum_i \left(1 - \frac{1}{4(n-1)}\right) P(B_i) = \left(1 - \frac{1}{4(n-1)}\right) P(C)
\]

The initial value of \(P(C_0)\) is bounded by \(2^n\). When after \(t\) random comparisons \(EP(C_t) \leq n^{-\eta}\) the sequence is sorted whp. This holds because the probability that after \(t\) random comparisons the sequence is not sorted is equal to the probability that the potential is greater than zero (i.e., at least 1 as the potential is always integral). This probability is less than or equal to \(EP(C_t)\) which is not bigger than \(n^{-\eta}\).
Let \( c = \left(1 - \frac{1}{4(n-1)}\right) \). Let also \( P(C_j) \) and \( P(C_{j+1}) \) be the potentials of the configurations separated by the \( j \)th consecutive comparison. We have shown earlier that \( EP(C_{j+1}), \) conditioned on the value of \( P(C_j), \) is at most \( c \cdot P(C_j) \). This implies that the unconditional value of \( EP(C_{j+1}) \) is at most \( c \cdot EP(C_j) \). Thus by an induction argument it follows that after \( t \) random comparisons \( EP(C_t) \) is at most \( c^t \cdot EP(C_0) \). Finally as \( EP(C_0) = P(C_0) \), where \( C_0 \) is the initial configuration and its potential is not a random variable, in order to estimate \( t \) we get inequality

\[
EP(C_t) \leq \left(1 - \frac{1}{4(n-1)}\right)^t P(C_0) \leq \exp\left(-\frac{t}{4(n-1)}\right) 2^n \leq n^{-\eta},
\]

which holds for \( n \ln 2 + \eta \ln n \leq \frac{t}{4(n-1)} \), equivalent with

\[
t \geq 4(n-1)(n \ln 2 + \eta \ln n).
\]

This concludes the proof of Theorem 3.

6 Strand self-replication

In this section we propose and analyse a self-replication mechanism allowing simultaneous reproduction of (possibly different) strands (line-segments carrying information) of agents. Each strand has the head, the first agent, and the tail, the last agent on the line-segment, separated by regular (internal) agents. We assume that each agent in the strand carries a 0/1 bit of information, and we interpret this sequence of bits as one or a collection of messages.

When a strand is ready for self-replication it first creates a copy of its head, then pushes through this new head (one by one, preserving the order) the bits of information pulled from its own agents. At the same time, in order to accept the incoming bits of information, first the new head and later the copies of the consecutive regular agents (one by one, preserving the order) the bits of information pulled from its own agents. At the same time, in order to accept the incoming bits of information, first the new head and later the copies of the consecutive regular agents (one by one, preserving the order) the bits of information pulled from its own agents.

In the neutral state \( H \) the relevant state is \( |B_2| \). Here the control message \( \psi \) indicates that further extension is expected at the current end of the new strand. In this strand we distinguish also state \( \psi \) (await further information) in agents just added at the tail end.

Below we explain how the information (the sequence of bits) is transferred from the old to the new strand. The full list of rules governing the strand self-replication protocol follows.

(R1) Start of the strand self-replication The replication process begins when the head \( H \) in the neutral state \( \phi \) interacts with a free agent in state \( F \):

\[
<H, B_0, \phi > + F + 0 \rightarrow
<H, B_0, \phi^H > + < H, B_0, \psi > + 1
\]
When this rule is applied, in the old strand signal $\phi^H$ (move all bits towards the head) is created, and in the new line signal $\psi$ means await further instructions (either to add a new agent or to conclude the replication process).

**R2** Create $|B_x|^H$ or $|B_T|^H$ bit message When signal $\phi^H$ arrives at the $(i-1)^{th}$ agent and the $i^{th}$ agent is neutral, message $|B_x|^H$ is placed in the buffer of the latter.

\[
< R, B_i, \phi > + < R|H, B_{i-1}, \phi^H > + 1 \rightarrow
< R, B_i, |B_x|^H > + < R|H, B_{i-1} + \phi^H > + 1
\]

A similar action is taken at the tail agent in neutral state $< T, B_T, \phi >$

\[
< T, B_T, \phi > + < R|H, B_{i-1}, \phi^H > + 1 \rightarrow
< T, B_T, |B_T|^H > + < R|H, B_{i-1} + \phi^H > + 1
\]

The rules in R2 enable propagation of the request to pipeline all information bits towards the head $H$. The rules R3 and R4 govern the relevant bit movement.

**R3** Move a non-tail bit message $|B_x|^H$ towards $H$

\[
< R, B_i, |B_x|^H > + < R|H, B_{i-1}, \phi^H > + 1 \rightarrow
< R, B_i, \phi^H > + < R|H, B_{i-1}, |B_x|^H > + 1
\]

Note that when the bit message $|B_x|$ is moved state $\phi^H$ requesting further bit messages remains in the $i^{th}$ agent.

**R4** Move the tail bit message $|B_T|^H$ towards $H$

\[
< T|R, B_i, |B_T|^H > + < R|H, B_{i-1}, \phi^H > + 1 \rightarrow
< T|R, B_i, \phi > + < R|H, B_{i-1}, |B_T|^H > + 1
\]

Note that when the tail message $|B_T|^H$ is moved the neutrality of the tail agent is restored. Eventually, thanks to the final transfer of the tail message (to the new strand) states of all buffers in the old strand are reset to $\phi$.

The following two rules govern transfer of bit messages between the old and the new strand.

**R5** Transfer a non-tail bit message $|B_x|^H$ to the head of the new strand

\[
< H, B_0, |B_x|^H > + < H, B_0, \psi^T > + 1 \rightarrow
< H, B_0, \phi^H > + < H, B_0, |B_x|^T > + 1
\]

After the transfer across the two strands the bit message is now targeting the tail end.

**R6** Transfer the tail bit message $|B_T|^H$ to the head of the new strand

\[
< H, B_0, |B_T|^H > + < H, B_0, \psi^T > + 1 \rightarrow
< H, B_0, \phi > + < H, B_0, |B_T|^T > + 0
\]

As indicated earlier, transfer of the tail message to the new strand and removal of the bridging edge restore the neutrality of the old strand which is now ready to reproduce again.

Finally, we discuss the remaining rules governing the new strand creation. Recall that the control message represented by state $\psi$ at the current end of the new strand indicates that this strand can be still extended.

**R7** Move a non-tail message $|B_x|^T$ towards the tail end

\[
< H|R, B_i, |B_x|^T > + < R, B_{i+1}, \psi^T > + 1 \rightarrow
< H|R, B_i, \psi^T > + < R, B_{i+1}, |B_x|^T > + 1
\]

After this move the $i^{th}$ agent in the new strand awaits further bit messages.

**R8** Move the tail message $|B_T|^T$ towards the tail end

\[
< H|R, B_i, |B_T|^T > + < R, B_{i+1}, \psi^T > + 1 \rightarrow
< H|R, B_i, \phi > + < R, B_{i+1}, |B_T|^T > + 1 >
\]
After this move the neutrality of the \(i^{th}\) agent in the new strand is restored, i.e., no further bit messages from the head end are expected.

When there is no room for the bit message coming from the head end another agent has to be added to the tail end of the new strand. This is done in two steps. In the first step the current tail end requests addition of a new agent with control message \(\psi^N\).

**(R9) Request strand extension with \(\psi^N\) on non-tail bit message** \(|B_x|^T\) arrival

\[
< R, B_i, |B_x|^T > + < R, B_{i+1}, \psi > + 1 \rightarrow \\
< R, B_i, |B_x|^T > + < R, B_{i+1}, \psi^N > + 1.
\]

The analogous rule requesting extension beyond the head of the new strand is

\[
< H, B_0, |B_1|^H > + < H, B_0, \psi > + 1 \rightarrow \\
< H, B_1, |B_1|^H > + < H, B_0, \psi^N > + 1.
\]

When ready (signal \(\psi^N\) is present) the new agent is added from the pool of free agents.

**(R10) Extend the new strand**

\[
< H|R, B_i, \psi^N > + F + 0 \rightarrow \\
< H|R, B_i, \psi^T > + < R, *, \psi > + 1
\]

Note that after this rule is applied the newly added agent still awaits its bit message which is denoted by \(*\). This new bit message arrives with the help of the following two rules.

**(R11) Arrival of a non-tail bit message**

\[
< H|R, B_i, |B_x|^T > + < R, *, \psi > + 1 \rightarrow \\
< H|R, B_i, \psi^T > + < R, B_x, \psi > + 1
\]

As a non-tail bit arrived the new strand will be still extended which is denoted by messages \(\psi^T\) (expect more bit messages from the head end) in the \(i^{th}\) agent and \(\psi\) (further extension still possible). The situation is different when the tail bit message arrives.

**(R12) Arrival of the tail bit message**

\[
< R, B_i, |B_T|^T > + < R, *, \psi > + 1 \rightarrow \\
< R, B_i, \phi > + < T, B_T, \phi > + 1
\]

After this rule is applied the neutrality at the tail end of the new strand is restored.

Note, however, that since the neutrality of the agents closer to the head of this line was restored earlier the front of the new line can be already involved in the next line replication process. But since we use different messages for the transfers in the old and the new lines, the two simultaneously run processes will not interrupt one another.

We conclude this section with two theorems. Theorem \(\text{[4]}\) refers to the correctness of the proposed self-replication protocol and Theorem \(\text{[5]}\).

**Theorem 4.** The strand self-replication protocol based on rules R1-R12 is correct.

*Proof.* We argue first about correction of the proposed protocol in the replicated (old) strand. One can observe that the bit messages stored in the agents of the strand move along consecutive edges towards the head \(H\). They do not change their order as they only move when the preceding bit message vacates the relevant buffer. Finally, to conclude the replication process neutrality of each agent need to be restored, and this is done by the eventual transfer of the tail message \(|B_T|^H\). In what follows we discuss the actions in all three types of agents in the strand.

- The actions of the tail node are governed by rules R2 and R4. The first rule creates bit message \(|B_T|^H\) and the second moves this message towards the head of the line, restoring the neutrality of the tail agent.

- The actions of a regular node require also rule R3 which supports movement of multiple non-tail bit messages towards \(H\). And when the tail bit message arrives the neutrality of this regular agent is restored by rule R4 applied to this agent twice, first on the right then on the left side of this rule.

- The actions of head \(H\) are more complex. The self-replication begins with application of rule R1 which comprises three different actions: forming a bridging edge, adding the head of the new line, and replication of its bit message in the newly formed head. This is followed by transfer of non-tail bit messages to the new strand by alternating use of rules R3 and R5. When eventually the tail bit message arrives during application of rule R4, the neutrality of the head is restored by rule R6. This concludes the replication process.
The new line formation requires different organisation of states and transitions. Note that all agents added to the line must originate in state $F$, see Figure 1. Also in this case we argue that the bit messages arrive in the unchanged order and eventually the neutrality of all agents is reached (starting from the head and finishing with the tail agent) with the help of the tail bit message $|B_T|^T$.

- Formation of the tail agent requires application of only two rules: $R_{10}$ to add a new agent and $R_{12}$ to equip this agents with the tail message $|B_T|$, when neutrality of this agent is reached.

- The situation with the regular nodes is more complex as they have to accept and store their own bit message $|B_i|$ (done by rule $R_{11}$), add additional agent (via alternating application of rules $R_9$ and $R_{10}$) moving all non-tail bit messages following $|B_i|$ in the old strand (rule $R_7$) until the tail bit message arrives (rule $R_8$) and finally neutrality of the regular agents is reached via rule $R_8$ or rule $R_{12}$ if the agent precedes the tail agent.

- Rule $R_1$ creates the head of the new line, rules $R_9$ and $R_{10}$ add a new agent, rules $R_5$ and $R_7$ move non-tail bit messages in the direction of the tail until the tail bit message arrives (rule $R_6$) when the neutrality of the head is reached (rule $R_8$).

As discussed earlier in the new strand what matters is that neutrality is reached earlier by agents located closer to the head, as this strand is allowed to start self-replication while some bit messages (from the old strand) are still being moved towards the tail end (which may not be fully formed yet). However, it is enough to observe that these two replication processes are independent as they are based on movement of bit messages towards the opposite directions, and in turn they share no rules.

**Theorem 5.** The strand self-replication protocol based on rules $R_1$-$R_{12}$ concludes in parallel time $O(n(k + \log n))$. 
Proof. In the proof we would like to use the parallel to the random bubble-sort process. When the replication happens the bit messages are moved from the old strand to the newly formed one without changing their order. Thus they behave the same way as ones in random bubble-sort which is a stable sorting (ones do not change their order). In the replication process any undelivered yet bit message attempts to move forward (along the combined strands) with probability $\frac{2}{2n(n-1)}$ (instead of $\frac{1}{n-1}$ adopted in the bubble sort argument). So after one interaction we get the following counterpart of the inequality from Lemma 9

$$EP(C') \leq \left(1 - \frac{1}{2n(n-1)}\right) P(C).$$

Also, as there are $k$ bit messages and the tail message has to move $2k$ times (other bit messages move shorter distances) we get $P(C_0) \leq 2^k$ in this case. In order to estimate the number of interactions $t$, after which $EP(C_t) \leq n^{-\eta}$, we get inequality

$$EP(C_t) \leq \left(1 - \frac{1}{2n(n-1)}\right)^t P(C_0) \leq \exp\left(-\frac{t}{2n(n-1)}\right) 2^k \leq n^{-\eta},$$

which holds for $3k \ln 2 + \eta \ln n \leq \frac{t}{2n(n-1)}$ and in turn for

$$t \geq 2n(n-1)(3k \ln 2 + \eta \ln n).$$

Corollary 1. The line self-replication protocol generates $l$ copies of the strand in parallel time $O(n(k + \log n) \log l)$.

7 Open Problems

We claim that our line formation protocol is optimal, i.e., no population protocol can construct a line containing all agents in parallel time $o(n \log n)$ whp. Going beyond strand self-replication protocols it would be interesting to learn what other network structures can self-replicate and at what cost. Also further studies on utilisation of strands (as carriers of information) in more complex distributed processes is needed. Finally, one should develop a computational model which captures more efficiently the features of classic population protocols and constructors. This is to reduce a substantial gap in the time complexity between the protocols designed for these two models.

References

[1] D. Alistarh, J. Aspnes, D. Eisenstat, R. Gelashvili, and R.L. Rivest. Time-space trade-offs in population protocols. In Proc. SODA 2017, pages 2560–2579, 2017.

[2] D. Alistarh, J. Aspnes, and R. Gelashvili. Space-optimal majority in population protocols. In Proc. SODA 2018, pages 2221–2239, 2018.

[3] D. Alistarh and R. Gelashvili. Polylogarithmic-time leader election in population protocols. In Proc. ICALP 2015, pages 479–491, 2015.

[4] D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In Proc. PODC 2004, pages 290–299, 2004.

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

[6] P. Berenbrink, G. Giakkoupis, and P. Kling. Optimal time and space leader election in population protocols. In Proc. STOC 2020, pages 119–129, 2020.

[7] P. Berenbrink, D. Kaaser, P. Kling, and L. Otterbach. Simple and efficient leader election. In Proc. SODA 2018, volume 61 of OASICS, pages 9:1–9:11, 2018.

[8] A. Bilke, C. Cooper, R. Elsässer, and T. Radzik. Brief announcement: Population protocols for leader election and exact majority with $O(\log^2 n)$ states and $O(\log^2 n)$ convergence time. In Proc. PODC 2017, pages 451–453, 2017.

[9] I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P.G. Spirakis. Passively mobile communicating machines that use restricted space. Theor. Comput. Sci., 412(46):6469–6483, 2011.

[10] H-L. Chen, R. Cummings, D. Doty, and D. Soloveichik. Speed faults in computation by chemical reaction networks. In Proc. DISC 2014, pages 16–30, 2014.
[11] M. Connor, O. Michail, and P. Spirakis. On the distributed construction of stable networks in polylogarithmic parallel time. *Information*, 12(6):254–266, 2021.

[12] A. Czumaj and A. Lingas. On truly parallel time in population protocols. *CoRR*, abs/2108.11613, 2021.

[13] D. Doty. Timing in chemical reaction networks. In *Proc. SODA 2014*, pages 772–784, 2014.

[14] D. Doty, M. Eftekhari, L. Gąsieniec, E.E. Severson, G. Stachowiak, and P. Uznański. A time and space optimal stable population protocol solving exact majority. *CoRR*, abs/2106.10201, to appear at FOCS 2021, 2021.

[15] D. Doty and D. Soloveichik. Stable leader election in population protocols requires linear time. In *Proc. DISC 2015*, pages 602–616, 2015.

[16] L. Gąsieniec and G. Stachowiak. Enhanced phase clocks, population protocols, and fast space optimal leader election. *J. ACM*, 68(1):2:1–2:21, 2021.

[17] L. Gąsieniec, G. Stachowiak, and P. Uznański. Almost logarithmic-time space optimal leader election in population protocols. In *Proc. SPAA 2019*, pages 93–102, 2019.

[18] Svante Janson. Tail bounds for sums of geometric and exponential variables. *Statistics and Probability Letters*, 135(1):1–6, 2018.

[19] R.A. Freitas Jr and R.C. Merkle. *Kinematic Self-Replicating Machines*. Landes Bioscience, Georgetown, TX, 2004.

[20] D.E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley, 1973.

[21] T.A. Lincoln and G.F. Joyce. Self-sustained replication of an rna enzyme. In *Science, Vol 323, Issue 5918*, pages 1229–1232, 2009.

[22] O. Michail and P. Spirakis. Simple and efficient local codes for distributed stable network construction. *Distributed Computing*, 29(3):207–237, 2016.

[23] E. Moulin and N. Giuseppone. Dynamic combinatorial self-replicating systems. In *Constitutional Dynamic Chemistry*, pages 87–105. Springer, 2011.