# Improving Network Services' Resilience using Independent Configuration Replication

Miguel Lopes
Informatics Department
University of Minho
Braga, Portugal
miguellopes@di.uminho.pt

Antonio Costa
Informatics Department
University of Minho
Braga, Portugal
costa@di.uminho.pt

Bruno Dias
Informatics Department
University of Minho
Braga, Portugal
bruno.dias@di.uminho.pt

***Abstract*—This paper proposes that the MiNSC's distributed architecture could be used to overcome the limitations of management configurations translations [1]. It uses a configuration independent approach to improve the managed service's resilience, which is based on the dynamic replication of a service node's configuration. This methodology was tested for configuration management of the DNS service and comparisons were made to methods based on traditional static service configuration redeployment instead. It was found that even though a longer period of time is necessary for completion, the replication of the node's dynamic configurations is possible.**

## I. Introduction

A well identified group of limitations was already summarized for integrated network management solution based on configuration translations [1] and includes:

- When performing syntactic translations, the semantics of management data on both source and destination is not considered. This content loss may result in management data inconsistencies or collisions [2], [3];
- Due to the lack of the semantic content of contemporary management data models the execution of semantic translation is a complex task and depends on the administrator's skills on the mapping process [2], [3]. This task is highly complex for large scale heterogeneous management environments;
- Translation mechanisms are prone to errors, i.e., eventual changes in the managed elements may require manual adaptation of the translation mechanism which may result in the insertion of errors.

As a result of the complexity of developing management translations, real integrated network management solutions are uncommon to find, most of them being implemented at a proprietary level with low level functionalities and reduced flexibility. MiNSC's management architecture [1], [4], [5] has important advantages for integration of network services management: it overcomes the use of management translations by unifying the management of heterogeneous network service implementations; promotes a technological independence between management applications and managed elements; simplifies management by using a higher level of configuration management abstraction; it realizes intermediary functionalities to improve the managed service resilience and scalability. This paper brings forth a complementary method to the one published in [1] which was targeted at the improvement of the network service's resilience. The present method uses MiNSC's distributed architecture as well as the execution of periodic and automatic independent node configuration replications which can be used for a seamless transfer of the service execution (including the node's static and dynamic configurations). This process was tested for the Domain Name Service (DNS) and it was concluded that it works flawless although it takes a longer period of time to be completed when compared to the former mechanism. The remaining paper is organized into five sections: in the next section some of the most relevant network management research projects are presented; MiNSC's framework principles and motivations are described in section three; in section four the DNS service resilience method is described and result experiments are presented; section five presents the conclusions.

## II. Related Work

There are several works that focus on the integrated network management. The FOCALE [6] project implements the autonomic control loop based on ontologies. It augments the managed resources' heterogeneous data models by mapping their facts into a common vocabulary. This enables the development of cognitive equivalences between them. This unified manner of dealing with network diversity includes a mid-level configuration conversion tool which will apply and retrieve resource-specific management data. The Model-Based Translation Layer (MBTL) is responsible for the task of semantically convert independent configurations into lower-level resources-specific configuration and commands. However, the FOCALE project does not define any particular implementation architecture or strategy to tackle this complicated mapping task. Besides, the configurations semantic conversion is not trivial and cannot be fully automated requiring the administrator intervention to create and validate the mappings [2], [3].

Considering FOCALE's architecture, one important limitation was referred in [7]: the MBTL is a potential management bottleneck because it must *speak* to all network resources' management interfaces. To solve these and other limitations, a modified version of FOCALE's autonomic element architecture was proposed, which included an agent-based archi-

978-3-901882-50-0 

tecture, resulting in a simplified MBTL operation which will then enforce vendor-neutral management data to the managed resources. This could overcome the MTBL's inefficient language translation and indeed it seems the modified architecture simplifies the MBTL operation, however some issues remain open. The translations associated to a conversion mechanism are still present, even though they are being performed in a distributed fashion. This is an obvious performance enhancement but the implementation-specific translation components must be maintained with no guarantee that the translation is correct.

WBEM [8] is another important integrated network management framework based on the implementation of standard-based management information models. WBEM acts as a middleware that enables a technological separation between the managed elements heterogeneity and the management applications. The problem of elements heterogeneity is taken care on *Providers* that syntactically translate from WBEM Common Information Model (CIM) to the managed elements data. Such as any other translation mechanism, limitations arise because *Providers* must be created and maintained.

One important configuration management tool is CFengine [9]. This tool uses a declarative language to describe low-level management policies expressed as *promises*. These *promises* are sent to agents that extract/compute the correct intent and ensure its automatic enforcement (such as package installation verifications, configuration file generation, file protection and consistency checking). Given the limitations found in current configuration management frameworks that include a lack of automation support, a dependency on administrator's manual work and an absence of support for management heterogeneity lead to the creation of a new configuration management proposal described in the following sections.

## III. Mid-Level Network Services Configuration Management

### A. Architectural Considerations

MiNSC's framework elements are organized as described in Fig. 1. Following a bottom-up approach, the *Network Service Node Management* layer is composed by the Active Service Nodes (ASNs) and Candidate Service Nodes (CSNs). The ASN nodes perform the productive operations of the network service to be managed at a server or device level, while the CSNs are not actively performing a service but may (are prepared to) do so in the future by automatically and/or periodically replicating the ASN's independent configurations. The Active Configuration Servers (ACSs) and Candidate Configuration Servers (CCSs) compose the *Service Management* layer. The ACS nodes manage the lower layer nodes (ASN and CSN), while CCS nodes improve framework resilience and scalability by maintaining up-to-date replications of ACS node configurations. Since the management nodes' classification changes along time, the Configuration Pointing Servers (CPS) are used to maintain an updated reference to the services being managed by the available ACS servers.

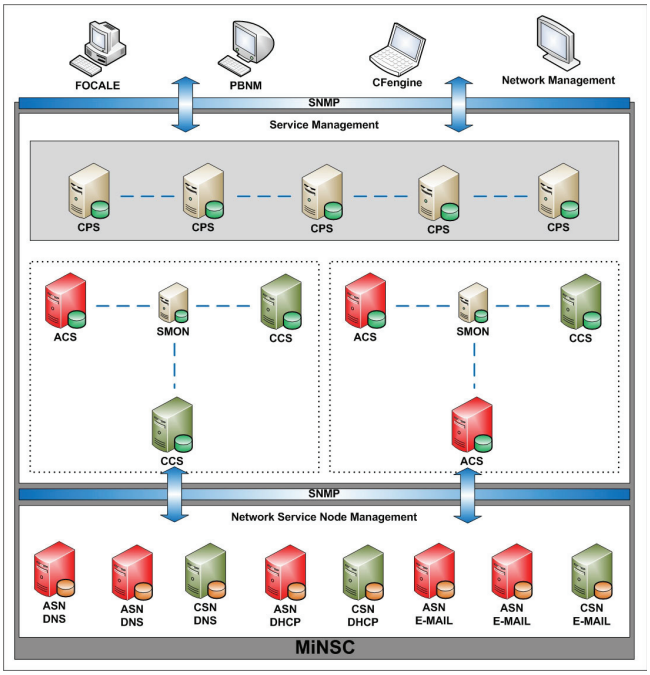


Fig. 1. MiNSC's network service management architecture

### B. Managed Service Resilience Improvement Methods

With MiNSC's two layer distributed architecture associated to the management independence provided by the *Network Service Node Management* layer and the service behavior included by the meta-configurations, defined by the *Service Management* layer, two methods for the improvement of the managed service resilience are possible: one method is based on the repetition of service deployment; the other uses active and candidate nodes to automatically and periodically replicate the service instance independent configurations.

*1) Network Service Configuration Redeployment Method:* In this method when a failure is detected in a service node (ASN), the configuration server (ACS) un-deploys the previously deployed configurations (for all network service nodes) redeploying new service configurations excluding the faulty node. This process is performed by using the managed service meta-configurations to calculate new service node configurations [1].

*2) Network Service Configuration Replication Method:* MiNSC also supports a complementary method which consists in using candidate service nodes (CSN) to automatically and periodically replicate the active (ASN) nodes' configurations. This method ensures that in case of ASN failure, its independent configurations enable the continuation of the service execution (at the CSN) with minimum data loss. This migration doesn't requires all the network service node's configuration calculation but, depending on the managed service, configuration dependencies must be ensured to guarantee the configuration integrity. This second method is further elaborated and result experiments are described in the following sections.

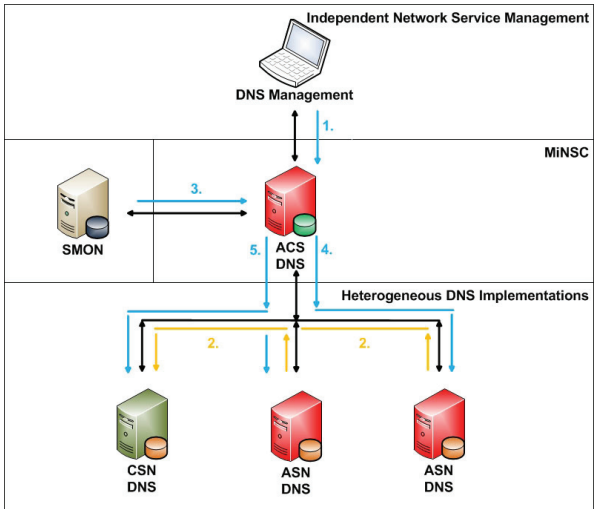


Fig. 2. DNS node configuration replication

Even though both methods improve the managed service resilience to node failures, they are performed using different processes and their results are also different. While in the first method only the node's static configurations are migrated (the configuration that directly result from the service meta-configurations), both the node's static and dynamic configurations (the configuration that result from the node's operations) are migrated in the second method. This means that when aiming towards a seamless service execution migration, both static and dynamic configurations must be included to avoid data losses and service disruption. The execution of one of these methods is of administrative decision, depending on the managed service (namely the importance of it's dynamic configurations) and the level of service disruption admitted. The realization of replication method requires the existence of redundant nodes which is not always possible. On the other hand, the redeployment method reduces the managed service physical support (excluding faulty nodes) which, has important consequences to the managed service scalability.

## IV. Improving DNS Resilience using Node Configuration Replication

With the objective of improving DNS service resilience to server failures, the test bed depicted in Fig. 2 was created. The created test bed was composed by three heterogeneous DNS implementations (DNS Bind9 for Linux and MS Windows and one Posadis for MS Windows), managed using the DNS *Node Management* information model [5] and one ACS node containing a MIB for the service meta-configuration definition (based on the DNS *Service Management* information model [10]). The details for each step include:

1) One DNS *Zone* is deployed over three DNS nodes using two different resilience levels: the first possessed a *high* resilience level where one DNS node was classified as ASN (primary name server) and two as CSN; next, with a *medium* resilience level two DNS nodes were classified as ASN (one primary and one secondary name server) and one as CSN. One must keep in mind that in order for this method to be performed, at least one CSN node must be deployed;
2) The CSN nodes are configured to (automatically) and periodically replicate the ASN node's configurations, which includes the node's static and dynamic configurations. During the replication process, the configurations are adapted to include the new node's details and promote a quicker service execution migration;
3) The configuration server (ACS) is notified by the SMON to execute a DNS node migration due to a faulty node (identifying the node);
4) The ACS server deactivates the service execution on the faulty DNS node;
5) The ACS server activates the faulty DNS node's replicated configurations on a CSN;
6) The ACS server updates the unchanged DNS node's configuration due to the configuration dependencies in order to maintain the configuration integrity.

**Algorithm 1** DNS Node Migration(source)

$zones \Leftarrow RetrieveConfiguration(source)$
**for** $zone \in zones$ **do**
  $asn \Leftarrow GetRegisteredActiveNodes(zone)$
  $csn \Leftarrow GetRegisteredCandidateNodes(zone)$
  **if** $csn > 0$ **then**
    $asn \Leftarrow UndeployZoneConfiguration(source, zone)$
    $DeploymentZoneConfiguration(asn, csn, zone)$
    $UpdateZoneConfigurationDependency(asn, zone)$
    $DeploymentRegistration(zone, asn, csn)$
  **end if**
**end for**

The DNS node configuration migration algorithm implemented by the ACS node is depicted in Algorithm 1 and can be summarized as follows:

1) The SNOM sends the identification of the faulty DNS node to be migrated (referred as *source*) to the ACS node. This activates the execution of the DNS Node Migration algorithm;
2) The ACS node retrieves the configured DNS *Zones* (called *zones*) from the source;
3) For each DNS *zone* included in the *zones*, the list of node classification for the *zone* deployment is retrieved (i.e. the *asn* and *csn* classification) to find where the *zone* can be migrated and which configuration dependencies exist. The ACS node maintains the DNS Nodes' classification for each deployed DNS *Zone*;
4) If each DNS *zone* included in the *zones* has candidate nodes (*csn*) then it is deactivated (deleted) on the DNS *source* node configuration. The same DNS *zone* is activated in one of its candidate nodes. When this is done, the *zone*'s configuration dependencies, on the remaining *asn*, are updated to include the new DNS node;

TABLE I
LOSSLESS DNS NODE CONFIGURATION REPLICATION PROCEDURE

| Deployment | Fault | Replication (seconds) | | | | |
|---|---|---|---|---|---|---|
| (seconds) | | Act. 1 | Act. 2 | Act. 3 | Act. 4 | Duration |
| Medium Resilience Level | | | | | | |
| 2.827 | P | 2.028 | 1.502 | 2.663 | 1.210 | 7.407 |
| 2.657 | S | 1.733 | 1.533 | 2.628 | 1.494 | 7.392 |
| High Resilience Level | | | | | | |
| 1.660 | P | 2.009 | 1.438 | 2.979 | 0 | 6.431 |

5) The *zone*'s new nodes' classification is registered on the ACS node.

Using the test bed depicted in Fig. 2, the DNS node migration method and algorithm were experimented. The results obtained are depicted in Table I. The activities identified in the table are the following:

- Act. 1: The CSN node's configuration replication procedure (defined when the service is deployed) is canceled (setting the status of the MIB replication table) and the service execution is deactivated on the faulty node (setting the status of the zone's configuration MIB table);
- Act. 2: The service execution is initiated on the CSN node using the faulty node's replicated configurations;
- Act. 3: New content for the CSN node's MIB replication table is defined. For the node becoming ASN the replication table is erased, for the remaining CSN nodes, new replication procedures are defined to include the new ASN node;
- Act. 4: To maintain the service configuration's integrity, the dependent ASN node's configurations dependencies are updated to include the new ASN node.

When the DNS *Zone* is deployed using a *medium* resilience level, two DNS nodes were classified as ASN (one Primary (P) and one Secondary (S) name servers) and the other was classified as CSN. When the SMON identifies the *P* name server as faulty, the process took an average of 7.4s to be completed. This roughly included 2.0s to realize Act. 1, 1.5s to perform Act. 2, 2.7s for Act. 3 and 1.2s to perform the Act. 4. When the SMON identifies the *S* name server as faulty, the process took an average 7.4s to be completed. This included 1.7s to perform Act. 1, 1.5s to perform Act. 2, 2.6s for Act. 3 and 1.5s to execute Act. 4.

When the DNS *Zone* is deployed using a *high* resilience level, one DNS node was classified as ASN (defined as Primary (P) name server) while the remaining two nodes were classified as CSN. When the SMON identifies the *P* name server as faulty, the process took an average of 6.4s to be completed. This roughly included 2.0s to perform Act. 1, 1.4s to perform Act. 2, 3s for Act. 3 and 0s for Act. 4 (since there are no configuration dependencies). Based on this experiment and its complementary work published in [1], the following conclusions may be observed:

- The redeployment of service node's configurations can be used to replace a faulty node even when no redundant nodes are defined. As demonstrated in [1] the redeployment takes on average 6.8s when the DNS service is deployed with *low* resilience level, 5.2s when deployed with *medium* resilience level and 2.5s when deployed with *high* resilience level;
- The configuration replication method takes more time to be completed than the configuration redeployment, although differences seem irrelevant (for the proposed test bed for a *medium* resilience level a difference of 2.2s and 3.9s for a *high* resilience level). The longer time taken by the replication method is mainly due to the following reasons: the replication tables (in MIBs) must be managed to define new replication procedures when changing active and candidate service nodes; configuration dependencies must be taken care when adding or removing service nodes. In the algorithm presented, most of the tasks are performed sequentially, in future works parallel execution will be performed to improve results.

## V. CONCLUSION

In this paper, MiNSC's resilience improvement method for DNS service was described and experimented, and lead to the conclusion that when performed using replicated configurations, the service execution migration does not incur into configuration data losses. However, it requires a small amount longer of time to be completed when compared to the alternative which is based on the service configuration redeployment.

## REFERENCES


[1] M. Lopes, A. Costa, and Bruno, "Improving network services resilience through automatic service node configuration generation," *in IEEE/IFIP Network Operations and Management Symposium, NOMS 2012, 16-20 April 2012 in Maui, Hawaii.*, 2012.

[2] J. E. L. D. Vergara, V. A. Villagr, and J. Berrocal, "Semantic management: advantages of using an ontology-based management information meta-model," in *Proceedings of the HP Openview University Association Ninth Plenary Workshop (HP-OVUA'2002), distributed videoconference*, 2002, pp. 11–13.

[3] J. Lpez de Vergara, A. Guerrero, V. Villagra, and J. Berrocal, "Ontology-based network management: Study cases and lessons learned," *Journal of Network and Systems Management*, vol. 17, pp. 234–254, 2009-09-01. [Online]. Available: http://dx.doi.org/10.1007/s10922-009-9129-1

[4] M. Lopes, A. Costa, and B. Dias, "Automated network services configuration management," *Integrated Network Management-Workshops, 2009. IM '09. IFIP/IEEE International Symposium on*, 2009.

[5] M. Lopes, A. Costa, and Bruno, "Towards automatic and independent internet services configuration," *6th International Conference on Network and Service Management (CNSM), Niagara Falls, Canada, October 25-29*, 2010.

[6] J. C. Strassner, N. Agoulmine, and E. Lehtihet, "Focale - a novel autonomic networking architecture," *ITSSA Journal 3(1)*, pp. 64–79, 2007.

[7] C. Hong, Z. Wenan, and L. Lu, "An approach of agent-based architecture for autonomic network management," *Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on*, 2009.

[8] "Web-based enterprise management," *Distributed Management Task Force*, http://www.dmtf.org/standards/wbem.

[9] M. Burgess, *Cfengine 3 Concept Guide*, 2008.

[10] B. Dias, M. Lopes, and A. Costa. (2011) Mid-level network services configuration (minsc). [Online]. Available: http://www.facebook.com/profile.php?id=100002078211438