TOWARDS PARSIMONIOUS RESOURCE ALLOCATION IN CONTEXT-AWARE N-VERSION PROGRAMMING

Jonas Buys, Vincenzo De Florio and Chris Blondia

University of Antwerp, Dept. of Mathematics and Computer Science, Antwerp, Belgium
Interdisciplinary Institute for Broadband Technology, Ghent-Ledeberg, Belgium
{jonas.buys, vincenzo.deflorio, chris.blondia}@ua.ac.be

Keywords: fault-tolerance, *n*-version programming, context-awareness, measurement techniques, dependability.

Abstract

Adopting classic redundancy-based fault-tolerant schemes in highly dynamic distributed computing systems does not necessarily result in the anticipated improvement in dependability. This primarily stems from statically predefined redundancy configurations employed within many classic dependability strategies, which as well known may negatively impact the schemes' overall effectiveness. In this paper, a novel dependability strategy is introduced encompassing advanced redundancy management, aiming to autonomously tune its internal configuration in function of disturbances observed. Policies for parsimonious resource allocation are presented thereafter, intent upon increasing the scheme's cost effectiveness without breaching its availability objective. Our experimentation suggests that the suggested solution can achieve a substantial improvement in availability, compared to traditional, static redundancy strategies, and that tuning the adopted degree of redundancy to the actual observed disturbances allows unnecessary resource expenditure to be reduced, therefore enhancing cost-effectiveness.

1 Introduction

Business- and mission-critical distributed applications are increasingly expected to exhibit highly dependable characteristics, particularly in the areas of reliability and timeliness. Redundancy-based fault-tolerant strategies have long been used as a means to avoid a disruption in the service provided by the system in spite of failures in the underlying software components. Adopting fault-tolerance strategies in dynamic distributed computing systems, in which components often suffer from long response times or temporary unavailability, does not necessarily result in the anticipated improvement in dependability. This primarily stems from statically predefined redundancy configurations employed within many classic dependability strategies, *i.e.* a fixed degree of redundancy and, accordingly, an immutable selection of functionally-equivalent software components, which may negatively impact the schemes' effectiveness from the following angles:

Firstly, a static, context-agnostic configuration may in time lead to a more rapid exhaustion of the available redundancy and therefore fail to properly counterbalance any disturbances affecting the operational status (context) of the components integrated within the dependability scheme. Indeed, the use of replicas of poor reliability can result in a system tolerant of faults but with poor reliability [5]. It is therefore crucial for the system to continuously monitor the operational status of the available resources and avoid the use of resources that do not significantly contribute to an increase in dependability, or that may even jeopardise the schemes' overall effectiveness.

Secondly, redundant implementations imply significantly higher development costs and increased infrastructural requirements. A predetermined degree of redundancy is therefore cost ineffective in that it inhibits to economise on resource consumption in case the actual number of disturbances could be successfully overcome by a lesser amount of redundancy. Reversely, when the foreseen amount of redundancy is not enough to compensate for the currently experienced disturbances, the inclusion of additional resources may prevent further service disruption.

In this paper, a novel, context-aware dependability strategy is introduced encompassing advanced redundancy management. Designed to sustain high availability and reliability, this adaptive fault-tolerant strategy dynamically alters the degree of redundancy and the employed selection of resources.

The remainder of this paper is structured as follows: We first present the concept of *n*-version programming (NVP) schemata in Sect. 2 and show how software design failures in the content failure domain may challenge their effectiveness. A set of ancillary metrics is first set forth in Sect. 3, allowing to deduce knowledge of the context in which the scheme is operating and to detect the proximity of hazardous situations that may require the adjustment of the redundancy configuration. We then move on to elaborate on the internals of the proposed adaptive fault-tolerant strategy in Sect. 4. Policies for parsimonious resource allocation are presented thereafter, intent upon increasing the scheme's cost effectiveness without breaching its availability objective. Sect. 6 reports on the strategy's effectiveness, after which this paper is concluded.

2 *n*-Version Programming with Majority Voting

NVP was first introduced in 1985 as a tool to provide protection against software design faults [5]. The rationale is that deploying multiple functionally-equivalent independently implemented software components will hopefully reduce the probability of a software fault affecting multiple implementations simultaneously, thereby keeping the system operational. An *n*-version composite constitutes a client-transparent repli-

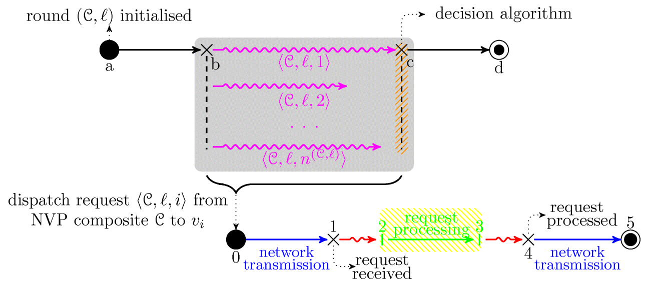
round $(\mathcal{C},\ell)$ initialised
decision algorithm
a
b
c
d
$\langle \mathcal{C},\ell,1\rangle$
$\langle \mathcal{C},\ell,2\rangle$
. . .
$\langle \mathcal{C},\ell,n^{(\mathcal{C},\ell)}\rangle$
dispatch request $\langle \mathcal{C},\ell,i\rangle$ from NVP composite $\mathcal{C}$ to v_i
0
1
2
3
4
5
network transmission
request received
request processing
request processed
network transmission

Figure 1: State transition diagram of voting round (c,ℓ) and the underlying invocations ‹c, ℓ, i› of versions v_i.

cation layer in which all *n* programs, called versions, receive a copy of the user input and independently perform their computations in parallel. Let $\{\ell_x\}_c$ be a sequence of monotonically increasing, strictly positive integer indices $\ell_x = x$ in $L = \mathbb{N}^+$, such that each voting round, *i.e.* a single invocation of an NVP composite c, is uniquely identified. As shown in Fig. 1, the arrival of a request message at the composite interface will trigger the initialisation of a new voting round (c,ℓ) with ℓ the next element in $\{\ell_x\}_c$. Immediately after, the system is to retrieve the redundancy configuration to be used throughout the newly initialised voting round (c,ℓ), *i.e.* the amount of redundancy used and, accordingly, a selection of functionally-equivalent versions (FeV) (transition from state *a* to *b*). We define V as the set of all FeV in the system. For a given round (c,ℓ), the amount of redundancy used within the NVP scheme is denoted as $n^{(c,\ell)} \geq 1$, such that the versions employed for round (c,ℓ) are contained within $V^{(c,\ell)} \subseteq V$ and $n^{(c,\ell)} = |V^{(c,\ell)}|$. Having acquired the redundancy configuration, the request message payload is then replicated and forwarded to each of the selected versions $v_i \in V^{(c,\ell)}$, for $i \in \{1, ..., n^{(c,\ell)}\}$. Each such invocation of a version v_i can be uniquely identified by the tuple ‹c, ℓ, i›. As soon as a result is available for each of the versions involved, the transition from state *b* to *c* fires and a majority voting (MV) procedure is called so as to adjudicate the result of the scheme, which is then returned to the client.

The effectiveness of a scheme such as NVP/MV is largely determined by its redundancy configuration and its ability to counterbalance the disturbances ensuing from the environment in which it operates and to which it is subject. For instance, an NVP/MV scheme can mask failures affecting the availability of up to a minority of its $n^{(c,\ell)}$ versions.

The essential part of any voting procedure is the construction of a partition $\wp^{(c,\ell)} = \{P_F^{(c,\ell)}, \dot\bigcup_{j\in\{1,\dots,k^{(c,\ell)}\}} P_j\}$ of the set of versions $V^{(c,\ell)}$. The notation of a disjoint union to represent the consensus blocks as part of this partition has been taken from [1]. This partitioning procedure is influenced by the disturbances that affected any of the requests ‹c, ℓ, i› involved during the voting round (c,ℓ). Throughout this paper, the notion of disturbance is used to denote the event of a single request ‹c, ℓ, i› struck by the some type of failure, resulting in the perturbation and, consequently, the (temporary) unavailability of the service that version v_i is expected to provide.

The application of NVP schemata in contemporary distributed computing systems is assumed to exhibit the properties of a timed asynchronous system model [2, 6]. For such environments, several types of potential disturbances were defined in the failure manifestation model provided in [2]. We will now elaborate on a subset of these disturbances that may ensue from the activation of a software design fault and manifest within the content failure domain. Such types of disturbances exhibit transient behaviour and emerge exclusively from the activation of a latent software design fault along the execution path followed whilst version v_i is processing some request ‹c, ℓ, i›, *i.e.* when the request is in the *request processing* state (see Fig. 1). One may distinguish two types of failures, each with different repercussions on the generated partition $\wp^{(c,\ell)}$.

Firstly, despite sharing a common specification, FeV may exhibit discrepancies resulting in response value failures (RVF) [4]. Such failures will usually not make the system appear to fail, as the content of the response returned via the service interface is syntactically correct, though not compliant to the functional specifications. During voting, a partition $\dot\bigcup_{j\in\{1,\dots,k^{(c,\ell)}\}} P_j$ is constructed for all versions $v_i \in V^{(c,\ell)}$ that returned a syntactically valid response, allotting versions that reported equivalent results to the same equivalence classes P_j.

Secondly, the activation of a design fault may cause an abrupt interruption of the flow of execution due to an exception – a case of so-called erroneous value failures (EVF). Falling within the content failure domain, a syntactically invalid response message will be returned for the affected invocation, containing the serialised exception thrown. Versions affected by an EVF are classified in $P_F^{(c,\ell)}$. EVF failures may overrule previously activated RVF failures affecting an invocation [2].

Adhering to the discrete-event model developed in [2] for the injection of transient disturbances in NVP schemata, these two disturbance types will be used to assess the performance of the dependability strategy introduced in Sect. 4.

3. Redundancy Configuration Effectiveness

Let $P^{(c,\ell)}$ be the set of largest cardinality in the generated partition $\wp^{(c,\ell)} \setminus P_F^{(c,\ell)}$. Then $c_{max}^{(c,\ell)} = |P^{(c,\ell)}|$ represents the largest consent found amongst the versions in $V^{(c,\ell)}$ within the scope of (c,ℓ). In order for the scheme to adjudicate a result *o*, there should be a consensus amongst an absolute majority of the $n^{(c,\ell)}$ versions, *i.e.* $c_{max}^{(c,\ell)} \geq m^{(c,\ell)}$, with

$$m^{(c,\ell)} = \left\lceil n^{(c,\ell)}/2 \right\rceil. \qquad (1)$$

Put differently, $m^{(c,\ell)}$ is indicative of the smallest degree of consent needed for a consensus block $P^{(c,\ell)}$ to qualify for the equivalence class [*o*]. Consequently, a scheme is resilient to withstand disturbances affecting up to a minority $n^{(c,\ell)} - m^{(c,\ell)}$ of the $n^{(c,\ell)}$ versions used throughout round (c,ℓ).

We now introduce a new metric defined as $c_{max}^{(c,\ell)} - m^{(c,\ell)}$ that was designed to provide a quantitative estimation of how closely the current amount and selection of resources matched the observed disturbances — by shortcoming or excess. More specifically, it can be used to deduce a measure of the proximity of hazardous situations that may necessitate the adjustment of the currently employed redundancy configuration. One can easily see that $\mathrm{Ran}(c_{\max}^{(c,\ell)} - m^{(c,\ell)})$ lies in $[-m^{(c,\ell)}, n^{(c,\ell)} - m^{(c,\ell)}]$, for $c_{\max}^{(c,\ell)}$ was defined in $[0, n^{(c,\ell)}]$. It provides an indirect estimation of the shortage or abundance of redundancy with respect to the disturbances affecting round (c,ℓ): a positive value essentially quantifies how many versions exist in excess of the mandatory $m^{(c,\ell)}$ versions that collectively constitute a majority for round (c,ℓ). For negative values, the absolute value represents the lack of consent relative to $P^{(c,\ell)}$ that would be required so as to constitute a majority. Such a value is interpreted as a symptom that the currently experienced disturbances cannot be successfully counterbalanced by the redundancy configuration used, and the scheme would fail to guarantee the availability of the service it seeks to provide despite its fault-tolerant nature. A critically low value 0 represents a situation for which the majority was attained by only $m^{(c,\ell)}$ versions: the available redundancy was completely exhausted to counterbalance the maximal number of disturbances the scheme could tolerate and any additional disturbance would have led to failure.

4. A Novel Adaptive Fault-Tolerant Strategy

We introduce our adaptive NVP-based fault-tolerant strategy (A-NVP) and elaborate on the advanced redundancy management it supports. Our context-aware reformulation of the classical NVP approach encompasses two complementary parameterised models that jointly determine the optimal redundancy configuration to be used throughout a newly initialised voting round (c,ℓ) in view of disturbances and how these were perceived to have affected the system context. The redundancy configuration is retrieved whilst preparing to fire the transition from state *a* to *b*, as shown in Fig. 1.

The first model is responsible for determining the appropriate degree of redundancy $n^{(c,\ell)}$ to be used and will allow economising on resource expenditures whenever it can be argued safe to do so. Intent upon increasing the scheme's cost effectiveness without breaching its dependability objective, policies for parsimonious resource allocation are presented in Sect. 5. Next, the replica selection model will establish an appropriate set of resources $V^{(c,\ell)}$ maximising the objectives expressed by means of a set of application-agnostic properties [3].

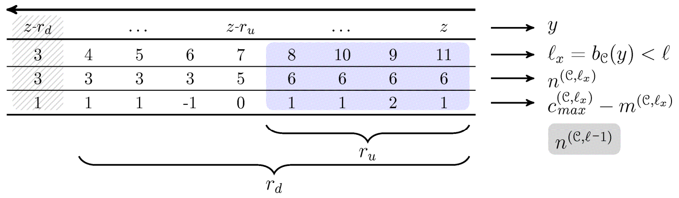
z-r_d
...
z-r_u
...
z
3 4 5 6 7 8 10 9 11
3 3 3 3 5 6 6 6 6
1 1 1 -1 0 1 1 2 1
y
$\ell_x = b_{\mathbb{C}}(y) < \ell$
$n^{(\mathbb{C},\ell_x)}$
$c_{max}^{(\mathbb{C},\ell_x)} - m^{(\mathbb{C},\ell_x)}$
$n^{(\mathbb{C},\ell-1)}$
r_u
r_d

Figure 2: View on window-based data structure before $n^{(c,\ell)}$ is set for use within a newly initialised round (c,ℓ).

4.1. Application-Specific Requirements

The optimal redundancy configuration is not only determined by the context property introduced in Sect. 3, but also by the characteristics of the application itself, or the environment in which it operates. Some applications may operate in a resource-constrained environment. The A-NVP algorithm was conceived to take these application-specific intricacies into account, in that both models can be configured by means of a set of user-defined parameters. Firstly, a parameter $n_{\max} \geq 1$ can be used to set an upper bound on the number of replicas to be used. Contrariwise, parameter $2e + 1 = n_{\min} \geq 1$ can be used to set a lower bound on the degree of redundancy to be used, such that the scheme is guaranteed to tolerate at least *e* disturbances. Lastly, parameter n_{init} will set the default degree of redundancy to be initially used, *i.e.* $n^{(c,\ell)} = n_{\mathrm{init}}$ for $\ell = 1$.

4.2. Redundancy Dimensioning Model

Given the FeV in V, this model is responsible for autonomously adjusting the degree of redundancy employed such that it closely follows the evolution of the observed disturbances. In the absence of disturbances, the scheme should scale down its use of replicas and avoid unnecessary resource expenditure. Contrarily, when the foreseen amount of redundancy is not enough to compensate for the currently experienced disturbances, it will dynamically revise that amount and enrol additional resources if available. The model determines $n^{(c,\ell)}$ upon initialisation of round (c,ℓ), such that $n_{\min} \leq n^{(c,\ell)} \leq min(n_{\max}, |V|)$. Ideally, the redundancy configuration would involve a number $n^{(c,\ell)}$ of versions consistently greater than or equal but close to $cr(e^{(c,\ell)})$, *i.e.* the minimal amount of redundancy required during the operational time interval for a voting round (c,ℓ) in order to stay dependable and have the decision algorithm return the correct result in spite of being challenged by a number of disturbances $e^{(c,\ell)}$, which can be expressed as the function $cr : \mathbb{N}_0 \rightarrow \mathbb{N}^+$ such that $cr(e^{(c,\ell)}) = 2\, e^{(c,\ell)} + 1$ [2].

4.2.1. Window of Context Information

Let $\{y_z\}_c$ be a monotonically increasing sequence of strictly positive integer indices $y_z = z$ in $Y = \mathbb{N}^+$ such that each consecutive completion of some voting round (c,ℓ) originating from an invocation of c is uniquely identified by the next element y in $\{y_z\}_c$. Observe how z denotes the number of completed voting rounds appertaining to c. The bijection $b_c : Y \rightarrow L$ defines the correlation between the terms in either of the sequences $\{y_z\}_c$ and $\{\ell_x\}_c$. The order in which rounds complete, *i.e.* the sequence $\{y_z\}_c$, is not necessarily the order in which these rounds have been initialised, represented by the sequence $\{\ell_x\}_c$. Indeed, in large-scale distributed computing environments, one may expect a significant amount of variability in the response time of an invocation on the NVP composite, due e.g. to its dynamically changing redundancy configuration used for different voting rounds. More information on timeliness issues can be found in [2].

In order to make an informed decision on the amount of redundancy $n^{(c,\ell)}$ to be used, the model will consider the course of the amount of redundancy used for previously completed voting rounds and whether or not the selected redundancy proved to be sufficient to guarantee the scheme's availability. As such, the context manager within the NVP composite will maintain a window-based data structure to store this contextual information for each y in the subsequence $\{y_{\min(1,z-r_d+1)}, \ldots, y_z\}$ of $\{y_z\}_c$. More specifically, for each completed voting round y, the context properties of interest are: the corresponding identifier $\ell_x = b_c(y)$, the amount of redundancy $n^{(c,\ell_x)}$ that was employed throughout its execution, and the extent to which this redundancy was found to be capable of masking disturbances — *cf.* Sect. 3.

Let r_d, r_u and r_f be numbers in $\mathbb{N}^+$ such that $1 \leq r_f \leq r_u \leq r_d$. Number r_d represents the minimum number of consecutive successful voting round completions before contemplating scaling down the current level of redundancy. In line with Sect. 2, a given voting round (c,ℓ) is observed to have completed successfully if a sufficiently large degree of consent could be found between the responses acquired from the subordinate invocations $\langle c, \ell, i \rangle$ of the involved versions $v_i \in V^{(c,\ell)}$ such that a majority could be found, *i.e.* $c_{max}^{(c,\ell)} - m^{(c,\ell)} \geq c_{sm}$, with a discretionary safety margin c_{sm} defined as an integer in $[0, n^{(c,\ell)} - m^{(c,\ell)}]$ **(SC)**. As shown in Fig. 2, r_d imposes an upper bound on the maximum window capacity maintained by the data structure. It is assumed that shorter window lengths may result in incautious downscaling of the redundancy, which in itself might lead to failure of the voting scheme in subsequent voting rounds. Contrariwise, one may reasonably expect that the redundancy scheme is less likely to fail due to the downscaling of the employed degree of redundancy for larger values of r_d, at the expense of postponing the relinquishment of excess redundancy [3].

The number r_u expresses the maximum number of successive voting round completions that failed to meet the criterion of success as defined hereabove, before responding by considering the use of additional redundancy. Such scenario includes (1) voting rounds for which a result *o* could be adjudicated, yet the cardinality $c_{max}^{(c,\ell)}$ of the corresponding consensus block $[o]$ was insufficient to match the agreed safety margin c_{sm}, and (2) rounds for which the decision algorithm failed to adjudicate a result — a case the model will endure at most r_f times within the observation window in spite of undershooting the required amount of redundancy. At risk of prolonging the scheme's unavailability, temporarily refrainning from increasing the employed degree of redundancy after observing a potentially hazardous situation might allow the replica selection model to regain the scheme's intended dependability by substituting poorly performing versions by more reliable, idling versions. Smaller values r_u would enable a more rapid detection of potentially hazardous situations, resulting in system resources to be allocated rather lavishly.

The data structure shown in Fig. 2 is updated upon each subsequent completion y of a voting round, as represented by the corresponding element in $\{y_z\}_c$. Each column refers to a single voting round $\ell = b_c(y)$ and holds information regarding the redundancy $n^{(c,\ell)}$ employed and its effectiveness to counterbalance the disturbances to which it was perceived to be subject — information that can be deduced from the generated partition $\wp^{(c,\ell)}$. Information pertaining to round $b_c(y_{z-r_d})$ will be discarded for values $z > r_d$ (see left in Fig. 2).

4.2.2. Window Semantics

We will now elaborate on the procedure used to determine the amount of redundancy to be used throughout a newly initialised voting round (c,ℓ). In doing so, we use the abstract notion of window semantics S_c to epitomise the specific conditionalities and correlational techniques that enable the redundancy dimensioning model to deduce the optimum degree of redundancy matching the scheme's operational context from the stored information. In this capacity, S_c defines two ancillary functions describing a relation $L \rightarrow \mathbb{N}^+$. More specifically, the upscaling function $f_u(c,\ell)$ is responsible for determining if and to what extent the current level of redundancy $n^{(c,\ell-1)}$ should increase, whereas the downscaling function $f_d(c,\ell)$ quantifies the extent by which $n^{(c,\ell-1)}$ should be lowered. The final degree of redundancy $n^{(c,\ell)}$ to be used for the continuation of (c,ℓ) is then resolved as follows:

$$n^{(c,\ell)} \begin{cases} \mathit{min}(\mathit{min}(n_{max}, |V|), n^{(c,\ell-1)} + f_u(c,\ell)) & | \; f_u(c,\ell) > 0 \quad (2a) \\ \mathit{max}(n_{min}, n^{(c,\ell-1)} - f_d(c,\ell)) & | \; f_d(c,\ell) > 0 \quad (2b) \\ n^{(c,\ell-1)} & | \; \text{else} \quad (2c) \end{cases}$$

The above Eq. (2a) and (2b) formalise the upscaling, resp. downscaling procedure for $\ell > 1$. Observe how the adjustment of the redundancy is constrained by the application-specific parameters n_{min} and n_{max}, as well as by the amount $|V|$ of resources available. Any specific S_c should implement these two functions such that a value is returned only if the window contains information abiding the success criterion **(SC)** and the definitions given for r_d, r_u and r_f hereabove.

The optional safety margin c_{sm} expresses the amount of consent supplementary to the mandatory $m^{(c,\ell)}$ required for the successful adjudication of a result. It serves as a parameter to the redundancy dimensioning model, primarily aiming to reduce the likelihood that the downscaling procedure itself would result in failure of the scheme in the first few subsequent voting rounds. Moreover, such safety margin could anticipate a shortfall in redundancy when the effectiveness of the employed redundancy is observed to exhibit a decreasing trend and proactively trigger the upscaling procedure. Either way, the underlying rationale for maintainning a slightly higher degree of redundancy stems from the assumption that the environment behaves unpredictably and the number of disturbances $e^{(c,\ell)}$ it brings about affecting ongoing voting rounds ℓ, therefore, may vary considerably. In contrast, the dimensioning model was designed to gradually adjust the degree of redundancy downwards, targeting $cr(e^{(c,\ell)})$, in line with the trend perceived from the data held within the observation window. The safety margin can therefore intuitively be seen as the maximum aberration in terms of additional disturbances that the scheme can tolerate as compared with the observed trend **(SM)**.

4.3. Replica Selection Model

Having established the degree of redundancy $n^{(c,\ell)}$ to be employed throughout round (c,ℓ), the replica selection model will then determine an adequate selection of versions $V^{(c,\ell)}$ to be used by the redundancy scheme. The proposed model has been designed so as to achieve an optimal trade-off between dependability as well as performance-related objectives such as load balancing and timeliness. To do so, a score is computed for all versions $v \in V$ depending on the contextual information accrued during previously completed voting rounds. For more information on this procedure, we refer to its initial announcement in [3], which also introduces a valuable metric named normalised dissent for approximating the reliability of versions $v \in V$. Moreover, it aims to mitigate the adverse effects of employing inapt resources that consistently perform poorly and that, consequentially, may threaten the effectiveness of the overall redundancy scheme. If the degree of redundancy $n^{(c,\ell)}$ to be utilised follows a constant or decreasing trend, then, depending on the availability of eligible versions that can be used as a substitute, the model will be successful in excluding such inapt replicas.

5. Parsimonious Resource Allocation

Building on the extensible and abstract concept of window semantics, we will now briefly discuss two specific implementations that will be used to validate the effectiveness of the redundancy dimensioning model in Sect. 6.

A first strategy was originally published in [7]. Even though it was applied on redundant data structures, it can readily be reused for dynamically determining the redundancy level. It assumes $n_{max} = 9$, and will only report an odd degree of redundancy. Moreover, the redundancy scheme is initialised such that it is capable of tolerating at least one failure, hence $n_{min} = 3$. If the voting scheme failed to find consensus amongst a majority of the replicas involved during the last completed voting round, the model will increase the number of redundant replicas to be used in the next voting round, to the extent that $f_u(c,\ell) = 2$. Conversely, when the scheme was able to produce an outcome with a given amount of redundancy for a certain amount r_d of consecutive voting round completions, a lower degree of redundancy shall be used for the next voting round, with $f_d(c,\ell) = 2$. Note how this strategy restricts the success criterion **(SC)** by requiring the same amount of redundancy to be used in the observation interval.

A second strategy is a plain implementation of the mechanism defined in Sect. 4.2.1. When the degree of redundancy $n^{(c,\ell-1)}$ is found to be overabundant, as can be perceived from the success criterion **(SC)**, the downscaling function $f_d(c,\ell)$ will try to adjust downwards by an amount of $(c_{max}^{(c,\ell_x)} - m^{(c,\ell_x)}) - c_{sm}$, with $\ell_x = b_c(y_z)$ the last completed round observed. Undershooting the safety margin will cause $f_u(c,\ell)$ to return $c_{sm} - | c_{max}^{(c,\ell_x)} - m^{(c,\ell_x)} |$, with ℓ_x the round in the observation window delimited by r_u exhibiting the largest deviation from the imposed security margin. The upscaling function $f_u(c,\ell)$ will try to inflate $n^{(c,\ell-1)}$ by $| c_{max}^{(c,\ell_x)} - m^{(c,\ell_x)} | + c_{sm}$ in case of redundancy undershooting, with ℓ_x the eligible round with the smallest degree of consent.

6. Effectiveness Analysis

Given the availability of spare system resources, our redundancy dimensioning model is indubitably capable of scaling up the employed degree of redundancy, either in response to a failure of the scheme, or as a precautionary measure if the effectiveness of the employed redundancy is observed to deteriorate — *cf.* Sect. 4.2.2. In this section, we will analyse whether the proposed model can effectively and safely reduce the employed redundancy. In doing so, we have used the discrete-event simulation toolbox and failure injection mechanisms that were developed in [2]. Even though the toolbox provides a multitude of failure injection mechanisms, a simplistic, trend-based injection mechanism was chosen for visualisation purposes, as illustrated in the upper graph in Fig. 3. The total number of randomly chosen versions $v \in V$ for which the system will inject failures in the course of the corresponding voting round (c,ℓ) is represented by the blue marks, whereas the green marks indicate how many of those affected versions were selected for participation in $V^{(c,\ell)} \subseteq V$. Injected failures are set to materialise as EVF and RVF content failures with a 20, respectively 80% probability. The discrepant response values returned for invocations affected by RVF failures are sampled from a uniform distribution [3]. Replicas are selected using the model referred to in Sect. 4.3 which has been configured to target sustained availability.

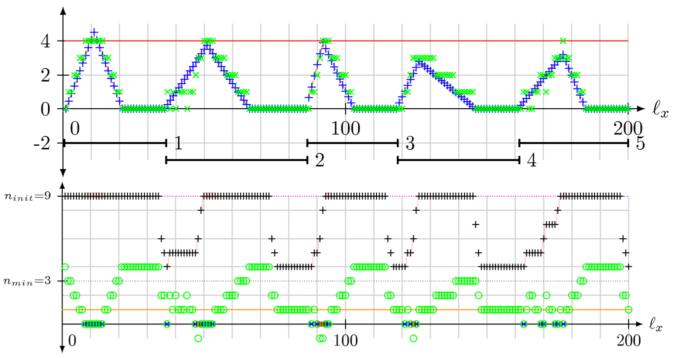

Figure 3: Number of injected failures (above); evolution of degree of redundancy and its effectiveness (below).

We now compare the effectiveness of four redundancy dimensioning strategies. Each of these strategies is assumed to be deployed within an A-NVP/MV scheme operating in an environment in which the same set V of 20 versions had been deployed and that exhibits identical failure behaviour as modelled by the blue trend line in Fig. 3. At any time, no more than four failures are assumed to affect the versions in $V^{(c,\ell)}$, a scenario that could successfully be overcome by a static redundancy configuration with $n = 9$. Apart from such classic strategy, we will evaluate the two strategies for parsimonious resource allocation introduced in Sect. 5: the former will be analysed without any safety margin applied, the latter considering two distinct values c_{sm}:

strategy	c_{sm}	r_d	$\#\ell_F$	*ttf*	$\sum cr^{(c,\ell)}$	$\sum n^{(c,\ell)}$	*ratio*
A: Classic	-	-	0	∞	724	1800	2.486
B: First	0	20	5	47	614	1358	2.211
C: Second	1	20	4	47	664	1482	2.232
D: Second	2	20	0	∞	716	1699	2.373

Table 1: Overall resource consumption and ineffectiveness. Number of voting rounds with scheme failure $\#\ell_F$.

So as to make a fair comparison, the application-specific parameters are configured with $n_{init} = n_{max} = 9$ and $n_{min} = 3$. For the sake of simplicity, it is assumed that the order in which voting rounds are initialised is the order in which they complete, *i.e.* $z = x$ for all $\ell_x = b_c(y_z)$, and that all context information is available by the time the next round is initialised **(SIO)**. This assumption improves comprehension and allows the reader to grasp the intuition behind the harvested contextual information $c_{\max}^{(c,\ell)} - m^{(c,\ell)}$ (represented by the circles in the lower graph in Fig. 3) driving the adjustment of the employed redundancy. The value $r_d = 20$ has been chosen for demonstration purposes so as to enable the model's effectiveness to be concisely captured (e.g. strategy C in Fig. 3). At the expense of an aggressive allocation strategy, we set $r_u = 1$ such that the availability of the scheme can swiftly be regained in case of redundancy undershooting.

For each voting round ℓ in the sequence $\{\ell_x\}_c$, we will consider $e^{(c,\ell)}$, that is, the number of versions $v_i \in V^{(c,\ell)}$ that are affected by some type of content failure. Recall that the green crosses in the upper graph of Fig. 3 essentially quantify $e^{(c,\ell)}$. We can then determine the contextual redundancy $cr(e^{(c,\ell)})$ and compare this value with $n^{(c,\ell)}$ to observe if the amount of redundancy $n^{(c,\ell)}$ actually used was overabundant or insufficient — *cf.* Sect. 4.2. As shown in Table 1, the relevant cumulative (contextual) redundancy during the operational life span of the scheme provides vital information regarding the effectiveness of a redundancy dimensioning model in estimating the extent to which it can economise on resource expenditure. One cannot merely judge a dependability strategy in terms of cumulative contextual redundancy; though sharing identical failure behaviour (blue trend), the actual number of injected disturbances (green marks) for any specific voting round is eventually determined by the amount of redundancy that is actually used — hence the variation in the values reported. Instead, one had better considered the ratio of total cumulative redundancy over cumulative contextual redundancy, for which the actual number of disturbances is indirectly accounted for. Though smaller, positive values are indicative of an increased level of parsimony, one should consider whether this reduction in resource allocation did not violate the dependability objective. In this respect, strategy B clearly performs worse than A, C and D despite the smallest degree of resource consumption, as the scheme experienced a failure in 5 of the 200 simulated voting rounds. This behaviour could have been anticipated, as strategy A responds to actual failures rather than proactively adjusting the redundancy upwards were the agreed safety margin violated. Observe the significant improvement with respect to the number of scheme failures resulting from a modest increase of the imposed safety margin — *cf.* strategies C and D.

In general, one may expect that a properly chosen value c_{sm} matching the variability of the environment and the ensuing disturbances aids in intercepting the trend and may lead to a reduction of scheme failures due to more efficient proactive upscaling – *cf.* **(SM)**. As can be seen from Fig. 3, a modest value $c_{sm} = 1$ enables the model to adjust the employed degree of redundancy such that it follows the trend of failures — *cf.* the blue trend in the graph above with the employed redundancy represented by the black marks in the lower graph.

7. Service-Oriented Prototype

The use of stateless web services and the confinement of any communication to take place via explicitly defined service interfaces appear to suggest that web services are an adequate technology for implementing fault containment units. A prototypical service-oriented implementation of the replica selection algorithm showed how WS-* specifications can be leveraged to sustain adaptive redundancy management, broadening the applicability of NVP schemata by increased interoperability [3]. This A-NVP/MV prototype was extended to include the proposed redundancy dimensioning model and policies defined in Sect. 4 and 5.

7. Conclusion

In this paper, we have enhanced the redundancy management facilities of our A-NVP/MV dependability strategy, which had initially appeared in [3]. The proposed redundancy dimensioning model aims to autonomously tune the employed degree of redundancy in view of encountered disturbances, and is fitted with accompanying policies intent upon increasing the scheme's cost effectiveness by parsimoniously allocating system resources. Based on the discrete-event simulation models presented in [2] with special attention paid to the impact of disturbances that manifest as transient failures in the content domain, these policies haven proven to be effective in identifying situations necessitating an adjustment of the redundancy level. It is apparent from our experimentation that the suggested solution can effectively achieve a substantial improvement in dependability, compared to traditional, static redundancy strategies. Furthermore, tuning the adopted degree of redundancy to the actually observed disturbances allows unnecessary resource expenditure to be reduced, therefore enhancing cost-effectiveness. As future work, we intend to analyse the impact of the defined parameters and investigate whether the system could tolerate graver whimsicality of its deployment environment, which under **(SM)** in itself would require an increase of the imposed safety margin c_{sm}. One possible direction is to dynamically alter the parameters r_d, r_u and r_f, and see how this may affect the effectiveness of the presented policies. Moreover, we will renounce simplification **(SIO)** of in-order voting round completions and analyse the effects of variations in the individual response times of versions.

References

[1] M. Aigner. "Combinatorial Theory", Springer (1997).

[2] J. Buys *et al.* "A Discrete-Event Model for Failure Injection in Distributed NVP", technical report (2012).

[3] J. Buys *et al.* "Towards Context-Aware Adaptive Fault Tolerance in SOA Applications", Proceedings of the 5th ACM International Conference on Distributed Event-Based Systems, pp. 63–74 (2011).

[4] F. Cristian. "Understanding Fault-Tolerant Distributed Systems", Communications of the ACM, 34(2), pp. 56-78 (1991).

[5] B. W. Johnson. "Design and analysis of fault tolerant digital systems", Addison-Wesley Publishing (1989).

[6] V. De Florio. "Application-layer Fault-tolerance", IGI Global (2009).

[7] V. De Florio. "Software Assumptions Failure Tolerance: Role, Strategies and Visions", Architecting Dependable Systems, LNCS 6420, pp. 249–272 (2011).