Positve Aging Admits Fast Asynchronous Plurality Consensus

Gregor Bankhamer
gbank@cs.sbg.ac.at
University of Salzburg
Austria

Dominik Kaaser
dominik.kaaser@uni-hamburg.de
Universität Hamburg
Germany

Robert Elsässer
elsa@cs.sbg.ac.at
University of Salzburg
Austria

Matjaž Krnc
matjaz.krnc@upr.si
University of Primorska
Slovenia

ABSTRACT
We study distributed plurality consensus among \( n \) nodes, each of which initially holds one of \( k \) opinions. The goal is to eventually agree on the initially dominant opinion. We consider an asynchronous communication model in which each node is equipped with a random clock. Whenever the clock of a node ticks, it may open communication channels to a constant number of other nodes, chosen uniformly at random or from a list of constantly many addresses acquired in previous steps. The tick rates and the delays for establishing communication channels (channel delays) follow some probability distribution. Once a channel is established, communication between nodes can be performed instantaneously.

We consider distributions for the waiting times between ticks and channel delays that have constant mean and the so-called positive aging property. In this setting, asynchronous plurality consensus is fast: if the initial bias between the largest and second largest opinion is at least \( \sqrt{n} \log n \), then after \( O(\log \log n \cdot k \cdot \log k + \log \log n) \) time all but a \( 1/\polylog n \) fraction of nodes have the initial plurality opinion. Here \( \alpha \) denotes the initial ratio between the largest and second largest opinion. After additional \( O(\log n) \) steps all nodes have the same opinion w.h.p., and this result is tight.

If additionally the distributions satisfy a certain density property, which is common in many well-known distributions, we show that consensus is reached in \( O(\log \log n \cdot k + \log n) \) time for all but \( n/\polylog n \) nodes, w.h.p. This implies that for a large range of initial configurations partial consensus can be reached significantly faster in this asynchronous communication model than in the synchronous setting.

To obtain these results, we first assume the existence of a designated base station and later present fully distributed algorithms. Additionally, we derive tail bounds on the Pólya-Eggenberger distribution, which might be of independent interest.

1 INTRODUCTION

Plurality Consensus is a fundamental problem in distributed computing. We are given a set of \( n \) nodes, each of which starts with its own initial opinion (or color) from a set of size \( k \). The goal is to design an efficient distributed protocol which ensures that all nodes agree on the opinion, which is initially supported by the most nodes, provided a sufficiently large initial bias is given.

In failure-rate distributions, the concept of aging describes how a component or a system improves or deteriorates with age. “No aging” means that the age of a component has no effect on the distribution of residual lifetime of the component. This unique case describes a Poisson-clock based survival distribution, which is widely used to describe asynchronous models. The family of positive aging distributions describes the more general situation where the residual lifetime decreases or remains the same with increasing age of a component [33]. Such situations are common in reliability engineering where components tend to become worn out with time due to increased wear and tear, as well as in real-life waiting time scenarios. Prominent members of this family of distributions include the exponential, Rayleigh, Weibull (with shape parameter at least 1), and Gamma (with parameter at least 1) distributions.

In this paper we consider an asynchronous communication model, where nodes are equipped with a random clock. If the clock of a node advances, then the node is activated, and we say that this node ticks. Upon a tick, nodes may start establishing communication channels to constantly many other nodes. The opening of communication channels is subject to random delays, and communication partners may be chosen uniformly at random or from a list containing constantly many node addresses acquired in previous communication steps. As long as both – the ticking time and the
channel delay – satisfy the positive aging property, our protocols guarantee fast convergence to the initial plurality opinion. Moreover, if these distributions also satisfy what we call the \( q \)-density property (see Property 2) – fulfilled by a number of well-known distributions (e.g., exponential, Rayleigh or Weibull with shape parameter at least 1) – then all but \( n/\text{polylog} n \) nodes agree w.h.p.\(^1\) on the initially dominant opinion significantly faster than in the corresponding synchronous setting for a large range of initial configurations. In that sense, our algorithms break the lower bound for plurality consensus in the synchronous model, see Section 4.

1.1 Related Work

**Synchronous Protocols.** In pull voting [30] each node contacts a neighbor uniformly at random in every step and adopts its opinion. This process eventually converges to a monochromatic configuration. In the two-choices process [18] each node contacts two random neighbors, and if the two opinions coincide, then the opinion is adopted. While pull voting requires convergence time \( \Omega(n) \), two-choices reduces this time to \( O(\log n) \) [18] in complete graphs for two colors, if the initial bias is large enough.

The so-called 3-majority dynamics for plurality consensus with \( k \) opinions was analyzed in [10]. In this protocol, each node samples three neighbors and adopts the majority opinion among the sample, breaking ties uniformly at random. The authors prove a tight running time of \( \Theta(k \cdot \log n) \) for this protocol, given a sufficiently large bias. In [9], the three-state population protocol from [5] is adopted and generalized to \( k \) opinions. The resulting bound on the running time depends on the 2-norm of the initial opinion configuration. More recently, a detailed study and comparison of the 3-majority dynamics and the related two-choices process has been performed by [12]. Subsequently, a tight analysis of these processes was presented in [27]. Together, [27] and [12] cover a large range of parameters \( k \).

In [15], two plurality consensus protocols are proposed. Both assume a complete graph and realize communication via the random phone call model. The first protocol is very simple and, w.h.p., achieves plurality consensus within \( O(\log(k) \cdot \log \log n + \log \log n) \) rounds w.h.p. using \( \Theta(\log \log k) \) bits of additional memory. The second, more sophisticated protocol achieves plurality consensus within \( O(\log n \cdot \log \log n) \) rounds w.h.p. using only 4 overhead bits. Here, \( k \) denotes the initial ratio between the largest and second-largest opinion. They require an initial absolute bias of \( \omega(\sqrt{n \log^2 n}) \).

In [24] and [28], two similar protocols were presented which achieve (almost) the same running time bounds.

**Asynchronous Protocols.** Population protocols [6] are a model for asynchronous distributed computation. In the basic variant, nodes are modeled as finite state machines. The protocols run in discrete time steps, where in each step a pair of nodes is chosen uniformly at random to interact. The interacting nodes update their states according to a simple deterministic rule.

In [5], a three-state population protocol for majority (consensus with two opinions) was proposed that converges after \( O(n \log n) \) interactions (\( O(\log n) \) parallel time) w.h.p. If there is a bias of at least \( \omega(\sqrt{n \log n}) \), the protocol converges to the majority w.h.p.

Two similar four-state protocols that solve exact majority were presented in [22, 34]. The protocols are guaranteed to converge to the initial majority opinion regardless of the initial bias, but they require \( \Omega(n^2) \) interactions in expectation. Recently, a large number of papers has considered the stabilization time for exact majority, see [2–4, 11, 13, 14]. The currently best known protocol from [11] requires \( O(\log n) \) states and \( O(\log^{3/2} n) \) parallel time.

Plurality consensus and the related dual problem of coalescing random walks [1] have also been considered in certain asynchronous models. For an arbitrary number of initial random walks which evolve according to some reversible Markov chain generator, the expected coalescence time is bounded by the largest hitting time of an element in the state space [35]. This time corresponds to the expected time needed for the corresponding pull voting process to converge. In [20], the so-called linear voting model has been introduced, which covers a number of synchronous and asynchronous voting protocols. They show that the expected time of asynchronous pull voting on a graph with minimum degree \( d_{\min} \) and conductance \( \Phi \) is bounded by \( O(nm/(d_{\min} \Phi)) \). Here, asynchronicity means that at each step one single node is selected u.a.r., and this node chooses a random neighbor for communication. So-called discordant voting processes have been considered in [17], where in every time step a pair of nodes with different opinions is selected for an interaction. In [16], plurality consensus in general graphs and for general bias is solved using load balancing in different communication models. In [24], plurality consensus in a synchronous and an asynchronous model is considered. In the asynchronous case, they assume that each node has a Poisson clock ticking with rate 1. Whenever the clock of a node ticks, it may choose up to a constant number of random neighbors, and revise its opinion based on the set of received opinions. They show that if initially the size of the largest opinion exceeds the size of the second largest one by some factor \((1 + \epsilon), \epsilon > 1 \) constant, and the number of opinions is \( O(\exp(\log n/\log \log n)) \), then (partial) consensus is achieved in time \( O(\log n) \) w.h.p. Note that there are no communication delays and once a communication partner is chosen, communication happens instantaneously.

1.2 Model

Our model comes with two different forms of asynchronicity, the waiting time between local operations (ticking time) and the delay required to engage in communication (channel delay). For the ticking time, every node is equipped with a random clock following a decreasing conditional survival or increasing failure rate \( \Phi \).

**Property 1 (Positive Aging).** Let \( T \) be a non-negative distribution and \( X \sim T \). Then \( T \) has the positive aging property if and only if \( P(X > s) \geq P(X > t + s | X > t) \) for all \( s, t > 0 \).

When a node ticks, it may start establishing communication channels to a constant number of nodes, chosen either uniformly at random or from a list of constantly many addresses acquired in some previous communication steps. In contrast to the synchronous...

\(^1\)With high probability (w.h.p.) refers to a probability of at least \( 1 - n^{-\Omega(1)} \).

\(^2\)Our results except Theorem 4.2) still hold if we require this to only hold for \( s > C \) for some constant \( C \). For the sake of readability of our analysis we assume that \( C = 0 \).
case, we assume that after initiating a communication channel, some time is required to build up a connection to the sampled node. This time – the channel delay – is also assumed to follow a distribution with the positive aging property. Once the channels to all requested nodes are established, messages can be exchanged. For such an exchange of messages no additional time is required. This reflects the fact that in various scenarios (e.g. three-way handshake, DNS lookup, or key-exchange for encryption) the time required for opening a communication channel may dominate the time required for the entire communication. For both the ticking time and the channel delay we assume that their distributions take values from a non-negative domain with constant mean.

**Remembering Node Addresses.** Many of the results in synchronous and asynchronous plurality consensus assume that each node may only contact random neighbors [16, 24, 28]. In our work we assume that nodes may remember the addresses of constantly many nodes, which may be reused for communication in future steps. This allows nodes to communicate with a designated base station or set of leader nodes. We note that such a modification of the random phone call model in rumor spreading leads to improvements of the running time [7, 21, 29] or computational complexity [25] of standard push-pull protocols. Also in plurality consensus remembering node IDs has lead to extended results in certain cases, see, e.g., [19].

### 1.3 Our Results

We are given \( n \) nodes, each of which holds initially one of \( k \) different opinions. We assume that \( 2 \leq k \leq n^\epsilon \) for any constant \( 0 < \epsilon < 1/2 \). Let \( a_0 \) and \( b_0 \) be the (relative) size of the initially largest and second largest opinion, respectively. We assume that the initial (absolute) bias \( n \cdot (a_0 - b_0) \) is at least \( \sqrt{n} \log n \) and we use \( \alpha \) to denote the corresponding relative bias, defined as \( \alpha = a_0/b_0 \).

**Algorithmic Approach.** Similar to the protocols mentioned above, our plurality consensus algorithms employ well-known population dynamics. In particular, we use pull voting and the \( 2 \)-majority dynamics (also called the two-choices process). The nodes pass through a sequence of numbered stages, which we call generations. The intuition is that a certain generation implies a certain chance for the nodes to have the initially dominant opinion. This latter property makes the concept of generations a crucial part of our algorithms.

The essential idea of our approach is the following. Every time a node \( v \) becomes active, it may sample two nodes. Depending on the sample, it may perform one of the following two actions. A so-called two-choices step is executed if:

- (i) the two sampled nodes are in the same, \( i \)-th generation,
- (ii) this generation is at least as high as \( v \)'s generation,
- (iii) they have the same opinion, and
- (iv) the total number of nodes of that generation is large enough.

In this case \( v \) adopts the sampled opinion and proceeds to generation \( i + 1 \). Otherwise, the node \( v \) performs a so-called propagation step, where it adopts the generation and opinion of the node with the highest generation among the sample, provided this generation is higher than its own (breaking ties arbitrarily). In the analysis we will show that the ratio between the largest and second largest opinions grows rapidly as the generations become higher. As a consequence, any node has the initial plurality opinion once it reaches a certain generation.

**Positive Aging in Plurality Consensus.** Many important distributions we consider for clock ticks and channel delays do not allow consensus among all nodes in time less than \( O(\log n) \). However, as we show later, partial consensus can be achieved much faster. Here, partial consensus means that all but at most \( n/\text{polylog} n \) nodes agree on the initial majority opinion. In particular, we show that in our setting partial consensus is reached in \( O(\log \log n \cdot k \cdot \log k + \log \log n) \) time w.h.p. Then, \( O(\log n) \) further steps suffice for all nodes to agree on the initial majority opinion, w.h.p.

We apply aforementioned algorithmic approach and use the concept of generations as well as the method of alternating between two-choices and propagation steps. In order to determine the time when a two-choices step may be performed (see requirement (iv) above), we introduce a leader-based mechanism, which allows the system to be aware of the moments in time when the number of nodes in the highest generation is large enough (which, in turn, results in the creation of a new generation).

We first present an algorithm where we assume that there is one predefined base station in the system. This base station has a restricted amount of memory \( O(\log n) \) bits and if a node sends a request to this node, then it answers with the values stored in this memory. More precisely, the base station has a value for the highest generation allowed to be created in the system (initially set to 1), and it stores a bit which indicates whether the nodes should perform two-choices or propagation steps.

When a node \( v \) is activated by a tick, it contacts the base station and two randomly chosen nodes. If the base station’s bit allows two-choices and the generation stored in it’s memory is higher than the generation of \( v \), then \( v \) performs a two-choices step – if conditions (i)-(iii) are fulfilled as described above (see Algorithmic Approach). When a node contacts the base station, it sends its generation number to it so that the base station can maintain the number of nodes in the highest generation created so far. Once the majority of all nodes are in the highest generation, the base station allows the nodes to promote themselves to a higher generation by setting the corresponding bit accordingly and allowing two-choices steps. These alternating two-choices/propagation stages are repeated until the last generation created is monochromatic w.h.p. A formal description of this protocol is given in Section 2.

Finally, we extend the algorithm described above to a distributed system without a predefined base station in Section 3. First, we partition almost all nodes into clusters of size polylog \( n \). During this procedure, leaders emerge in all these clusters. Then, these
leaders act in a distributed manner to coordinate the actions of the nodes, and we derive an algorithm that mimics the procedure designed for the case with a base station. This allows us to show a similar result as in the previous case, however, without assuming the existence of a designated base station.

Comparison with Related Work. For initial configurations with \(\kappa = \Theta(1)\) our protocols match the optimal \(O(\log n)\) convergence time for full consensus. A similar result is achieved by [5, 24] with respect to the Poisson-clock model and population protocols. If \(\kappa = \omega(1)\) then our protocols reach partial consensus faster than related approaches [16, 20, 24] that operate in a comparable asynchronous model (i.e., Poisson-clock model, population protocols and sequential model of [16] with \(O(\log n)\) bits of memory per node).

Some of this improvement is related to the fact that our model allows nodes to remember (and reuse) addresses of constantly many nodes (see Section 1.2).

Our algorithmic approach can also be implemented in the synchronous round-based model. This algorithm achieves (full) plurality consensus in \(O(\log k \cdot \log \log n + \log \log \log n)\) rounds w.h.p. Note that this matches current state-of-the-art results of approaches operating in the synchronous setting (e.g. [15, 24, 28]). The basic idea is to define a sequence of rounds \(\{t_i\}_{i \geq 1}\) at which each node is allowed to perform a two-choices step. Then, at every \(t_i\), a new generation \(i\) is created via two-choices step w.h.p. This sequence of time steps is chosen in such a way that throughout the steps \(t_i, t_i+1, \ldots, t_{i+1} - 1\) the generation created at time \(t_i\) grows to a constant fraction of nodes. We achieve this by setting \(t_{i+1} - t_i = C \cdot \log k\) for some sufficiently large constant \(C\).

Breaking the Lower Bound for Synchronous Consensus Processes. Many well-known distributions such as exponential, Rayleigh or Weibull with shape parameter at least 1 satisfy besides positive aging also the \(q\)-density property (Property 2, formally defined in Section 4). This property guarantees that within any time frame of length \(1/\log n\) any node ticks and establishes its communication channels to constantly many nodes with probability at least \(1/\text{polylog} n\). If the distribution of the waiting time between two ticks as well as of the channel delays satisfy this additional property, then the partial consensus time can significantly be reduced. We show that under these conditions, in time \(O(\log \log n \cdot k + \log \log n)\) all but \(n/\text{polylog} n\) nodes agree on the initial majority opinion w.h.p.

For a large range of initial configurations, this convergence time is significantly better than any synchronous algorithm can achieve with the same limitations on the number of communication partners of a node per time step as in the asynchronous model. Note that a similar phenomenon has been observed in rumor spreading w.r.t. synchronous vs. asynchronous algorithms [26]. Furthermore we show that, assuming that communication can be performed instantly and nodes are activated according to Poisson clocks, partial consensus can be reached in time as long as \(O(\log \log n)\) for an initial bias of at least \(2\sqrt{n} \log^4 n\). This is a significant improvement over the \(O(\log n)\) (partial) convergence time of [24]. While their model does not allow node addresses to be stored, they otherwise operate in this Poisson clock based model and consider a much higher initial bias of \(\alpha > (1 + \epsilon)\) for constant \(\epsilon > 0\). See Section 4 for further discussion.

Tail Bounds on the Pólya-Eggenberger distribution with \(s = 1\). We model parts of our analysis with the help of a so-called Pólya-Eggenberger urn process [23]. The process starts with \(a\) black and \(b\) white balls and consists of \(n\) steps in total. In each step, a black ball is added with probability corresponding to the fraction of black balls currently in the system. Otherwise, a white ball is added to the urn. The related distribution \(\text{PE}_1(a, b, n)\) models the number of black balls added throughout these \(n\) steps, and is denoted by \(\text{PE}_1(a, b, n)\) in the following. It is known (e.g. page 181 of [31]) that this distribution is equivalent to the binomial distribution \(\text{Bin}(n, P)\), where the success probability \(P\) is drawn a priori from the beta distribution Beta\((a, b)\). Using this representation together with a recently developed tight bound on the Beta distribution [36], we state a result that might be of independent interest. For additional discussion and a proof see [8].

Theorem 1.1. Let \(A \sim \text{PE}_1(a, b, n - (a + b))\), \(\mu := (a/(a + b))n\) and \(a + b \geq 1\) as well as \(n \geq a + b\). Then, for any \(\delta\) with \(0 < \delta < \sqrt{a}\) it holds for some universal constant \(c > 0\) that

\[
\Pr\{A + \mu \leq \sqrt{a} \cdot \frac{n}{a + b} + \delta\} < 4 \exp(-c_2 \cdot \delta^2)
\]

\[
\Pr\{A + \mu > \sqrt{a} \cdot \frac{n}{a + b} + \delta\} < 4 \exp(-c_2 \cdot \delta^2)
\]

2 PROTOCOL WITH A BASE STATION

The main difficulty in analyzing our asynchronous protocols lies in the fact that we cannot predict (accurately) when a new generation has to be created, since the nodes lack a global notion of time. This is further complicated by the fact that nodes cannot easily decide based on their local view when to execute two-choices and propagation steps. As a first approach, we therefore resort to a so-called base station that is constrained to \(O(\log n)\) bits of memory. Later, we present a fully distributed algorithm, which does not require any base station. Our intermediate result is the following.

Theorem 2.1. Assume a designated base station is present. The protocol defined in Algorithm 1 reaches partial consensus in

\[O(\log \log n \cdot k \cdot \log k + \log \log n)\]

\(\text{time}\) w.h.p. Within additional \(O(\log n)\) time, all nodes have the initially dominant opinion w.h.p.

2.1 Our Protocol

We analyze the protocol defined in Algorithm 1, where we assume that a base station is present. This base station receives signals from nodes and performs simple counting operations, which are defined in Algorithm 2. It’s purpose is to orchestrate the distributed computation by providing two variables, \(\text{gen}\) and \(\text{mode}\). The variable \(\text{gen}\) represents the currently highest allowed generation in the system, initially set to 1. The variable \(\text{mode}\), initially set to \(\text{TC}\) (meaning two-choices), indicates whether nodes in generation \(\text{gen}\) should perform two-choices steps.

When a node ticks, it requests the state of the base station and uses its variable \(\text{node}\) to decide which operation to execute (see Line 8 and Line 11 of Algorithm 1). If a tick occurs while waiting for the channel(s) in Line 6 to be established, we only allow \(v\) to send out a 0-signal to the base station. The remaining operations
Algorithm 1 Consensus protocol for node $u$.

initialize $u$.gen, $u$.col[0]) ← (0, initial color of node $u$)
for each tick of node $u$ do
  send signal 0 to the base station $\ell$.
  if a previous tick is still being processed then
    skip the remainder of the procedure
  sample nodes $v_1$ and $v_2$ u.a.r.
  wait for communication channels to $\ell$, $v_1$, and $v_2$ to open
  w.l.o.g. assume $v_1$.gen ≥ $v_2$.gen
  if $\ell$.mode = propagate and $v_1$.gen > $u$.gen then
    $\ell$.mode ← Propagation
    ($u$.gen, $u$.col[$v_1$.gen]) ← ($v_1$.gen, $u$.col[$v_1$.gen])
  else
    $\ell$.mode ← Two-Choices
    $v_1$.col[$\ell$.gen − 1] = $v_2$.col[$\ell$.gen − 1] ≠ NIL then
      ($u$.gen, $u$.col[$\ell$.gen]) ← ($v_1$.gen, $u$.col[$v_1$.gen])
      send signal $u$.gen to the base station $\ell$

Algorithm 2 Consensus protocol for the base station.

initialize ($\ell$.gen, $\ell$.mode, $\ell$.gensize, $\ell$.ticks) ← (1, TC, 0, 0)
for each incoming signal $i$ do
  if $i = 0$ then
    $\ell$.ticks ← $\ell$.ticks + 1
    if $\ell$.ticks = $\ell$.ticks + 1
      $\ell$.mode ← allow propagation
    $\ell$.mode ← propagate
  if $i = \ell$.gen then
    $\ell$.gensize ← $\ell$.gensize + 1
    if $\ell$.gensize ≥ $\ell$.ticks + 1, TC
      ($\ell$.gen, $\ell$.mode) ← ($\ell$.gen + 1, TC)
    ($\ell$.gensize, $\ell$.ticks) ← (0, 0)

are skipped in such a case. Note that a 0-signal may need time to reach the base station (the channel opening delay), but nodes do not need to wait for the actual channel to be established.

Besides knowledge of $n$, we require that the base station knows upper and lower bounds on the means of the waiting time and channel delay distributions (hidden in a constant $C_1$, see Observation 2.2).

For simplicity of presentation we defined Algorithm 1 in such a way that node $u$ stores the opinion of generation $i$ as $u$.col[i]. Note that this is done in the pseudocode for presentation purposes only. For our analysis, it suffices that nodes store their current opinion and the opinion of the previous generation, $u$.col[u].gen] and $u$.col[u].gen − 1], respectively. If a node $u$ does not hold any opinion for generation $i$, we say that $u$.col[i] = NIL. This is initially the case for all $i > 0$ and might occur, e.g., if node $u$ jumps two generations in a propagation step. For the range of initial configurations we consider, $O(log k + log log n)$ bits are required for the transmission and storage of the color and generation values.

Notation and Conventions. We define $g_i(t)$ to be the fraction of nodes of generation $i$ at time $t$. Furthermore, we denote by $c_{j,i}(t)$ the fraction of these $g_i(t)$ · $n$ nodes which have $u$.col[i] = $j$, and let $p_i(t) = \sum_j c_{j,i}(t)^2$. Note that $1/k ≤ p_i(t)$ holds as long as $g_i(t) > 0$.

Let $a_i(t)$ denote the relative ratio between the most and second-most dominant color in generation $i$ at time $t$. We denote by $t_i$ the point in time when generation $i$ was first allowed by the base station, and let $t_i(\gamma)$ correspond to the time when generation $i$ globally reaches cardinality $n$. Throughout the analysis we may fix a generation $i$ and time $t$ and let $a$ and $b$ be the opinions with the largest and the second largest support in generation $i$ at time $t$, respectively. We then define $a_i(t) = c_{a,i}(t)$ and $b_i(t) = c_{b,i}(t)$ for easier readability. Furthermore, for variables with generation subscript $i$ we sometimes omit the parameter $t$ to denote time $t_{i+1}$ (e.g., $a_i = a_i(t_{i+1})$). Also, if we say that a node $v$ is of color $j$ at some time $t$, we mean $v$.col[v].gen] = $j$. Similarly, we will say $v$ takes (or adopts) color $j$, if $v$ increases its generation to some generation $i$ and sets $v$.col[i] ← $j$.

2.2 Core Concepts of our Analysis

Time Measures. At the core of the analysis lies the so-called time unit. A time unit denotes the number of time steps $C_1$ with the following property: Within any time interval of length $C_1$, each node establishes with probability $0.9$ the channels to three nodes chosen for communication. The crucial point is that this time unit is independent of the nodes execution history. If the distributions of the channel delays and the time between ticks have the positive aging property, we show that such a time unit $C_1$ is of constant length. Unless explicitly stated otherwise, we measure the time in time units. Counting 0-signals in Algorithm 2 allows the base station to approximate the time accurately.

Observation 2.2. Consider a set of nodes $U$ sending 0-signals to a designated node $v$ upon each activation, where $|U| ≥ log^2 2+\epsilon$ for some constant $\epsilon > 0$. Let $T ≥ C_1$. Then, there exists a constant $C_1'$ that depends only on $C_1$ such that node $v$ receives $C_1' log |U|$ many 0-signals in at least $T$ and at most $O(T)$ time steps w.h.p.

Note that proper bounds on the constant $C_1'$ as well as the constant hidden in $O(T)$ can be computed explicitly, see [8]. In this section, the designated node $v$ is the base station, and $U$ contains all other nodes.

Time Between Consecutive Generations. We now consider a fixed generation $i$. That means, we consider the time frame $[t_i, t_{i+1}]$ in which the base station has $\ell$.gen = $i$. We are interested in an upper bound on the time frame $t_{i+1} – t_i$. Starting from time $t_i$, we know by Observation 2.2 that after $\Theta(1)$ time units the condition in Line 4 of Algorithm 2 becomes satisfied w.h.p. Throughout this time, sufficiently many nodes promote themselves to generation $i$ via two-choices steps.

Proposition 2.3. Fix some generation $i$ and assume that $g_i(t_i) ≥ 1/2$. Let $t_i + t'$ denote the time when the base station allows promotions to generation $i$ via propagation. Then, $g_i(t_i + t') ≥ p_i(1/5)$ w.h.p.

From time $t_i + t'$ until $t_{i+1}$, the base station only allows propagation steps. Therefore, one can see the set of nodes of generation $i$ as a set of informed nodes, which grows by pull broadcasting (cf. [32]). That is, the set of nodes of generation $i$ increases by a constant factor in every time unit w.h.p.

Proposition 2.4. Fix some generation $i$ and let $t_i + t'$ denote the time when the two-choices phase of generation $i$ ends. Then,
\[ i'' = \log_{1.4}(\frac{3}{p_{i-1}}) \text{ time units after the base station starts allowing propagation steps, the cardinality of the } i\text{-th generation exceeds } n/2 \text{ w.h.p.} \]

Remember that as soon as \( t_1(1/2) \) is reached, generation \( i + 1 \) is allowed by the base station (see Line 8 of Algorithm 2). Therefore, it follows that \( t_{i+1} - t_i = O(\log(1/p_{i-1})). \) For the proofs we refer to the full version [8].

**Concentration Results.** We again consider some fixed generation \( i. \) Let \( a \) and \( b \) be the largest and second largest opinion in generation \( i - 1 \) at time \( t_i. \) We show that the color fractions \( a_i(t) \) and \( b_i(t) \) are well concentrated around their expectation. Throughout the analysis we assume that color \( b \) still has significant support, i.e., \( b_{i-1} \gg 1/\sqrt{n}. \) Here \( x_1 \gg x_2 \) means that there exists a constant \( c > 0 \) s.t. \( x_1 \geq x_2 \cdot n^c. \) Otherwise \( a_i = 1 - o(1), \) and within \( O(1) \) generations, the first monochromatic generation is reached. A monochromatic generation \( t' \) w.r.t. color \( a \) is a generation where all nodes \( v \) either have \( v.\text{col}(i') = a \) or \( v.\text{col}(i') = \text{NIL} \) at any time \( t. \)

We start by focusing on the time frame \([t_i, t_i + t']\), where \( t' \) is defined s.t. at time \( t_i + t' \) the two-choices phase of generation \( i \) ends. Observe that a node \( v \) that attempts a two-choices step (see Line 11 in Algorithm 1) at time exactly \( t_i \) samples two nodes \( v_1, v_2 \) with defined color value and \( v_1.\text{col}[i] = v_2.\text{col}[i] = i \) with probability exactly \( c_{i,i-1}^2 \cdot g_i^2. \) As in the time frame \([t_i, t_i + t']\) the base station only allows two-choices steps to generation \( i \), no other node \( v' \) will modify its \( \text{color} \) of \( \text{col}(i-1) \) field. Hence, any node that joins generation \( i \) through \([t_i, t_i + t']\) takes some fixed color \( j \) with probability exactly \( c_{j,i-1}^2/p_{i-1}. \) This allows us to state the following.

**Lemma 2.5.** Let \( a \) and \( b \) be the largest and second largest opinion in generation \( i - 1 \) at time \( t_i \) and assume that \( a_{i-1} > b_{i-1} \gg 1/\sqrt{n}. \) Let \( t_i + t' \) be the time when the propagation phase for the \( i\text{-th generation} \) begins. Then w.h.p.

\[
\begin{align*}
a_i(t_i + t') &= a_i(t_i) \left( 1 + O\left( \frac{\log n}{n} \frac{1}{a_{i-1}} \right) \right), \quad \text{and} \\
b_i(t_i + t') &= b_i(t_i) \left( 1 + O\left( \frac{\log n}{n} \frac{1}{b_{i-1}} \right) \right).
\end{align*}
\]

Combining Lemma 2.6 and Lemma 2.5, we can describe how color fractions behave throughout generation \( i \), and we show that the bias almost squares when generation \( i + 1 \) is arises.

**Lemma 2.7.** Let \( a \) and \( b \) be the largest and second largest opinion in generation \( i - 1 \) at time \( t_i \) and assume that \( a_{i-1} > b_{i-1} \gg 1/\sqrt{n}. \) Let \( t_i + t' \) be the second largest opinion in generation \( i \) at time \( t_i+1. \) If \( a_{i-1} - b_{i-1} \geq \log n/\sqrt{n} \), then w.h.p.

\[
\begin{align*}
(1) \quad a_i &= a_i(t_i + t') \left( 1 + O\left( \frac{\log n}{n} \frac{1}{a_{i-1}} \right) \right), \quad \text{and} \\
(2) \quad b_i &= b_i(t_i + t') \left( 1 + O\left( \frac{\log n}{n} \frac{1}{b_{i-1}} \right) \right).
\end{align*}
\]

A repeated application of the above gives us that the initially most supported color stays dominant, and after \( O(\log \log n) \) generations the second-most dominant color is of insignificant size. This implies that after \( O(1) \) further generations the first monochromatic generation appears w.h.p. The proofs for above statements can be found in the full version [8].

**Putting Everything Together.** Summarizing, we established that \( t_{i+1} - t_i = O(\log(1/p_{i-1})) = O(\log k). \) As the relative bias is roughly squared each time a new generation is created, the generation \( \log_{1.5} \log_{\log n} n + O(1) \) will be monochromatic. Note that from this point on (i) every further generation will also be monochromatic, and (ii) at least \( n/2 \) nodes carry the majority opinion. Hence, \( O(\log \log n) \) time units suffice to reach partial consensus. This translates into a required time of \( O(\log_{\log n} n \cdot \log k \cdot \log \log n). \) This bound can be tightened slightly to yield the result stated in Theorem 2.1 by observing that \( a_{i-1} \geq k \) implies \( \alpha_{i+1} - \alpha_i = O(1) \).

### 3 DECENTRALIZED PROTOCOL

The centralized approach with a predefined base station from Section 2 violates the distributed computing paradigm and has several drawbacks. Most notably, a huge number of requests is induced on the base station in each time step and thus the base station becomes the bottleneck of the execution of the protocol. Furthermore, the system becomes highly vulnerable against attacks, since an adversary can compromise the entire computation by taking over the base station. To avoid these drawbacks and decentralize the
computation, we introduce some changes to our protocols, which guarantee a maximum congestion of $O(\text{polylog } n)$ per node.

The execution of the protocol runs in two parts, clustering and consensus. In the clustering part we first use a distributed algorithm to cluster the nodes into groups of roughly polylog $n$ nodes and each cluster elects its own leader. In the consensus part we define the behavior of the leaders of different clusters and their interactions with non-leader nodes, such that all of them collaborate in order to emulate the protocol described in Section 2. For both parts, the required storage per node as well as the size of information exchanged through each communication channel can be bounded by $O(\log n)$ bits. Formally, we show the following statement.

**Theorem 3.1.** The decentralized protocol reaches partial consensus in $O(\log \log n \cdot k \cdot \log k \cdot \log \log n)$ time w.h.p. Within additional $O(\log n)$ time, all nodes have the initially dominant opinion w.h.p.

**The Clustering Algorithm.** In the first part, all but a fraction of $O(1/\text{polylog } n)$ nodes are partitioned into clusters of polylogarithmic size, each containing a distinguished leader which is the leader of this cluster. Our clustering algorithm achieves this w.h.p. in $O(\log \log n)$ time. It also ensures that, w.h.p., each such cluster has size at least $\log^{c-1} n$, where $c > 4$ is an arbitrary constant that is governed by the algorithm. In that way, we no longer have one designated base station, but $O(n/\text{polylog } n)$ decentralized cluster leaders. Additionally, these cluster leaders trigger the start of the consensus algorithm. The clustering algorithm is presented and analyzed in the full version [8].

### 3.1 Description of the Consensus Protocol

After the above-mentioned clustering algorithm, all nodes have to perform our consensus protocol, however the nodes that emerged as leaders throughout the clustering protocol also have to carry out so called leader tasks. We start by describing the protocol for the follower nodes as it does not differ much from the centralized procedure (see Algorithm 1).

**The Follower Perspective.** Each time the clock of a node $v$ ticks, it sends a 0-signal to its leader and (unless an execution started by a previous tick is still in progress) executes the following algorithm. It opens channels to three nodes $v_1$, $v_2$, and $v_3$ chosen uniformly at random, as well as to its own leader $l$ and to $l_3$, the leader of node $v_3$. As soon as all connections are established, $v$ requests the current opinion and generation from $v_1$ and $v_2$. Furthermore, the state of the leader $l_3$ is pulled. Recall that once the channels are established, this information can be retrieved instantly and simultaneously. The possible actions of $v$ are very similar as in the centralized protocol; however, they depend on the generation number and propagation bit of the (almost) uniformly sampled $l_3$ instead of its own leader $l$. If the information provided by $v_1$ and $v_2$, together with the state of $l_3$ satisfies the two-choices conditions, then a two-choices step is performed. More precisely, if

- $v_1$ and $v_2$ have non-NIL color values for generation $i-1$ as well as $v_1$, col $i-1] = v_2$, col $i-1]$, and
- the highest generation allowed by $l_3$ is $i$, and $l_3$ allows promotion via two-choices steps

then, $v$ will adopt the opinion of $v_1$ and $v_2$ and set its generation to $i$. If according to $l_3$ a propagation step is to be performed, then $v$ executes a propagation step just as in the centralized procedure (see Line 8 of Algorithm 1). That is, $v$ adopts the color and generation of either $v_1$ or $v_2$ in case one of them is of generation higher than $v$. Finally, $v$ transfers state information of $l_3$ to its own leader $l$, together with $v$’s possibly increased generation value.

**The Leader’s Perspective.** As opposed to the centralized case, where the base station simply switches between two-choices and propagation mode, leaders now pass through two additional phases. These two additional phases, called sleeping and preparation phase, ensure that leaders progress through their generations quite synchronously. For one, achieve that leaders start allowing any fixed generation $i$ at roughly the same time. Additionally, prevent leaders from allowing two-choices steps while other leaders allow propagation (or vice versa), in order to reuse many parts of the analysis of the centralized case, where the two-choices and propagation phase are properly separated.

With this in mind, the leaders procedure can be described as follows. Consider some leader $l$ that just started allowing nodes to promote themselves to a new generation $i$. This leader will employ a counter $l$.ticks (just as in the centralized case, see Algorithm 2) in order to count all 0-signals it receives from its followers. At the beginning of a generation $i$, the leader starts by allowing two-choices steps towards generation $i$, and keeps counting the received 0-signals of its followers to measure time. After receiving sufficient 0-signals (an amount linear in the number of its followers), the leader enters the so-called sleeping sub-phase. Note that the 0-signal counting threshold is set to ensure that w.h.p. there exists a one time-unit frame in which all leaders simultaneously allow promotions via two-choices before the first leader enters the sleeping phase.

During this sleeping sub-phase, which lasts for a constant amount of time, the leader again counts incoming 0-signals to measure time, but neither allows two-choices nor propagation steps. This forces leaders to wait for some time before entering the propagation phase and allowing promotion via propagation, preventing an interleaving of two-choices and propagation phases throughout the system. Recall that $l$ receives the state information of randomly sampled leaders $l_3$ at each execution of its followers. In case $l$ is currently in the sleeping phase and some leader $l_3$ already allows propagation steps, $l$ will stop sleeping and switches to the propagation phase immediately. This way we ensure that no leader is left asleep while some of them may already be finishing their propagation phase.

After the sleeping sub-phase ends, the leader starts allowing propagation steps and thereby enters the propagation phase. The idea behind this sub-phase is the same as in the centralized case, to quickly spread generation $i$. However, when it comes to determining when the next generation $i+1$ should be allowed, a more elaborate mechanism than then the one from the centralized algorithm in Section 2 is needed. In the centralized protocol, the base station simply incremented a counter each time a node promotes to generation $i$. As in this decentralized case each leader only has a limited view consisting of its followers, a different approach needs to be employed to estimate the time at which at least $1/2$ of all nodes belong to generation $i$. We interrupt our explanation of the leaders protocol to explain how this can be achieved.

**Estimating Global Properties.** Recall that each follower sends the state of the randomly sampled leader $l_3$ to its own leader $l$ upon
each execution of the follower procedure. This state information allows leaders to harvest some information about the global state of the network. Indeed, if a leader receives polylog \( n \) of such randomly sampled leader-states, it may accurately predict the (global) fraction of leaders satisfying a certain property. For example, let \( R \) be such a leader-property which is satisfied whenever the majority of this leaders followers is of generation \( i \). Clearly, a leader \( i \) can determine this property by maintaining an \( l_{\text{gen}} \)ize variable. Suppose now that \( l \) receives polylog \( n \) consecutive messages regarding random leaders \( l_t \) satisfying property \( R \). In this case \( l \) can be (almost) sure that globally 1/2 of all nodes are already in generation \( i \). A detailed description of this sampling mechanism together with its analysis can be found in the full version [8].

Using the above approach, the leaders are only allowed to enter the preparation sub-phase after estimating that at least 1/2 of all nodes belong to generation \( i \). This guarantees w.h.p. that no leader will start this sub-phase too early. Upon entering the preparation sub-phase, a leader will still allow propagation steps for some time, but additionally it will again count the incoming 0-signals. This is done to ensure further \( \Theta(1) \) waiting time after which all the leaders are guaranteed to have reached this sub-phase w.h.p. Afterwards, the leader denies both two-choices and propagation steps for \( \Theta(1) \) time, which prevents propagation steps from occurring during the two-choices phase of the next generation \( i + 1 \). Finally, the leader resets its counters, increases its highest allowed generation to \( i + 1 \) and starts passing through the 4 sub-phases as part of generation \( i + 1 \). See [8] for a more detailed explanation.

### 3.2 Core Concepts of the Analysis

Roughly, the correctness of our algorithm follows from the analysis results of the centralized approach. To show this we start by the following observations: (i) a follower node \( v \) will perform two-choices or propagation steps based on the leader \( l_t \) that is chosen independently of the nodes \( t_1, t_2 \) and \( l \), (ii) if at some point all leaders allowed the same generation and sub-phase (e.g. two-choices), then the protocol mimics the behavior of the centralized approach, and (iii) leaders progress through some fixed generation \( i \) almost synchronously. To further elaborate on the third point, we now state a selection of the most important invariants which are maintained as the leaders progress through the mentioned sub-phases of arbitrary generation \( i \). We employ the same notation as defined on page 5, with the exception of \( t_1 \) now denoting the time at which the fastest leader starts allowing generation \( i \). The proofs can be found in the full version [8].

**Lemma 3.2.** Fix some generation \( i \). Under assumption that all leaders start allowing this generation within a constant time frame, the following statements hold w.h.p.:

1. All leaders allow two-choices steps towards generation \( i \) for at least one simultaneous time unit.
2. Starting at \( t_1 \), no leader allows any propagation steps until every leader exits the two-choices sub-phase.
3. The last leader enters the propagation phase at most \( O(1) \) time after the first does so.
4. No leader enters the preparation phase before \( t_1(1/2) \).
5. Every leader allows generation \( i + 1 \) before time \( t_{i+1} + \Theta(1) \).

Note that Statement 5 implies that w.h.p. all the above statements hold in the following generations as well. To this end, define \( t' \) such that at time \( t_1 + t' \) even the slowest leader has just finished its two-choices phase. In the analysis of the centralized approach, we established that if the base station allows two-choices steps for (at least) one full time unit, then Theorem 2.3 follows. Hence, Statement 1 allows us to carry over this result. Furthermore, by Statement 3 it follows that leaders quickly allow nodes to start spreading generation \( i \) via pull propagation, implying the statement of Theorem 2.4. Therefore, the time between \( t_{i+1} \) and \( t_1 \) follows the asymptotic bounds as in the centralized case. Also, Statement 4 guarantees that majority of all nodes belong to generation \( i \) before the two-choices phase of the next generation starts.

When it comes to the concentration of color fractions and evolution of the bias, Statement 2 is of importance. It implies that w.h.p. leaders never allow two-choices and propagation steps at the same time. Hence, each time a node in \( [t_1, t_1 + t'] \) promotes to generation \( i \), it is a result of a successful two-choices step. Due to similar reasons as in the centralized case (see paragraph before Lemma 2.5), and because \( l_t \) is selected independently from \( v_1 \) and \( v_2 \), such a promotion will cause the node to take color \( j \) with probability \( c_{i,j}^2 t_{j-1} / p_{j-1} \). This is the main ingredient of the proof of Lemma 2.5. Furthermore, starting at time \( t_1 + t' \), nodes will join generation \( i \) via propagation steps only. This allows us to again model the set of nodes that take some color \( j \) when promoting to generation \( i \) during \( [t_1 + t', t_{i+1}] \) with a Pólya-Eggenberger process. This leads to the statement of Lemma 2.6 and finally Lemma 2.7.

Summarizing, we show the same asymptotic guarantees as in the centralized case for both the required number of generations, as well as for the increase of bias with each further generation. A detailed discussion regarding these results can be found in [8].

### 3.3 Termination

This algorithm as well as the centralized algorithm in Section 2 guarantee that the nodes eventually reach partial and full consensus, w.h.p. However, without additional modifications neither of both procedures terminate such that nodes eventually know that they are in (global) consensus and cease the execution of the protocol. In [8] we present an extension to our algorithms that achieves proper termination.

### 4 BREAKING THE LOWER BOUND ON SYNCHRONIZED PROTOCOLS

In this section we first outline the *Accelerated Consensus Protocol*, a modification of the decentralized protocol from Section 3 and then we argue that this protocol breaks a lower bound on plurality consensus protocols in the synchronous model. As before, we assume that all but \( n/polylog \( n \) nodes are partitioned into clusters of size at least \( polylog \) \( n \). For the accelerated protocol we now assume that in addition to the positive aging property the distributions for the waiting time between ticks and the channel delays are \( q \)-dense for some constant \( q > 0 \).

**Property 2 (\( q \)-dense distribution).** Let \( T \) be a non-negative distribution and \( X \sim T \). Then \( T \) is \( q \)-dense if and only if there exists a constant \( t > 0 \) such that \( P(X < s) > s^q \) for all \( 0 < s < t \).
The main difference to the decentralized protocol is the following. All nodes in a cluster share the same generation and color, which are stored at the cluster leader. Each time a follower performs a two-choices or propagation step, the shared variable of its cluster leader is updated (instead of its own as part of the decentralized procedure). So whenever a two-choices or propagation step updates color or generation, this change is reflected at the leader. Similar, each time a node is queried for its color or generation, it will answer with its leaders shared values instead. That way, followers only act as proxies and help to achieve consensus among the shared color values that are stored at each cluster.

Property 2 together with positive aging guarantees that in every time frame of length $O(1/\log n)$ a follower of each cluster ticks and establishes communication channels to all chosen nodes w.h.p., as long as the clusters are of large enough (polynomial) size. In case of the decentralized protocol, leaders spend most of their time in the propagation sub-phase (which is the only sub-phase taking $o(1)$ time each generation). Now, consider the Accelerated Consensus Protocol, and assume that at some point during generation $i$, all leaders allow propagation to generation $i$. As at least one follower of each cluster ticks within every time frame of $O(1/\log n)$, this can be seen as spreading generation $i$ between clusters via pull broadcast at an $\Omega(\log n)$ accelerated rate. This way, the time between two consecutive generations, $t_{i+1} - t_i$, can be reduced to $O(1)$ w.h.p. More details and an analysis can be found in [8].

**Theorem 4.1.** Assume that the initial absolute bias is greater than $2\sqrt{n}\log^c n$ for any constant $c' > 4q + 4$. Then the Accelerated Consensus Protocol reaches partial consensus in $O(\log\log n \cdot k \log \log n)$ time w.h.p.

For a simple lower bound on synchronous protocols, we consider the classical synchronous model [9, 10], where we assume that each node may communicate with $O(\log\log n)$ nodes per round. Additionally, we assume that the nodes do not know the set of initial opinions (however $k$ may be known to the nodes). For a node to adopt a certain opinion in this model, it must have interacted at least once with a node that knows about the existence of this opinion. As each node may communicate with at most $O(\log\log n)$ other nodes in each round, in order to spread the initially dominant opinion $a$ (with initial relative support $a_0$) to at least $n/\log\log n$ nodes, one needs $O(\log(1/a_0)/\log\log n)$ time steps.

To compare the running time of the asynchronous protocol with this lower bound, consider for example an initial configuration with $a = 2$ and $k = n^{\epsilon}$ for some constant $0 < \epsilon < 1/2$. If, initially, all opinions besides the majority opinion have roughly the same support, then our algorithm requires $O(\log\log n)$ time to reach partial consensus w.h.p. Any protocol operating in the synchronous round-based model requires $\Omega(\log n/\log\log n)$ time for this task.

**Further Acceleration.** In case all nodes are activated by Poisson clocks with mean $1$, and the exchange of information can be performed instantly, above protocol can be further improved. Instead of being constrained to approximate time frames of (at least) constant length via counting of 0-signals (see Observation 2.2), leaders can approximate time frames of $1/\log\log n$ accurately in this setting - as long as their cluster is of large enough polynomial size. This is implied by the so-called memoryless property of the exponential distribution, as well as the fact that instant communication implies that during some time frame $[t', t'']$, leaders will only receive 0-signals that were initiated exactly during this time frame. This allows us to speed up not only the propagation phase but also every other phase by a factor of $\Omega(\log^2 n)$ - in some sense this can be seen as reducing the length of a time unit to $O(1/\log^2 n)$. This allows full consensus between leaders to be reached after $O(1)$ time. The total running time is then dominated by the clustering procedure and the time followers require to collect the final color values of their leaders. We show the following in [8].

**Theorem 4.2.** Assume the waiting time between ticks follows Exp(1) and information between nodes can be exchanged instantly. Then, the Accelerated Consensus Protocol can be modified s.t. for an initial bias of at least $2\sqrt{n}\log^4 n$, it reaches partial consensus in time $O(\log\log n)$.

### 5 Conclusion

In this paper we considered the plurality consensus problem for the setting where we require a certain initial bias between the largest and second largest opinion. We focused on a particular variant of an asynchronous communication model and showed that asynchronous plurality consensus is fast: after $O(\log\log n \cdot k \log k + \log\log n)$ time steps all but a $1/\log n$ fraction of nodes have the initial majority opinion. Furthermore, we modify these algorithms such that for a large range of initial configurations and distributions, partial consensus is achieved faster than in any algorithm that operates in the corresponding synchronous setting.

In the future we would like to look at several related questions which are still open. One possible extension would be to model communication delays on a message basis instead of a channel basis. However in such a model it seems that one cannot avoid the interleaving of the two-changes sub-phase with the propagation sub-phase within the same generation. An even more ambiguous question would be to try analyze the leaderless variant of the protocol: each time a node ticks it samples two random nodes and executes a propagation step or a two choices step (whichever possible). In such a setting there are no limitations when, e.g., a higher generation is allowed. While this approach raises many technical difficulties related to the analysis of the running time, our experimental results show that this leaderless algorithm, despite its simplicity, behaves similarly as the ones described in this paper.

### Acknowledgments

The authors would like to thank Felix Biermeier and Janko Gravner for helpful discussions and important hints.

All authors were partially supported by the Austrian Science Fund (FWF) under grant no. P 27613 ("Distributed Voting in Large Networks"). The first and the second author received funding from the European Union’s Horizon 2020 research and innovation programme under Grant Agreement no. 824115 (HiDALGO). The fourth author acknowledges partial support of the Slovenian Research Agency (research programs P1-0383, P1-0297 and research projects J1-1692, J1-9187) and the European Commission for funding the InnoRenew CoE project (Grant Agreement no. 739574) under the Horizon 2020 Widespread-Teaming program and the Republic of Slovenia.
