Dependable Systems through Region-Adherent Distributed Algorithms

Jan Steffen Becker, Dilshod Rahmatov, Oliver Theel
System Software and Distributed Systems
Carl von Ossietzky University of Oldenburg
Oldenburg, Germany, 26129
Email: theel@informatik.uni-oldenburg.de

Abstract—We present a new class of non-masking fault-tolerant systems. A system of this class may be implemented based on a distributed algorithm that employs redundancy to bound the observable, incorrect behavior of the system in space. With bounding in space, it is meant that the decrease of quality of the service which the system provides to its environment is upper-bounded per fault. We call such an algorithm (resp. the system implementing it) region-adherent. Region-adherence turns out to be an orthogonal concept to self-stabilization – the only known concept for implementing non-masking fault-tolerant systems so far. We illustrate the concept of region-adherence by an example.

I. INTRODUCTION

A dependable system is able to deliver the service it provides to a potential user or second system using it, very robustly. This “robustness” is generally quantified by dependability measures like reliability, limiting availability, mean time to repair or mean time to failure [1]. For example, limiting availability states the probability that at an arbitrary point in time, the system is able to successfully provide its service, i.e., “functions (correctly).” Reliability, on the contrary, states the probability that the system functions at a given time t provided that it has functioned at time 0 and ever since. The higher these probabilities, the more dependable is a system. Clearly, highly dependable systems are very valuable since they can be trusted to provide a correct service “at almost every time” or “for very long time.” Consequently, highly dependable systems should be used for services that are essential for the well-being of individuals or societies or for protecting substantial financial resources. Examples are safety-critical services like collision-avoidance systems of airplanes, breaking systems of trains, power plant control, telephone and computer networks and alike.

When building a dependable system, one must be able to cope with the potential threads to correct system functioning: faults, errors, and failures. A fault is a deficiency that might lead to an error in a system. Examples of faults are radiation modifying computer memory or under-specified problem descriptions. An error, though, is the manifestation of a fault in the state of a system: parts of the system state might be erroneous rendering the state incorrect wrt. the system specification. But still, the system exhibits normal behavior. For example, due to some fault, a program variable now contains a value that has never been calculated by the program. Finally, an error might lead to a (system) failure, meaning that the system’s behavior deviates from the correct behavior as stated in the system’s specification. A system controlling traffic lights at a road crossing, for example, suffers a failure if due to some error, the lights in all directions indicate “green” at the same time.

One way of building a dependable system is by using subsystems in a fault-tolerant manner: the sub-systems used for constructing the system are still subject to faults, errors, and failures, but they are smartly combined such that the resulting system is not “that severely” or not at all impacted by the malfunctioning of (some of) these components. A so-called fault model F specifies the deficiencies of the system, the system components, sub-systems and the environment of the system. Often, the fault model is divided into two parts: a structural and a functional fault model [2]. In the structural fault model, it is stated what components are subject to faults. It might also state how many components are assumed to have maximally (or minimally, etc.) failed at any given time or with what probability a certain components might fail, whether components fail independently of one another or not, etc. The functional fault model, on the contrary, specifies the semantics of a component failure. Failure semantics often assumed are crash failure, omission failure, timing failure, incorrect computation failure, or byzantine failure. In ascending order, this enumeration gives more and more severe, i.e., general, behaviors of a component failing. For example, a components suffering a crash failure simply stops functioning when failing. Prior to the failure it worked perfectly, after the failure not at all. A component suffering a byzantine failure, though, – at the time of failing – violates its specification in some way. It may or may not return to a correct functioning at some time thereafter, still it has failed. Coping with components subject to byzantine failures is often very complicated, due to the various possible behaviors such a components may exhibit and which must be tolerated by the system using it.

Figure 1 shows such a possible composition of a fault-tolerant system in its environment. Assume the fault model $F_1$, that states that each of the two sub-systems may fail with crash failure semantics but not independently: if sub-system A fails (or has failed), then sub-system B does not and vice versa. The sub-systems A and B deliver identical services, for example, reading some data. The overall system
should also provide read access to this data. The design
(an algorithm implementing the interface of the system and
accessing the interfaces for the two sub-systems) realizes the
system’s interface as follows: A read operation invoked at
the system’s interface is executed by any one of the two
sub-systems A or B. Clearly, the system is able to tolerate
the failure scenarios possible under the given fault model.
Furthermore, it completely masks failures of components: if
any single one of the two sub-systems has failed and the
user is nevertheless issuing a read operation, the operation can
successfully and immediately be executed. The literature calls
such a system masking fault-tolerant [3], [4]. The system’s
specification, generally consisting of an intersection of a safety
and liveness property is always honored: the system is always
live and safe. Thus, the system exhibits a reliability of 1.0 at
any given time \(t \geq 0\) as long as the assumptions of the fault
model hold.

Now, let us assume a different fault model \(F_2\) and a
potentially different design of the system but with the same
layout as in Figure 1. In fault model \(F_2\), any of the two
sub-systems is independently allowed to fail with incorrect
computation semantics.\(^1\) In particular, the two sub-systems
may even fail at the same time. Again, the system provides a
read access operation to some data stored in the sub-systems.
Obviously, in the case that the two sub-systems have failed,
the system cannot “right away” return the requested data.
But the system’s design is such that it is guaranteed that the
sub-systems – in case the are failed – are repaired in finite
time, thereby restoring or somehow reconstructing the data.
While the system is repairing its failed sub-systems, invalid
data might be visible at the system’s interface. But due to
the repair, it is eventually replaced by the correct data. Thus,
the user invoking a read operation, might observe incorrect
operation results for some bounded time. The literature calls
such a system non-masking fault-tolerant wrt. to fault model
\(F_2\) [3], [4]. Is is fault-tolerant since, obviously, the failures
are eventually tolerated. It is non-maskingly doing so, since
an incorrect system behavior is observable by the system
environment, i.e., the user in the presented example. Note that
the system’s liveness property is not impacted by failures of
\(F_2\); the system is always live. But the safety property may be
invalided. Fortunately, through the particular system design,
the violation of the system’s safety property is bounded in
time. Self-stabilizing systems [5] belong to this class of non-
masking fault-tolerant systems.

In this paper, we present a new class of non-masking
fault-tolerant systems that we call region-adherent systems.
Contrary to a self-stabilizing system that bounds the violation
of the system’s safety property in time, a region-adherent
system bounds the violation of safety in space.

The paper is structured as follows. In Section II, we state
the concept of region adherence in an informal manner and relate
it to self-stabilization. Section III gives the system model and
a formal definition of region adherence. In Section IV, an
example of a region-adherent system is given. Furthermore,
we formally prove this property for the example. Section V,
finally, draws a conclusion and sketches future work in this
direction.

II. BASIC IDEA OF REGION ADHERENCE

As stated in the introduction and in correspondence with
[3], [4], fault-tolerant systems can be differentiated along
two dimensions: a safety and a liveness property dimension
as shown in Figure 2. In each dimension, two cases are

\[
\begin{array}{|c|c|}
\hline
\text{live} & \text{not live} \\
\hline
\text{safe} & \text{Fail–Stop} \\
\hline
\text{Masking Fault–Tolerant} & \text{Fault–Tolerant} \\
\text{Not Fault–Tolerant} & \text{Not} \\
\hline
\text{not safe} & \\
\hline
\text{Non–Masking Fault–Tolerant} & \\
\hline
\end{array}
\]

\[
\text{Fig. 2. Fault Tolerance Classes}
\]

considered: whether or not the particular property always holds
for a particular system under a particular fault model. If only a
single fault scenario exists that leads to an invalidation of the
liveness (resp. safety) property of the system, then the system
generally does not exhibit liveness (resp. safety), indicated
by “not live” (resp. “not safe”) in the figure. Examples for
masking and non-masking fault-tolerant systems have already
been presented. A system whose safety property is guaranteed
to hold while its liveness property might be compromised, is
called a fail-stop (fault-tolerant) system. Systems of this class
are relevant in application scenarios where safety is of primary
concern, like in vehicles. Here, a vehicle hopefully ceases to
run if the braking sub-systems have failed. In other words:

\(^1\)An incorrect computation failure is a byzantine failure where the compo-
nent responds arbitrary values but does not suffer a crash, omission or timing
failure.
liveness of the system may be invalidated by failures but not safety. If liveness as well as safety may be compromised, then the system is not fault-tolerant. This uninteresting class is only given for completeness.

Let us focus on the class of non-masking fault tolerance. Since the safety property of a system belonging to this class (under a given fault model) may be invalidated, the system – obviously – cannot guarantee correct system behavior as claimed by the system’s specification. Thus, such a system might not be of too much use. In particular, if the invalidation of safety (due to some failures) occurred very early in the lifetime of the system. Without any restoration of the safety property, the system might be completely useless. It might be even “dangerous,” since the violation of safety is observable at the system’s interface to the environment. A user or system invoking the system’s service and not checking its correctness might get serviced in an incorrect way.

Consequently, in order to make use of those systems over a longer period of time – in particular after failures of subsystems have occurred –, the violation of the system’s safety property must be “somehow restricted.”

Self-stabilizing systems are constructed in such a way, that they restrict the violation of safety in time. According to [5], a system is self-stabilizing if and only if:

- starting from any state, it is guaranteed that the system will eventually reach a correct state (convergence), and
- given that the system is in a correct state, it is guaranteed to stay in a correct state, provided that no fault happens (closure).

More precisely, a self-stabilizing system is guaranteed to return to a correct state after a finite number of computational rounds. A round is completed, if all active entities of the system (in a distributed system: all processes participating in providing the system service) have taken at least one computational step in the absence of new faults occurring. Figure 3 shows a typical behavior of a self-stabilizing system over time. Until time $T_1$, the system is in a correct state (and consequently, the safety as well as the liveness property hold). At time $T_1$, though, the system suffers a fault (indicated by the red thunderbolt) leading to an invalidation of the safety property. This is indicated by the “step” in the graph from 100% (safety) to 0% (safety). But since the system is a self-stabilizing one, it is still live and guaranteed to converge to a correct state in finite time. This can be observed in the figure after time $T_2$, where it is gradually “restoring” its safety property (in the absence of further faults). This is indicated by the green arrow.

An alternative to restricting the violation of safety in time is a restriction in space which we propose. But what could that mean?

In Figure 3, a single fault invalidated the safety property of the system. The safety property is of Boolean nature: it either holds or not. For a self-stabilizing system, as long as its safety property is invalidated, not too much can be said about the service it is providing at that time: the service might (by chance) still be correct or “totally incorrect.” Our idea is to restrict the possible decline in system service quality, once the safety property is invalidated due to faults. Informally speaking, we would like to bound the loss of service quality per fault by some $1 > \alpha > 0$ for a non-masking fault-tolerant system ($\alpha$ can be interpreted as a percentage). Such a system behavior is sketched in Figure 4. In the figure, a possible behavior of such a system over time is given. At the beginning, the system is live and safe. After the first fault occurred (indicated by thunderbolt 1), the safety property of the system is invalidated but the service quality of the system (which has been at 100% prior to the first fault) is reduced by not more than $\alpha$. This property holds for all subsequently occurring faults as well. Thus, in contrast to a self-stabilizing system, where no guarantees wrt. service quality can be given at all after the first fault, the system of Figure 4 provides such a guarantee for any number of faults.

Figure 5 shows this system behavior in a different, “topological” manner. Initially, the system, as said, is live and safe. So, it is in some state in the innermost region (the dark pink circle). After the first fault, the system is either (by chance) still in this region or in the region directly around the innermost region. For both these regions, it is guaranteed that the service quality provided by the system is at least $1 - \alpha$ (1 represents 100%). After the second fault, the system is in a state of the three innermost regions and the system service quality is at least $1 - 2 \cdot \alpha$ and so on. Topologically speaking, a single fault is not able to transfer the system from any of those “service quality regions” to any other service quality region but only to
regions where the service quality is at most reduced by $\alpha$ wrt. to the region the system was in when the fault occurred. In this sense its behavior – when unsafe – is bounded in space and the system is adheres to these regions. Consequently, we call such a system a region-adherent non-masking fault-tolerant system. It provides upper bounds of possible negative impacts of faults on the service quality. Knowing the number of faults (leading to errors and failures) enables a system user to judge the quality of the delivered service. If the service quality is too low at a particular point in time, then the user might decide to not use the service any more, since, for example, the correct behavior of subsequent services being based on this service is likely to be endangered.

In the next section, we formally define the concept of region-adherence of a system.

III. DEFINITION OF REGION-ADHERENCE

We perceive a (distributed) system $P$ as a finite set of $n$ processes $\{P_0, \ldots, P_{n-1}\}$. The state of a process is given by the valuation of a Cartesian product of its variables. A Cartesian product of the states of all $n$ processes defines the configuration of the system. W.l.o.g., let $s_i$ be the only local variable of a process $P_i$ and $r_i$ its only global communication register. A global communication register can be read by other processes whereas the local variable $s_i$ cannot. A configuration $c$ is then of the form $(s_0, \ldots, s_{n-1}, r_0, \ldots, r_{n-1})$. We refer to the contents of variable $r$ in a configuration $c$ as $c.r$. The set of all possible configurations of the system is $C$. $c_0$ is the initial configuration of the system.

Every process executes a local algorithm in atomic steps according to the read/write atomicity paradigm [5]. An atomic step leads to a state change transferring the distributed system from the current configuration $c$ to some subsequent configuration $c'$. The set of all possible (fault-free) transitions of configurations according to the algorithm is $A$. It presents the algorithm of the system. We denote a configuration transition due to $A$ from $c$ to $c'$ by $c \rightarrow_A c'$.

The fault model $F$ can be regarded as a specification of all possible configuration transitions from a configuration $c$ to a configuration $c'$ due to a fault of the model, denoted by $c \rightarrow_F c'$. The fault model basically also describes an algorithm that “an enemy to the system” executes at some time, thereby “disturbing the system” by faults (leading to errors). Such a fault step, in our model, is also assumed to be an atomic step.

A nonempty sequence of configurations $\gamma = c_0c_1 \cdots c_n \in C^+$ with $c_0$ being the initial configuration and configuration $c_i, i > 0$, leading to a configuration $c_{i+1}$ by either $A$ or $F$ is called an execution. A configuration is called reachable if there exists a finite execution ending in it.

Now, we can formally define region-adherence of a system.

Definition 1 (Region-Adherence of a System): We assume a system with configurations $C$ and algorithm $A$ under the fault model $F$. Let $g : \gamma \rightarrow [0, 1]$ be a function stating the service quality of the system and let $f$ be a natural number. Algorithm $A$ is called $f$-region-adherent wrt. $g$ and $F$, if and only if for all reachable configurations $c \in C$ and all executions $\gamma = c_0 \cdots c$ ending in $c$ the following holds:

$$g(c) \geq 1 - \# g(\gamma) \cdot \alpha$$

with $\alpha \in R$ and $1/f > \alpha > 0$, where $\# g(\gamma)$ is the number of fault steps of execution $\gamma$. A system executing an $f$-region-adherent algorithm is also called $f$-region-adherent.

Note that an $f$-region adherent system is able to tolerate at least $f$ faults prior to exhibiting a service quality of 0. An $f$-region-adherent algorithm (resp. system) is also called a region-adherent algorithm (resp. system) for short, if the concrete value $f > 0$ is of no interest.

In the next section, we present an example of a region-adherent system.

IV. EXAMPLE: ARITHMETIC MEAN OF SENSOR VALUES

As an example, we assume a sensor network for measuring some phenomenon. Let it be the air humidity. The sensor network consists of $n > 0$ sensor nodes (running a process each) that independently measure the air humidity in some (small) geographical region and send the value to a data sink process at some gateway node. From the scenario, it is expected that the sensor nodes measure identical values. Thus, the measured values inherently are redundant. In case a sensor node works perfectly, only a single node would be required. But, unfortunately, this is not the case. We assume that

1) sensor nodes work independently of each other,
2) a sensor node delivers a sensor value out of an a priori known interval $[l, u]$,
3) even when not failed, a sensor node delivers sensed data with a certain inaccuracy (details below),
4) sensor nodes fail independently with an incorrect computation failure semantics: even after a failure, a sensor node reports an arbitrary value of the $[l, u]$ interval
5) the gateway node is failure-free (for simplicity of presentation).

With these assumptions, it is a good idea to calculate the arithmetic mean of the $n$ reported sensor values and use the resulting value as an estimation of the air humidity in the region. In the following, we will analyze the system wrt. to
region-adherence, thereby proving that the deviation of the estimated value from the true air humidity value is bounded even in the case of failed sensor nodes.

a) Model and Algorithm: We assume \( n \) sensor nodes. Each node \( P_i \) delivers a sensed data value \( v_i, i = 0, \ldots, n - 1 \) for the true air humidity value \( v \). \( v, v_0, \ldots, v_{n-1} \in [l, u] \). The sensor nodes send their values \( v_i \) to the same data sink where the value of \( v \) is estimated by

\[
\bar{v} := \frac{1}{n} \sum_{i=0}^{n-1} v_i
\]  

(2)

b) Fault Model: We assume that every sensor node, when functioning correctly – senses a data value \( v_i \) that deviates by at most \( \epsilon > 0 \) from the true value \( v \). If a sensor node has failed, then it reports arbitrary values of the interval \([l, u]\) to the gateway node, i.e.,

\[
v_i \in \begin{cases} [l, u] & \text{if failed} \\ [v - \epsilon, v + \epsilon] \cap [l, u] & \text{otherwise} \end{cases}
\]

(3)

c) Proof of Region-Adherence: In the following, we give a proof of the region-adherence of the system. In particular, we show that the calculated value \( \bar{v} \) of the algorithm (representing the service delivered by the system) is region-adherent wrt. to \( g \) and \( F \) with \( \alpha = 1/n \) where

\[
g(\bar{v}) := \min \left\{ 1 - \frac{|\bar{v} - v| - \epsilon}{u - l}, 1 \right\}.
\]

(4)

Thus, \( g(\bar{v}) \geq 1 - k \cdot \alpha \) must hold with \( k \) being the number of occurred faults. A fault leads to the failure of a sensor node as described in the fault model. First, we need a lemma.

Lemma 1:

\[
|\bar{v} - v| \leq \frac{k}{n} (u - l) + \frac{n - k}{n} \epsilon.
\]

(5)

Proof: W.l.o.g., we assume that the sensor nodes \( P_0, \ldots, P_{k-1} \) with \( k < n \) have failed. Then,

\[
|\bar{v} - v| = \frac{1}{n} \sum_{i=0}^{n-1} v_i - v
\]

(6)

\[
= \frac{1}{n} \left( \sum_{i=0}^{k-1} (v_i - v) + \sum_{i=k}^{n-1} (v_i - v) \right)
\]

(7)

holds which proves the lemma.

We use the lemma for the proof of region-adherence.

Proof: Having

\[
|\bar{v} - v| \leq \frac{k}{n} (u - l) + \frac{n - k}{n} \epsilon
\]

(8)

\[
|\bar{v} - v| - \epsilon \leq \frac{k}{n} (u - l)
\]

(9)

\[
\frac{|\bar{v} - v| - \epsilon}{u - l} \leq \frac{k}{n}
\]

(10)

\[
1 - \frac{|\bar{v} - v| - \epsilon}{u - l} \geq 1 - \frac{k}{n}
\]

(11)

Because of

\[
0 \leq 1 - \frac{k}{n} = 1 - k \cdot \alpha \leq 1, \quad 0 \leq k \leq n,
\]

(12)

is holds that

\[
g(\bar{v}) \geq 1 - k \cdot \alpha.
\]

(13)

This proves region-adherence of the system.

V. Conclusion and Future Work

In this paper, we presented a new class of non-masking fault-tolerant systems which we call region-adherent systems. The concept of region-adherence of a system is in some sense orthogonal to the concept of self-stabilization: whereas a self-stabilizing system restricts the violation of the system’s safety property in time, a region-adherent system restricts this violation in space. When the number of faults that lead to failures of sub-systems are known, then a lower bound of the service quality provided by the system can be guaranteed. It is then up to the system user to decide, whether such a service quality is still acceptable or not. A region-adherent system, therefore, exhibits a particular – and as we believe very useful – sort of gracefully degrading behavior: namely, it possesses an upper-bounded reduction of service quality per fault. On the other hand, a region-adherent system does not necessarily recover from failures. Since we are only at the beginning of region-adherence-related research, a detailed theory of region-adherence has yet to be derived. In particular, we are interested in the composition of region-adherent systems and their properties. Furthermore, a combination of the concept of region-adherence and self-stabilization seems very promising: such a system would, on one hand side, exhibit a region-adherent behavior while being hit by faults. On the other hand side, the system would recover to full service quality in the course of time. We hope to report on those systems in the future.

References