# Bounded Fairness for Probabilistic Distributed Algorithms*

Pepijn Crouzen, Ernst Moritz Hahn, Holger Hermanns
Saarland University
Saarbrücken, Germany
Email: {crouzen,hahn,hermanns}@cs.uni-saarland.de

Abhishek Dhama, Oliver Theel
Carl von Ossietzky University
Oldenburg, Germany
{abhishek.dhama,oliver.theel}@informatik.uni-oldenburg.de

Ralf Wimmer, Bettina Braitling, Bernd Becker
Albert-Ludwigs-University
Freiburg, Germany
{wimmer,braitlin,becker}@informatik.uni-freiburg.de

***Abstract*—This paper investigates quantitative dependability metrics for distributed algorithms operating in the presence of sporadic or frequently occurring faults. In particular, we investigate necessary revisions of traditional fairness assumptions in order to arrive at useful metrics, without adding hidden assumptions that may obfuscate their validity. We formulate faulty distributed algorithms as Markov decision processes to incorporate both probabilistic faults and non-determinism arising from concurrent execution. We lift the notion of bounded fairness to the setting of Markov decision processes. Bounded fairness is particularly suited for distributed algorithms running on nearly symmetric infrastructure, as it is common for sensor network applications. Finally, we apply this fairness notion in the quantitative model-checking of several case studies.**

***Keywords*-Markov decision processes; schedulers; bounded fairness**

## I. Introduction

In a distributed algorithm several separated processes with only local knowledge must cooperate to achieve a common goal. Such algorithms have been extensively studied over more than 40 years [19]. Abstract properties of distributed algorithms are traditionally established by proving correctness guarantees under certain assumptions.

Such analyses may be called *qualitative* as they establish a particular quality of the algorithm under study. Such guarantees are usually based on strong assumptions about the dependability of the system. Either it is (implicitly) assumed that no communication fault occurs [19], or it is assumed that only a limited number of faults occur [13], or it is assumed that at some point faults stop occurring.

With the rise of the internet and of wireless networks, solutions capable of operating in unpredictable and unreliable environments have found practical relevance. For the field of distributed algorithms this means that we can no longer make strong assumptions about the (eventual) absence of faults. Especially sensor networks must operate in an environment where the frequent occurrence of faults is the rule, not the exception.

Traditional qualitative guarantees therefore do not apply. This motivates an effort to instead quantitatively analyze distributed algorithms [21], [9], [18], [17]. Instead of proving that an algorithm works, *quantitative* analysis aims at studying *how well* an algorithm works. This shift in focus however implies that established assumptions, models and algorithms that have been used for qualitative analysis are not necessarily adequate in the quantitative setting. First of all we must alter the assumptions on dependability. Instead of assuming that there only occur a limited number of faults, or that faults occur only in a certain time-period, we now assume that transient faults can occur at any time, but with a specific probability. This allows to calculate probability bounds on the correct operation of the algorithm [18], [17] provided fault probabilities are given.

The present paper focuses on a further set of assumptions that must be reconsidered in order to arrive at an insightful quantitative analysis. These assumptions concern the order in which the different processes operate in a distributed algorithm. For a qualitative analysis, it is common to pose the most general assumption under which the algorithm is assumed to work correctly. Dijkstra, for instance, states that "nothing may be assumed about the relative speeds of the $N$ computers" [19] in his seminal work on distributed mutual exclusion.

However under such an assumption quantitative analysis quickly becomes uninteresting: the worst-case success probability (that all processes can enter their critical section within some finite time) is provably zero under such a weak assumption. Also standard fairness assumptions do not change this phenomenon, as we will discuss. More appropriate assumptions have already been studied in the context of qualitative analysis, namely where the amount of time between two steps of a process is *bounded* [4], [15], [22]. This paper investigates the concept of bounded fairness in the context of quantitative

*This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS) (see www.avacs.org for more information) and the Graduiertenkolleg "Leistungsgarantien für Rechnersysteme".

1550-4808/11 $26.00 © 2011 IEEE
DOI 10.1109/ACSD.2011.21

analysis of probabilistic distributed algorithms. We formally define the notion of bounded fairness for probabilistic models with non-determinism and apply it in the analysis of several case studies.

The models we are dealing with are non-deterministic and probabilistic at the same time, and known as Markov decision processes [29]. As our modeling and analysis vehicle, we use the probabilistic model-checker PRISM [27], and its modeling language. The theory developed in this paper is however not bound to this tool. It draws some inspiration from a recent experimental scheduler comparison [5].

Our work can be considered in contrast to the simulation-based analysis of distributed systems as it prevails in the systems community. In that approach, dependability metrics are often skewed by hidden assumptions that are added by the simulator, the libraries, the runtime environment, the random number generator, or the postprocessing of simulation traces. In the wireless sensor community, there is a growing awareness [1], [24], [26] for this problem, and this paper constitutes a step towards a solution.

## II. Preliminaries

In this section we briefly discuss the general theory of Markov decision processes.

### A. Markov decision processes

We denote the collection of all probability distributions on (Borel) subsets of a set $S$ as $\mathcal{P}(S)$.

*Definition 1:* A Markov decision process (MDP) is a tuple $(S, A, R)$ where: $S$ is the state space, $A$ is a set of actions and $R : S \times A \times \mathcal{P}(S)$ is the transition relation such that for every pair $(s, a) \in S \times A$ we have at most one transition $(s, a, \pi) \in R$. If we find *exactly* one transition $(s, a, \pi)$ per pair $(s, a)$, we say that the MDP is *enabled*.

The state space and the set of actions are discrete. For the remainder of this section we fix a MDP $M = (S, A, R)$.

We say that an action $a \in A$ is *enabled* in a state $s \in S$ if there exists a triple $(s, a, \pi)$ in $R$. We denote the set of enabled actions in a state $s \in S$ as $A_s$. For an enabled MDP we have that $A_s = A$ for all states $s \in S$. We write $\bar{R}(s, a)$ for the random variable which takes values in $S$ and is distributed according to $\pi$. A *path* of $M$ is a, possibly infinite, sequence of states and actions $\sigma = s_1, a_1, s_2, a_2, \ldots$, where each action $a_i$ is enabled in the previous state $s_i$. A finite path $\sigma$ ends in state $last(\sigma)$. The length of a path is equal to the number of actions in the path. This means that a path consisting of a single state has length 0. We denote the set of all finite paths as $(S \times A)^* \times S$ and the set of all infinite paths as $(S \times A)^\omega$. Given a path $\sigma = s_1, a_1, s_2, a_2, \ldots$ the *action trace* $\sigma[A]$ of $\sigma$ consists of the sequence of actions in $\sigma$, i.e. $\sigma[A] = a_1, a_2, \ldots$.

A *scheduler* $f$ is a function from finite paths to distributions over the actions $f : ((S \times A)^* \times S) \to \mathcal{P}(A)$ such that for any finite path $\sigma = s_1, a_1, \ldots, s$ we have for $f(\sigma) = \pi$ that, if the distribution $\pi$ assigns a probability greater than zero to $a$, then $a$ is enabled in $s$. We write $\bar{f}(\sigma)$ for the random variable which takes values in $A$ and is distributed according to $\pi$. A scheduler $f$ induces a Markov chain (MC) $(X^k_{f,M})_{k \in \mathbb{Z}_+}$ with state space $(S \times A)^* \times S$ where for finite paths $\sigma = s_1, a_1, \ldots, s$ and $\sigma' = s_1, a_1, \ldots, s, a, s'$ (i.e., $\sigma$ is the largest prefix of $\sigma'$) we have:

$$P(X^{k+1}_{f,M} = \sigma' | X^k_{f,M} = \sigma) = P(\bar{f}(\sigma) = a) \cdot P(\bar{R}(s, a) = s')$$

All other transition probabilities are zero, since they are not selected by the scheduler under consideration. If we now fix a starting distribution over $S$, we can compute the probability of MDP $M$ being in a state $s \in S$ at a time-point $k \in \mathbb{Z}_+$ given a scheduler $f$ by summing over all paths of length $k$ that end in $s$.

A scheduler is called *deterministic* if it always chooses a point distribution over the set of actions, i.e., a distribution where one action has probability 1 and the others have probability 0. We refer to a set of schedulers as a *scheduler class*. The set of all schedulers for a MDP $M$ is denoted $C_M$.

In the analysis of MDPs, it is often considered interesting to study the maximum or minimum probability of being in a state $s \in S$ in the Markov chains $X_{f,M}$ induced by all schedulers $f$ in a scheduler class. We refer to these probabilities as extremal probabilities. Extremal probabilities correspond to best-case and worst-case behaviors. Thus, bounds on minimal and maximal probabilities imply guarantees for any possible behavior of the system. For instance, we can use these bounds to prove safety properties of the system. If we wish to show that a system is in a "safe" state at a certain time-point with at least probability $p$, we need only show that the minimum probability to be in this "safe" state is larger than $p$. The minimum probability to be in a particular state at a particular time is given by the *infimum transient probability function* of a MDP. Conversely, maximum probabilities are given by the *supremum transient probability function*.

*Definition 2:* Given a MDP $M = (S, A, R)$, a set of initial states $S_{init} \subset S$, a scheduler class $C$, a set of goal-states $G \subset S$ and a time-point $k \in \mathbb{Z}_{\geq 0}$ we define the infimum *transient probability* function $\mathcal{R}^-$ as follows:

$$\mathcal{R}^-(M, S_{init}, C, G, k) = \inf_{f \in C, s \in S_{init}} P(X^{(k)}_{f,M} \in G \mid X^{(0)}_{f,M} = s)$$

The supremum *transient probability* function $\mathcal{R}^+$ is obtained by replacing $\inf$ by $\sup$ in the above.

Another interesting property of a MDP is the average probability to be in a set of states in the long run. That is, if we let the MDP "run" indefinitely, which percentage of the time does the MDP occupy one of the goal states. For minimal long-run average probabilities, the *infimum long-run average probability* function $\mathcal{S}^-$ is defined as the Cesàro limit of the infimum transient probability:

$$\mathcal{S}^-(M, S_{init}, C, G) = \inf_{f \in C, s \in S_{init}} \lim_{t \to \infty} \frac{1}{t} \sum_{k=0}^{t-1} P(X^{(k)}_{f,M} \in G \mid X^{(0)}_{f,M} = s).$$

The *supremum long-run average probability* function $\mathcal{S}^+$ is defined analogously.

For finite models, the Cesàro limits exist and match the standard steady-state limits, i.e., $\lim_{k\to\infty}\mathcal{R}^-(M,S_{init},C,G,k)$ and $\lim_{k\to\infty}\mathcal{R}^+(M,S_{init},C,G,k)$, if these limits exist [29]. The extremal transient probability for a time-point $k$ ranging over all schedulers is always attained by a deterministic scheduler with step-bound $k$.

### B. Extremal probabilities of scheduler classes

We now consider the problem of computing extremal probabilities for a subset $C$ of all schedulers. Existing algorithms compute extremal probabilities over *all* schedulers and can, in general, not be used to compute extremal probabilities over a strict subset $C$ of all schedulers [29], since the scheduler that realizes the extremal probability may not belong to $C$. We show in this subsection that for certain classes of schedulers, relevant to our analysis goal, we can construct a new MDP $M'$ such that the extremal probabilities in $M$ over the schedulers in $C$ agree with the extremal probabilities over all schedulers in $M'$. This then allows to apply the aforementioned algorithms on $M'$, resulting in the extremal probabilities for scheduler class $C$ of $M$.

*Definition 3:* Given a MDP $M=(S,A,R)$ and a function $F$ from finite paths to sets of actions, $F:((S\times A)^*\times S)\to 2^A$ such that for a path $\sigma$ ending in state $s$ we have that $F(\sigma)$ contains only actions enabled in $s$, $F(\sigma)\subset A_s$. Class $C_F$ is the set of all schedulers $f$ of $M$ which satisfy the following: for all paths $\sigma$ of length $k$ which are reached with non-zero probability by $f$, this scheduler only selects actions contained in $F(\sigma)$ with non-zero probability. Formally:

$$P(X^{(k)}_{f,M}=\sigma)>0\wedge P(\bar{f}(\sigma)=a)>0\Rightarrow a\in F(\sigma).$$

We say that the function $F$ *characterizes* the scheduler class $C_F$.

We now show that for any MDP $M$ and scheduler class $C_F$ there exists another MDP $M'$ such that the extremal probabilities over $C_F$ for $M$ can be derived from the extremal probabilities over all schedulers for $M'$.

*Theorem 1:* Given a MDP $M=(S,A,R)$, a set of initial states $S_{init}\subset S$ and a scheduler class $C_F$ characterized by a function $F$, there exists a MDP $M'=(S',A',R')$ and functions $g:S'\to S$ and $h:S\to S'$ such that, for any set $G\subset S$ and time-point $k$, we have:

$$\mathcal{R}^-(M,S_{init},C_F,G,k)=\mathcal{R}^-(M',h(S_{init}),C_{M'},g(G),k)$$
$$\mathcal{S}^-(M,S_{init},C_F,G)=\mathcal{S}^-(M',h(S_{init}),C_{M'},g(G))$$

where $h(S_{init})=\{h(s)\mid s\in S_{init}\}$ and $g(G)=\{x\mid x\in S'\wedge g(x)\in G\}$. Furthermore, the supremum probabilities for $M$ and $M'$ are related in the same way as the infimum probabilities.

*a) Sketch of proof.:* We construct $M'$ as follows. The set of states $S'$ of $M'$ is the set of paths of $M$. The action sets are identical, i.e., $A'=A$. The transition relation $R'$ is defined to ensure that the states of $M'$ indeed match the paths of $M$, for any path $\sigma$ of $M$, action $a$ in $A$ and state $s$ in $S$, where $a\in F(\sigma)$ we have:

$$P(\bar{R}'(\sigma,a)=\sigma as)=P(\bar{R}(last(\sigma),a)=s).$$

If the action $a$ is not allowed by $F$ for *path* $\sigma$ of $M$, then $a$ is not enabled in *state* $\sigma$ of $M'$. In this way we ensure that the set of all schedulers for $M'$ matches the set of schedulers induced by $F$ for $M$.

The function $h$ now simply maps any state $s$ in $S$ to its corresponding zero-length path $s$ in $S'$. The function $g$ maps any path $\sigma$ to its last state $last(\sigma)$.

We can now prove Theorem 1 by mapping the schedulers in $C_F$ for $M$ to schedulers of $C$ and proving that they induce the same path probabilities. The full proof of Theorem 1 can be found in an extended version of this paper [12].

In theory, the derived MDP $M'$ is infinitely large and therefore not amenable to analysis. However, we will see in Subsection IV-D that in practice $M'$ need not always encode the entire path of $M$ to enable the analysis of the scheduler class $C_F$. The size of $M'$ then depends on the nature of the function $F$.

## III. Related work

The work presented here draws some inspiration from the comparative study [5] where PRISM has been used to study a selection of schedulers experimentally on a gossiping information spread algorithm. Since the focus of this paper is on fairness notions for distributed algorithms we here give an overview of fairness notions. For a more extensive overview of related work, which also covers notions of communication and faults we refer to the extended version of this paper [12].

The concept of fairness has been studied extensively as it facilitates discarding certain unrealistic execution paths while verifying liveness properties of distributed algorithms. To that end, various notions of fairness have been proposed in literature. In a broad sense, fairness ensures that an action or a process is activated sufficiently often if it has been enabled often enough [20]. *Weak fairness* ensures that a *continuously* enabled transition is taken *infinitely* often. An *infinitely* often enabled transition is taken *infinitely* often under *strong fairness*. It has been observed that in many instances fairness alone is not sufficient to guarantee that a transition is eventually activated because it is not enabled long enough due to "race conditions" [3]. The notion of *hyperfairness* is proposed to exclude such pathological traces where a transition is not enabled due to intermittent unavailability of the required resources. Hyperfairness is deployed with an underlying notion of fairness. It guarantees that a transition is enabled sufficiently often so that it can be activated by the underlying fairness notion. This notion of hyperfairness is generalized in [28]. Lamport's notion of hyperfairness ensures that a transition is activated *infinitely* often if it is *infinitely often possible*. The relative strengths of various notions of fairness have also been studied [30]. The notion of *probabilistic fairness* has been defined for a class of distributed algorithms [2], [8]. A *probabilistic* scheduler resolves non-determinism by choosing a next transition according to certain probability distribution. It has been shown that a probabilistic scheduler is *fair with probability* 1 if each transition has non-zero

probability of being selected *and* the probability distribution remains unchanged during the system runtime [8].

All the above notions of fairness only require certain actions to take place *at some time-point* under various conditions, but generally do not enforce these actions to occur in a particular time-frame. *Bounded fairness*, on the other hand, guarantees that a *continuously* enabled transition is activated within a *bounded* number of steps and is stronger than the notion of weak fairness [15], [22]. All *k*-bounded schedulers are shown to be equivalent, provided they are "potentially stable," for probabilistically stabilizing algorithms in [6]. A method to compute the best and the worst stabilization time of probabilistically stabilizing algorithms by showing it to be an instance of stochastic shortest path problem is presented in [7]. Distributed agreement has also been studied with lower and upper bounds for the time between two steps of a process [4].

Our main contribution is to extend this notion of bounded fairness, with lower and upper bounds, first to probabilistic models (the MCs induced by MDPs) in Subsection IV-A and then to non-deterministic and probabilistic models, namely MDPs, in Subsection IV-C.

## IV. Bounded fairness for MDPs

Given that in a distributed system no single process has global control, it is not determined in which order the different processes execute steps of their local algorithms. Even if considering symmetric hardware with identical clock speed settings for all partners, the order of execution cannot be assumed fixed, due to the unavoidable phenomenon of clock drift. Thus, the order of execution must be considered to be *non-deterministic*. We now consider what this means for the worst-case stabilization time of a self-stabilizing algorithm, in absence of any assumption on the order of execution.

*Example 1:* As an example we use a simplistic distributed self-stabilizing minimal spanning tree (MST) algorithm. It is a simplification of the one in [11]. Given a set of processes $V = \{1, \ldots, N\}$ which are connected in a weighted graph $(V, E)$ with one special process called the *root*, the task of the algorithm is to compute, for each process, its distance to the root along edges of the graph (the minimal spanning tree of the graph can easily be derived from this information). For an edge $(i, j) \in E$ we write $d_{i,j} \in \mathbb{N}$ for its weight, which represents the fixed distance of process $i$ to process $j$. Each node knows only the distance to its direct neighbours. We write $c_i$ for the current root distance estimate of process $i$. The estimates $c_i$ are initially set to an arbitrary value. Each process $i$ now perpetually repeats the following updates:

$$c_i := \begin{cases} 0, & \text{if } i \text{ is the root,} \\ \min\{d_{i,j} + c_j \mid (i,j) \in E\}, & \text{otherwise.} \end{cases}$$

This algorithm relies on a dependable mechanism for process $i$ to inspect the current values $c_j$.

Assume that, at the start of the minimal spanning tree algorithm, none of the processes has the correct distance value stored locally. A possible order of execution is one where a single process continuously executes local steps. It is obvious that for this order we will never reach a desirable state in which all nodes know their distance to the root even in the absence of transient faults. This is because other processes are blocked indefinitely. This well-known phenomenon is usually addressed by some fairness assumptions.

Under the assumption of strong fairness (in every infinitely long execution of the algorithm each process executes infinitely many local computations), the above algorithm provably converges to the setting where each variable $c_i$ indeed holds the distance of process $i$ to the root [11].

However, if we now consider transient probabilistic faults, this is no longer the case. Assume that during an update of the $i$-th node in the MST algorithm a fault may occur with some non-zero probability that changes the estimate $c_i$ to an arbitrary value. Consider now the situation where exactly one of the nodes has computed the correct distance to the root node. Now the execution, that forces this correct node to execute updates until a fault occurs and its estimate becomes incorrect, is strongly fair. Note that the probability that the node executes infinitely many updates without faults is zero. It follows that there is a strongly fair scheduler for which the probability to succesfully compute the minimal spanning tree is zero. Whenever a node correctly computes the correct distance to the root, this scheduler would force it to perform updates until a local fault occurs.

The scheduler in the above example can be considered unrealistic as, although it is strongly fair and does not allow nodes to execute at an infinitely faster rate than other nodes, it does allow nodes to execute at an *unboundedly* faster rate than other nodes. To combat this issue we extend the notion of bounded fairness [15], [22] to MDPs. As a first step we define bounded fairness for MCs induced from MDP models of a distributed algorithm.

### A. Period of a scheduler

We wish to reason about the relative speeds of the processes in a distributed algorithm. We quantify the speed of a process $v$ by counting how many other processes execute steps between two subsequent steps of process $v$. In the MDP models of distributed algorithms each process $v$ is associated with a unique action $a_v$. Now given a scheduler for such a MDP, we look at the paths induced by this scheduler and the distances between consecutive $a_v$-transitions. We call this distance the *period* of the path at a particular time-point. We assume that the processes of the distributed algorithm are deadlock-free. For this reason we consider only enabled MDPs in the remainder of the paper.

*Example 2:* Consider the MDP $(S, A, R)$ of a distributed algorithm with 4 nodes such that $A = \{a, b, c, d\}$. Now we look at an example of a path of the MDP:

$$s_1, b, s_2, c, s_3, a, s_4, b, s_5, d, s_6, c, s_7, a, \ldots.$$

In particular we are interested in the action trace of the path and the number of steps until the current action appears again, i.e., the *period*.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| action: | $b$ | $c$ | $a$ | $b$ | $d$ | $c$ | $a$ | $\ldots$ |
| period: | 3 | 4 | 4 | $\ldots$ | | | | |

The period tells us at what intervals the nodes in the distributed algorithm execute local steps. Since we consider stochastic models we define the period of a MDP given a particular scheduler as a stochastic process which tells us the probability of observing a particular period at a particular time-point.

*Definition 4:* Given a MDP $M = (S, A, R)$, with a scheduler $f$ that induces a MC $X_{f,M}$, the *period* of the induced MC at time-point $k$ is a stochastic process $\left(\Lambda^{(k)}_{f,M}\right)_{k\in\mathbb{Z}_{\geq 0}}$ which takes values in $\mathbb{Z}_{>0} \cup \{\infty\}$ and has distribution:

$$P(\Lambda^{(k)}_{f,M} = i) = \sum_{a\in A} P(\bar{f}(X^{(k)}_{f,M}) = a \wedge \bar{f}(X^{(k+1)}_{f,M}) \neq a \wedge \ldots \wedge \bar{f}(X^{(k+i-1)}_{f,M}) \neq a \wedge \bar{f}(X^{(k+i)}_{f,M}) = a), i \in \mathbb{Z}_{>0},$$
$$P(\Lambda^{(k)}_{f,M} = \infty) = 1 - \sum_{i\in\mathbb{Z}_{>0}} P(\Lambda^{(k)}_{f,M} = i).$$

Note that $\bar{f}(X^{(k)}_{f,M})$ is the decision of scheduler $f$ at time-point $k$ and $P(\bar{f}(X^{(k)}_{f,M}) = a)$ is then the probability that scheduler $f$ selects action $a$ at time-point $k$. We write $\Lambda^{(k)}$ when the MDP and the scheduler are clear from the context.

If we now consider all possible schedulers for a MDP $M$ then, for the worst-case expected period length we have for any $k \in \mathbb{Z}_{\geq 0}$ that $\sup_{f\in C_M} E(\Lambda^{(k)}_{f,M}) = \infty$. Such a worst-case scheduler is any scheduler that schedules an action $a$ at time $k$ with probability one, but schedules that same action with probability zero for all subsequent steps.

The period of an induced Markov chain does not tell us anything about the *first occurrence* of a particular action. To be able to enforce that the first occurence of an action is not delayed indefinitely we define it as a random variable.

*Definition 5:* Given a MDP $M = (S, A, R)$, with a scheduler $f$ that induces a MC $X_{f,M}$ and an action $a \in A$, the *first occurence* of $a$, $\Delta^{(a)}_{f,M}$ is a random variable which takes values in $\mathbb{Z}_{>0} \cup \{\infty\}$ and has distribution, for $i \in \mathbb{Z}_{>0}$:

$$P(\Delta^{(a)}_{f,M} = i) = P(\bar{f}(X^{(0)}_{f,M}) \neq a \wedge \ldots \wedge \bar{f}(X^{(i-2)}_{f,M}) \neq a \wedge \bar{f}(X^{(i-1)}_{f,M}) = a),$$
$$P(\Delta^{(a)}_{f,M} = \infty) = 1 - \sum_{i\in\mathbb{Z}_{>0}} P(\Delta^{(a)}_{f,M} = i).$$

The notion of a period allows us to reason about bounded fairness in the context of probabilistic models. We can say an induced MC is bounded fair if its period (and first occurences) lies within certain bounds with probability one. In Subsection IV-C we will extend bounded fairness to models with non-determinism *and* probabilistic transitions. We will define bounded fairness for MDPs by finding the class of schedulers that induces exactly the set of MCs whose period is bounded with probability one. Note that this probability depends both on the inherent probabilistic transitions in the MDP as well as the, possibly probabilistic, scheduler choices. Before we continue, we review some pragmatic solutions to the problem of blocking described in Example 1.

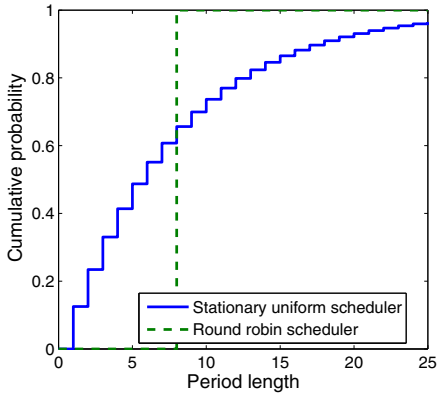


Fig. 1. Cumulative distribution of the period $\Lambda^{(k)}$ for a round-robin and the stationary uniform scheduler.

### B. Pragmatic solutions

A different, more pragmatic, approach to fairness problems is to consider specific instances, especially by fixing a particular scheduler and then analyzing the induced model [5], [17]. Two different kinds of schedulers are used widely: *round-robin* schedulers, which fix a particular execution order for the processes and force the distributed algorithm to adhere to this order, and the *randomized* scheduler, which assigns equal probabilities to each enabled action for every path. These types of schedulers also occur in the form of hidden assumptions in simulation environments for distributed algorithms [1], [25], [10].

*Definition 6:* For an enabled MDP $M = (S, A, R)$ of a distributed algorithm with $N$ processes, the *stationary uniform*, or randomized, scheduler $f$ for $M$ picks each action $a \in A$ with equal probability, for all paths $\sigma$: $P(f_R(\sigma) = a) = 1/N$.

Scheduler $f$ for $M$ is a *round-robin* scheduler if a bijection $g : A \to [0, N-1]$ exists which defines an order on the actions of $M$ such that for any finite path $\sigma$ of length $k$ and any action $a \in A$ we have:

$$P(f(\sigma) = a) = \begin{cases} 1, & \text{if } k \bmod N = g(a) \\ 0, & \text{otherwise.} \end{cases}$$

For a round-robin scheduler we find $\Lambda^{(k)} = N$ for all time-points $k$. While this is well suited for symmetric hardware systems without drifting clocks, we consider this a too strict assumption for a distributed algorithm. Since perfect clocks are unrealistic one would need a reliable clock synchronization service on which the algorithm runs. Though this is feasible, it is expensive, and limits the application domain of distributed algorithms.

For a stationary uniform scheduler we find that $\Lambda^{(k)}$ is geometrically distributed with parameter $1/N$ for all time-points $k$. We feel that there is no good reason to assume that the period of a distributed algorithm is geometrically distributed. For example the variance of a geometrically distributed random variable with parameter $1/N$ is $N^2 - N$, which is highly unrealistic in practical applications. Figure 1 compares the

distribution of the period $\Lambda^{(k)}$ for round-robin and stationary uniform schedulers. We can see from this figure that the round-robin schedulers are completely deterministic with respect to period length, while the stationary uniform scheduler allows components to act multiple times in a row or not at all for a long period of time, with some probability. The advantage of using a stationary uniform scheduler is that it is memoryless and therefore easy to analyze or simulate.

### C. Bounded fairness for MDPs

We extend the notion of bounded fairness to MDPs by restricting the schedulers such that the period length of the induced MC is bounded both from above and from below. Given a path of length $k$, $\sigma = s_1 a_1 s_2 a_2 \dots s_k a_k s_{k+1}$ we write $A_\sigma^{(i)}$ for the set of all actions that occur in the last $i$ steps of the path if $i \leq k$. In case $i > k$, $A_\sigma^{(i)}$ is undefined:

$$A_\sigma^{(i)} = \begin{cases} \{a_j \mid k - i < j \leq k\}, & \text{if } i \leq k \\ \mathit{undefined}, & \text{if } i > k. \end{cases}$$

*Definition 7:* Consider an enabled MDP $M$ with $|A| = N$ and bounds $L, U \in \mathbb{Z}$ with $1 \leq L \leq N$ and $N \leq U$. We define the class $C_{[L,U]}$ of $[L,U]$ *bounded fair* schedulers characterized by the function $F : ((S \times A)^* \times S) \to 2^A$, which describes which actions are allowed for any finite path $\sigma \in ((S \times A)^k \times S)$ of length $k \in \mathbb{Z}_{\geq 0}$.

$$F(\sigma) = \begin{cases} A \setminus A_\sigma^{(U-1)}, & \text{if } k \geq U \wedge |A \setminus A_\sigma^{(U-1)}| = 1 \\ A \setminus A_\sigma^{(k)}, & \text{if } k < L \\ & \text{or } k < U \wedge |A \setminus A_\sigma^{(k)}| = U - k \\ A \setminus A_\sigma^{(L-1)}, & \text{otherwise.} \end{cases}$$

The intuitive meaning of function $F$ defined in Definition 7 is as follows. For paths longer than $U-1$ we have that, if a particular action did not occur in the last $U-1$ steps, then we must pick this action to ensure the period does not exceed $U$ (first case). If there is no such action, we must pick an action that has not appeared in the last $L-1$ steps to ensure the period is at least $L$ (last case).

For paths shorter than $L$ we must pick an action that has not occurred yet (second case, first condition); this avoids that the first period is less than $L$. Finally we have for paths of length $k < U$ that, if the number of actions that have not yet occurred equals $U - k$, then we must schedule one of these missing actions (second case, second condition). This ensures that we will never arrive in a situation in which multiple actions have not occurred in the last $U-1$ steps. We now give an example of how the function $F$ works to ensure bounded fairness.

| $\sigma[A]$ | $F(\sigma)$ | $\sigma[A]$ | $F(\sigma)$ |
|---|---|---|---|
| $\epsilon$ | $\{a,b,c,d\}$ | $abac$ | $\{d\}$ |
| $a$ | $\{b,c,d\}$ | $abacd$ | $\{a,b,c\}$ |
| $ab$ | $\{a,c,d\}$ | $abacda$ | $\{b\}$ |
| $aba$ | $\{c,d\}$ | $abacdab$ | $\{a,c,d\}$ |

*Example 3:* Consider the enabled MDP $M = (S, A, R)$ of a distributed algorithm with 4 processes, and action set $A = \{a, b, c, d\}$. On the right we list the action traces of several example paths and the corresponding sets of actions allowed by the function $F$, which characterizes $[2,5]$ bounded fairness as defined in Definition 7. We can see that for this particular path we have that the first three period lengths are 2, 5, and 3.

The following theorem shows the relationship between $[L,U]$ bounded fairness and the periodicity of the MDP.

*Theorem 2:* Given an enabled MDP $M = (S, A, R)$ with $|A| = N$ and given bounds $L, U \in \mathbb{Z}$ with $1 \leq L \leq N$ and $N \leq U$, a scheduler $f$ is $[L,U]$ bounded fair if and only if for all time-points $k \in \mathbb{Z}_{\geq 0}$ the period of the induced MC at time-point $k$ lies between $L$ and $U$ and for any action $a$ the first occurrence of $a$ is less than or equal to $U$:

$$P\left(\Lambda_{f,M}^{(k)} \in [L,U]\right) = 1 \wedge P\left(\Delta_{f,M}^{(a)} \leq U\right) = 1.$$

The proof can be found in an extended version of this paper [12].

### D. Complexity

Definition 7 shows that the class of $[L,U]$ bounded fair schedulers is induced by a function $F$ which gives, for each path, the allowed actions. From Theorem 1 we know that we can construct an MDP $M'$ to compute extremal reachability probabilities. In principle, the MDP $M'$ is infinitely large, as its state space consists of all paths of $M$. However, the proof of Theorem 1 depends on the fact that we can determine from the current path ($\sigma$) of $M$, which actions ($F(\sigma)$) are allowed. It turns out that if we consider classes of bounded fair schedulers we need less information to determine the allowed actions $F(\sigma)$.

We now reason about what information about the current path $\sigma$ is actually necessary to decide which actions are allowed. We can see, from Definition 7, that for each action $a_i \in A$ we must know whether $a_i$ is in one of the sets $A \setminus A_\sigma^{(U-1)}$, $A \setminus A_\sigma^{(k)}$, and $A \setminus A_\sigma^{(L-1)}$, where $k < U$. To establish this fact, it is enough to know how many steps ago in $\sigma$ each action $a_i$ occurred. We denote this distance as $d(\sigma, a_i)$ which is defined recursively, where $s \in S$ and $a_j \in A$:

$$d(\sigma, a_i) = \begin{cases} 0, & \text{if } \sigma = s \\ d(\sigma', a_i) + 1, & \text{if } \sigma = \sigma' a_j s \wedge a_i \neq a_j \\ 1, & \text{if } \sigma = \sigma' a_j s \wedge a_i = a_j \end{cases}$$

We now have that $a_i \in A \setminus A_\sigma^{(k)}$ if and only if $d(\sigma, a_i) \leq k$. Since we consider only $[L,U]$ bounded fair schedulers, all paths $\sigma$ for which we have $d(\sigma, a_i) > U$ occur with probability zero. Finally, we must also take care of the corner cases where the length of path $\sigma$ is less than $L$ or less than $U$. To do this we record the length of the current path $\sigma$ up to $U$ recursively, where $s \in S$ and $a \in A$:

$$k(\sigma) = \begin{cases} 0 & \text{if } \sigma = s \\ k(\sigma') + 1 & \text{if } \sigma = \sigma' a s \text{ and } k(\sigma') < U \\ k(\sigma') & \text{if } \sigma = \sigma' a s \text{ and } k(\sigma') \geq U. \end{cases}$$

Since the functions $d(\cdot, a)$ and $k(\cdot)$ are defined recursively, it is enough for the MDP $M'$ to keep track of the values of these functions instead of keeping track of the entire path of $M$ followed so far. Given an MDP $M$ of a distributed algorithm with state space $S$ and actions $A$ and given the bounds $L$ and $U$, we find for the size of MDP $M'$ for which the class of all schedulers matches the class of $[L, U]$ bounded fair schedulers of $M$ as per Theorem 1 the following:

$$|S'| = O(|S| \cdot U^{|A|+1}).$$

For each pair of state and action of $M$, we must keep track of the amount of steps since the last occurrence of the action (the functions $d(\cdot, a_i)$). This leads to $|S| \cdot U^{|A|}$ states. We further annotate each state with the length of the current path (the function $k$), where we count only up to $U$. This gives $|S| \cdot U^{|A|} \cdot U = |S| \cdot U^{|A|+1}$ states. However, not every combination will be reachable with probability greater than zero. This show that the size of $M'$ grows polynomially in $U$ (for constant $|A|$) and exponentially in $|A|$.

The practical problem of describing the MDP $M'$ given a description of the MDP $M$ in the PRISM language is extensively discussed in an extended version of this paper [12].

## V. Case studies

In this section we illustrate the use of bounded fairness for distributed algorithms with probabilistic faults by applying our approach to two simple case studies. For each algorithm we compute bounds for the probability that the system is in a *safe* configuration, i.e., its maximum and minimum *availability*. We consider transient availabilities (the probability to be in a safe state at a certain time-point) and long-run average availabilities (the expected percentage of time to be in a safe configuration if the system runs forever). Since we are interested in studying the effect of non-determinism arising from the interleaving of processes we have fixed, for each case study, a single starting state. We have used the PRISM model-checker [23] to compute transient availabilities and a semi-symbolic algorithm to compute long-run average availabilities [31]. All PRISM models are available upon request. A detailed discussion of the modeling assumptions we have made and a description of how to construct the PRISM models can be found in extended version of this paper [12].

### A. Minimal spanning tree algorithm

We consider the distributed minimal spanning tree algorithm (MST) developed throughout the paper. We use a fault model where communication may be garbled (with probability 0.1) or may fail to take place (with probability 0.2). When a node fails to communicate with another (i.e., it is not able to read the value of the distance variable of one of its neighbours), then it simply does not change its state. We consider here a network of four nodes. In the starting state we have that all nodes have their distance parameter set to the maximum value.

In Figure 2 we give best- and worst-case transient *availabilities*, i.e., the figure shows the probability to be in a state in which all processes have correctly computed their distance to the root node at a certain time-point. We consider two different $[L, U]$ bounded fair (BF) scheduler classes, the class of all round-robin (RR) schedulers and the stationary uniform (SU) scheduler, for which we find the unique transient availability directly, instead of a maximum and minimum.

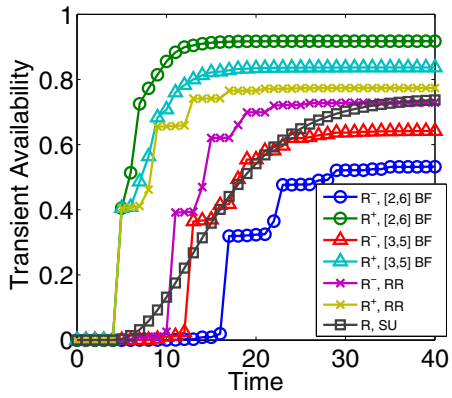


Fig. 2. Transient availabilities for the MST algorithm. Minimum and maximum availabilities are shown for two classes of bounded fair (BF) schedulers and the class of round-robin (RR) schedulers. The transient availability for the stationary uniform (SU) scheduler is also shown.

In the figure we can see the effect of increased scheduler freedom. For each scheduler class, transient availability is initially low, but increases as the nodes exchange information. While the class of RR schedulers is rather restricted, since the nodes always execute in the same order, bounded fairness allows some nodes to act faster than others. In the worst case, nodes that have not yet computed the correct distance from the root are scheduled last, while nodes that have already correctly established their distance from the root are scheduled as soon as possible. We see that this scheduling freedom has a noticeable effect on the bounds for the transient availabilities.

The effect that the algorithm needs several "rounds" to establish stability can be clearly seen for the class of RR schedulers and the classes of BF schedulers by "jumps" in the transient availability curves. For the SU scheduler this important effect cannot be seen as its curve is smooth.

The long-run average availabilities for the different scheduler classes can be seen in figure 4 for both case studies. With increased scheduling freedom, the bounds for the long-run average availability become less tight.

Similar results were found for another case study [12] which considers a gossiping information spread algorithm inspired by [14].

### B. Tree-network leader election algorithm

Finally, we consider a leader election algorithm for tree-structured networks adapted from [16]. The algorithm establishes a directed tree in an anonymous network, by having each node select a "parent" node. This tree then has a unique root node (which selects itself as its parent), which is subsequently

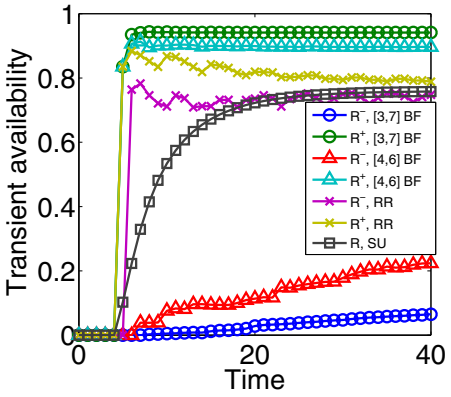


Fig. 3. Transient availabilities for the leader election algorithm. Minimum and maximum availabilities are shown for two classes of bounded fair (BF) schedulers and the class of round-robin (RR) schedulers. The transient availability for the stationary uniform (SU) scheduler is also shown.

| Model | Schedulers | Min | Max |
|---|---|---|---|
| MST | [2,6] BF | 0.553683 | 0.904349 |
| | [3,5] BF | 0.666325 | 0.820220 |
| | RR | 0.743096 | 0.754497 |
| | SU | 0.744122 | 0.744122 |
| Leader | [3,7] BF | 0.127691 | 0.891229 |
| | [4,6] BF | 0.336496 | 0.848096 |
| | RR | 0.763570 | 0.780348 |
| | SU | 0.760131 | 0.760131 |

Fig. 4. Extremal long-run average availabilities for various bounded fair (BF), round-robin (RR), and stationary uniform (SU) classes of schedulers.

elected as leader. The selection of a *unique* node in such anonymous networks, however, depends on the choices made by the underlying scheduler. We embellish the algorithm with local faults that reset the "parent" of a node to a value chosen with a uniform distribution. Such a local fault occurs with a probability of $0.1$ whenever a node executes a local step. We consider a network with five nodes. In the initial state each node has its "parent" variable set to itself.

Figure 3 depicts the probability that one node has been elected as leader. The availabilities are behave similar to the previous case study, although the impact of bounded fair schedulers is greater here. This is most likely caused by the fact that this algorithm is only *weakly* self-stabilizing [16]. This means that there are schedulers for which the algorithm is not guaranteed to reach a safe configuration. In our setting, where we have transient faults and where we consider bounded fairness we can clearly see that there are bounded fair schedulers which greatly decrease the availability of the algorithm.

We also see that the bounds of the transient availability drop in time for the RR scheduler class. This is due to the fact that with increasing time we have a higher probability of observing faults.

## VI. Conclusion

This paper has discussed a subtle point in the quantitative analysis of probabilistic models of distributed algorithms. Starting off with the observation that classical fairness notions are too permissive, and thus unsuitable for quantitative verification, we have focussed on the notion of bounded fairness. Bounded fairness naturally captures clock drift in near-symmetric distributed systems. We have shown how to apply this new fairness notion to models of distributed algorithms and discussed properties of our scheduler classes. The theory is developed in the context of Markov decision processes.

Via some small case studies we managed to illustrate that customary ways of fixing an execution order, such as round-robin scheduling and random scheduling, may lead to too optimistic or too pessimistic estimations of quantitative properties of distributed algorithms. In particular, we see that random scheduling cannot be used to approximate the transient behavior of distributed algorithms, relative to bounded fairness. On the other hand, we see that bounded fairness increases the model sizes considerably, and this makes model checking-style analysis difficult. For this reason, it is worthwhile to investigate effective approximations of quantitative properties for the class of bounded fair schedulers. Another interesting application of bounded fairness for MDPs lies in the study of quantitative effects of clock drift. One avenue to investigate in this context is whether the resilience to clock drift can be quantified for different distributed algorithms. Another further direction seeded in this work might be the analysis of distributed algorithms which are known to exhibit certain qualitative properties only under a specific subset of schedulers. We then aim at adapting the techniques described in this paper, such as to identify the bounded fair schedulers which maximize or minimize the dependability metrics related to such qualititative properties.

## References


[1] T. R. Andel and A. Yasinsac. On the credibility of Manet simulations. *IEEE Computer*, 39(7):48–54, 2006.

[2] D. Angluin, J. Aspnes, Z. Diamadi, and M. J. F. R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.

[3] P. C. Attie, N. Francez, and O. Grumberg. Fairness and hyperfairness in multi-party interactions. *Distributed Computing*, 6(4):245–254, 1993.

[4] H. Attiya, C. Dwork, N. Lynch, and L. Stockmeyer. Bounds on the time to reach agreement in the presence of timing uncertainty. In *23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 359–369. ACM Press, 1991.

[5] R. Bakhshi and A. Fehnker. On the impact of modelling choices for distributed information spread. A comparative study. In *6th Int'l Conf. on Quantitative Evaluation of Systems (QEST)*, pages 41–50. IEEE CS, 2009.

[6] J. Beauquier, C. Johnen, and S. Messika. All k-bounded policies are equivalent for self-stabilization. In *SSS*, number 4280 in LNCS, pages 82–94. Springer, 2006.

[7] J. Beauquier, C. Johnen, and S. Messika. Brief announcement: Computing automatically the stabilization time against the worst and the best schedules. In *DISC*, number 4167 in LNCS. Springer, 2006.

[8] I. Chatzigiannakis, S. Dolev, S. P. Fekete, O. Michail, and P. G. Spirakis. Not all fair probabilistic schedulers are equivalent. In *Int'l Conf. on Principles of Distributed Systems*, LNCS, pages 33–47. Springer, 2009.

[9] A. Coccoli, P. Urbán, and A. Bondavalli. Performance analysis of a consensus algorithm combining stochastic activity networks and measurements. In *DSN*, pages 551–560. IEEE CS, 2002.
[10] U. M. Colesanti, C. Crociani, and A. Vitaletti. On the accuracy of omnet++ in the wireless sensornetworks domain: simulation vs. testbed. In *PE-WASUN '07*, pages 25–31, 2007.
[11] Z. Collin and S. Dolev. Self-stabilizing depth-first search. *IPL*, 49(6):297–301, 1994.
[12] P. Crouzen, E. M. Hahn, H. Hermanns, A. Dhama, O. Theel, R. Wimmer, B. Braitling, and B. Becker. Bounded fairness for probabilistic distributed algorithms. Reports of SFB/TR 14 AVACS 57, SFB/TR 14 AVACS, April 2010.
[13] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. When birds die: Making population protocols fault-tolerant. In *DCOSS*, volume 4026 of *LNCS*, pages 51–66. Springer, 2006.
[14] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. S. H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *PODC*, pages 1–12. ACM Press, 1987.
[15] N. Dershowitz, D. N. Jayasimha, and S. Park. Bounded fairness. In *Verification: Theory and Practice*, volume 2772 of *LNCS*, pages 304–317. Springer, 2003.
[16] S. Devismes, S. Tixeuil, and M. Yamashita. Weak vs. self vs. probabilistic stabilization. In *28th Int'l Conf. on Distributed Computing Systems (ICDCS)*, pages 681–688, Washington, DC, USA, 2008. IEEE CS.
[17] A. Dhama, O. Theel, P. Crouzen, H. Hermanns, R. Wimmer, and B. Becker. Dependability engineering of silent self-stabilizing systems. In *SSS*, volume 5873 of *LNCS*, pages 238–253. Springer, 2009.
[18] A. Dhama, O. Theel, and T. Warns. Reliability and availability analysis of self-stabilizing systems. In *SSS*, volume 4280 of *LNCS*, pages 244–261. Springer, 2006.
[19] E. W. Dijkstra. Solution of a problem in concurrent programming control. *Commun. ACM*, 8(9):569, 1965.
[20] N. Francez. *Fairness*. Springer, 1986.
[21] E. Gafni and M. Mitzenmacher. Analysis of timing-based mutual exclusion with random times. *SIAM J. Comp.*, 31(3):816–837, 2001.
[22] W. H. Hesselink. Progress under bounded fairness. *Distributed Computing*, 12(4):197–207, 1999.
[23] A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *TACAS*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.
[24] P. Hurni and T. Braun. Calibrating wireless sensor network simulation models with real-world experiments. In *Proc. of the 8th Int'l IFIP-TC 6 Networking Conf.*, pages 1–13. Springer, 2009.
[25] A. Köpke, M. Swigulski, K.Wessel, D.Willkomm, P. Haneveld, T. Parker, O. Visser, H. Lichte, and S. Valentin. Simulating wireless and mobile networks in omnet++: The mixim vision. In *1st Int'l Workshop on OMNeT++*, 2008.
[26] S. Kurkowski, T. Camp, and M. Colagrosso. Manet simulation studies: the incredibles. *MC2R*, 9(4):50–61, 2005.
[27] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *STTT*, 6(2):128–142, 2004.
[28] L. Lamport. Fairness and hyperfairness. *Distributed Computing*, 13(4):239–245, 2000.
[29] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.
[30] H. Völzer. Refinement-robust fairness. In *Int'l Conf. on Concurrency Theory*, volume 2421 of *LNCS*, pages 547–561. Springer, 2002.
[31] R. Wimmer, B. Braitling, B. Becker, E. M. Hahn, P. Crouzen, H. Hermanns, A. Dhama, and O. Theel. Symblicit calculation of long-run averages for concurrent probabilistic systems. In *7th Int'l Conf. on Quantitative Evaluation of Systems (QEST)*, 2010.