                   Autonomic Functions Coordination
                draft-ciavaglia-anima-coordination-01.txt

Abstract

   This document describes a management solution capable of avoiding
   conflicts between autonomic functions. The objective of such a
   solution is to avoid network instabilities, by insuring that the
   autonomic functions pursuing different goals will cooperate instead
   of antagonize each other. This document provides both requirements
   and specifications for such a solution.

   Disclaimer: the version -01 of the draft has been issued to
   reactivate the document in order to allow discussion within the ANIMA
   WG about the coordination of autonomic functions.

Requirements Language

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in RFC 2119 [RFC2119].

Table of Contents

1. Introduction

   The document Autonomic Networking: Definitions and Design Goals
   [RFC7575] explains the fundamental concepts behind Autonomic
   Networking, and defines the relevant terms in this space. The
   central concepts are Autonomic Nodes and Autonomic Functions.

An Autonomic Function is characterized by its implementing a closed
control-loop, which we can summarize as successively:

1. Gathers metrics monitored by network equipments (that could be
    Autonomic Nodes, but not limited to)

2. Determines/computes new actions out of these inputs plus possibly
    some of the additional elements: e.g. contextual inputs, provided
    intents and gathered experience,

3. Set the computed parameters values (from the previous actions)
    inside the appropriate network equipments,

4. These new parameters values influence the network behavior, such
    that the metrics gathered by the autonomic function will evolve,

(Section 7.5 of [I-D.behringer-anima-reference-model] details more
the control loops).

The Autonomic Functions are normally designed to stabilize
(converge), at least when the network conditions are themselves
stable. However, conflicting interactions among Autonomic Functions
can create instabilities even when the network conditions have not
varied.

The document A Reference Model for Autonomic Networking
[I-D.behringer-anima-reference-model] describes the reference model
of autonomic networks, by describing the architecture and enumerating
fundamental blocks (either infrastructure pieces or enabling
functionalities). One of these functionalities pertains to the
concomitant execution of multiple autonomic functions in a safe way
(i.e. avoiding conflicts between these different autonomic loops).
Section 8 of [I-D.behringer-anima-reference-model] (Coordination
between Autonomic Functions) provides a brief introduction to this
functionality.

This document tackles this topic by successively:

1. Explaining why such a functionality is needed,

2. Detailing which objectives such a functionality should reach,

3. Sketching a simple behavior of this function,

4. Providing requirements on autonomic functions (a tentative list
    in this document version),

5. Providing some specifications items (in this preliminary version,
    while future versions would provide specifications),

2. Problem Statement

The need to coordinate the joint behavior of autonomic functions
arises from the need to cope with conflicting situations and to
provide the operator with the ability to steer autonomic network
performance to a given (intended) operational point.

Several interaction types exist among autonomic functions such as
cooperation, dependency, or conflict (and possibly others [TBD]).

Cooperation happens when an autonomic function can improve the
behavior or performance of another autonomic function, such as a
traffic forecasting function used by a traffic allocation function.

Dependency happens when an autonomic function cannot work without
another one being present or accessible in the autonomic network.

Conflicts among autonomic functions emerges from direct and indirect
interactions. A metric value conflict is a conflict where one metric
is influenced by parameters of different autonomic functions. A
parameter value conflict is a conflict where one parameter is
modified by different autonomic functions. A simple example of
conflicting interaction between autonomic functions is the
oscillations caused by an energy-saving function (which switches-off
interfaces to reduce power consumption) and a load-balancing function
(which switches-on interfaces to reduce link load).

Solving the coordination problem beyond one-by-one cases can rapidly
become intractable if one considers networks composed of tens,
hundreds or thousands of simultaneously interacting functions.
Specifying a common functional block on coordination is a first step
to address the problem in a systemic way.

3. Guiding principles

A coordination function appears as an essential component of the
ANIMA reference model in order to achieve better control on the
performance, stability and convergence of autonomic networks.

As guiding principles, the ANIMA coordination function should:

o Maximize the autonomic network utility, i.e. mitigate the
   (observed or inferred) detrimental effects of conflicting
   autonomic functions (Efficiency property).

o Balance the autonomic network goal(s) and autonomic functions
   individual goal(s) (Congruence and Coherence properties).

o Inform the autonomic network operator (being a human or a machine)
   with processed and aggregated "call(s) for governance" in case the
   goals are incompatible and no satisfactory solution can be found
   (i.e. compliant with the intent).

o Deviate the least possible autonomic functions from their design
   objective(s) and individual goal(s) (Liberality property).

o Impose minimal additional requirements on the external
   specifications of autonomic functions, such as the format and
   content of the autonomic function descriptor(s)/capabilities
   (Economy property).

o Not impose any requirement on the internal specifications of
   autonomic functions (Independence property).

o Support multiple coordination mechanism types (Plurality
   property).

o Enable coordination mechanisms to be plugged in at deploy- and
   run-time (Modularity property).

o Determine the most suitable coordination mechanism(s) to apply
   according to contexts (e.g. change in autonomic functions, change
   in intents/goals, change in coordination mechanisms available)
   (Dynamicity property).

o Develop a long-term vision of the autonomic functions interactions
   and devise the most suitable plan to address the conflicting
   cases, based on available coordination mechanisms and mission(s)
   set by the intent (Adaptivity property).

o Be able to fully or partially suspend/stop one or multiple
   autonomic functions, temporarily or an undetermined amount of time
   until the situation evolves (Authority property).

o Be able to operate equally well in a distributed or centralized
   manner (Distributivity property).

o Be able to cope with several thousands of simultaneous
   interactions (Performance and scalability property).

4. Initial sketch of a Coordination Function

   For the sake of the following sections, this section is providing a
   rough description of the functioning of a coordination function, and
   how it organizes itself along the network time.

4.1. Preliminary assumptions

   Autonomic functions do exist in different states corresponding to
   different steps in their life-cycle. The description of some of
   these steps is better understood by referring to the
   High level view of an Autonomic Network which is depicted in Figure 1
   of [I-D.behringer-anima-reference-model], which Figure is copied
   below:

```
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
:               :        Autonomic Function 1      :              :
: ASA 1         :      ASA 1      :      ASA 1      :       ASA 1  :
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
               :                :               :
               :  +- - - - - - - - - - - - - - +  :
               :  :   Autonomic Function 2     :  :
               :  : ASA 2      :       ASA 2   :  :
               :  +- - - - - - - - - - - - - - +  :
               :                :               :
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
:             Autonomic Networking Infrastructure              :
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
+--------+   :     +--------+   :     +--------+   :        +--------+
| Node 1 |--------| Node 2 |--------| Node 3 |----...-----| Node n |
+--------+   :     +--------+   :     +--------+   :        +--------+
```

             Figure 1: High level view of an Autonomic Network

   Undeployed -   In this state, the Autonomic Function is a mere piece
      of software, which may not even be copied on any node, but which
      may well be the code of the Autonomic Service Agents (ASA)
      corresponding to this Autonomic Function.

   Instantiated/Deployed -   In this state the Autonomic Function is
      deployed, which means the ASA are available in the Nodes and
      gathered together into an Autonomic Function. In this state the
      autonomic function is bind to a scope which is the part of the
      network on which the autonomic function is meant to perform its
      duties. As a first approximation, the scope matches the Nodes

which receive instructions from one of the ASA gathered in the
Autonomic Function.

Running -   In this state, the autonomic function is deployed and is
    executing its closed control loop, hence acting on network, by
    modifying Nodes parameters.

The above list of states is not meant to be exhaustive, and would be
better expanded in a document dedicated to Autonomic Functions,
nevertheless the distinctions between the three above states are
unavoidable.

4.2. Algorithms for coordination

This sub-section does not intend to specify algorithms capable of
achieving coordination between autonomic functions, but means to
illustrate different ways of avoiding conflicts, we can briefly list
the following families of algorithms:

Random token -   This algorithm is insuring that each autonomic
    function is executing its control-loop the one after the other,
    the sequence is following a random pattern.

Time separation -   This algorithm is insuring that each autonomic
    function is executing its control-loop at different rates, e.g.
    for 2 functions: one is running fast enough to have time to
    converge in between two iterations of the slower one (this
    algorithm requires proper settings with regards of the autonomic
    functions to coordinate).

Efficiency bids -   In this algorithm, each autonomic function
    predicts which improvement its executing of its control-loop would
    bring, hence the coordination algorithms, picks the autonomic
    function promising the "best" improvement, and grants it the right
    to execute.

4.3. Behavior of the coordination function

This function is expected to steer the network towards a better
"operating" point, by avoiding/mitigating detrimental interactions
between Autonomic Functions.

The first step of such a process is the identification of these
interactions and their classification in order to determine which
ones have to be handled (at least the problematic ones i.e.
conflicting ones).

The second step is the gathering of the identified interactions in groups that can be handled together while insuring the proper behavior of the network. This step intends to avoid handling all the interactions in one raw, but possibly to split the whole problem in smaller pieces, easier to handle.

The third step is the instantiation of coordination mechanisms well suited to handle each groups of interactions previously identified. Hence these coordination mechanisms would control the autonomic functions in order to insure a network behavior matching the intents of the network operator.

4.3.1. Times of the identification of interactions between AF

As the coordination function handles autonomic functions, its working is related to the different states of autonomic functions, namely, build-time, deploy-time and run-time. Hence the coordination function also present a life-cycle consisting in these 3 different states , in which the coordination function behaves according to the following descriptions:

At build-time, a common description of the autonomic function attributes (metrics, parameters, actions, capabilities...) allows to construct a "static interaction map" from the a-priori knowledge that can be derived/inferred from the functions attributes relationship. The static interaction map can be used as a first element by the operator (or mechanism) to (pre-)define policies and priorities as coordination strategies to manage the a-priori conflicts identified.

At deploy-time, autonomic functions are deployed on the network (i.e. installed, configured, instantiated...) but are not yet active/acting on the network. At this stage, for each instance of the autonomic functions and on a per resource basis, an inventory of the metrics monitored, of the actions performed and their relationships can be realized, resulting in a "dynamic interaction map". The dynamic interaction map provides the basis to identify conflicts that will happen at run-time, categorize them and plan for the appropriate coordination strategies/mechanisms.

At run-time, conflicts happens and arbitration is driven by the coordination strategies and available mechanisms. This is also the stage where new dependencies can be observed and inferred, ultimately resulting in update of the dynamic interaction map and possible adaptation of the coordination strategies and mechanisms.

4.3.2. Times of the coordination of AF

   TBC

4.4. Conclusions

   Some of the previous elements impact directly the coordination
   function, some other imply capacities of external elements such as
   Autonomic Functions and the Autonomic Control Plane. This conclusion
   is briefly categorizing and summarizing those:

   Requirements onto the AF -

        a descriptor of metrics and parameters/actions: a generic way
        of describing the inputs and outputs of the closed control
        loop, in order to identify the interactions.

        a life-cycle: to match the process of the coordination (shortly
        stated, interaction identification and then conflict solving).

        a common command interface of the autonomic functions: for the
        coordination to control the pace at which an autonomic function
        executes its control loop.

   Requirements onto the ACP -

        a common representation of information and knowledge: a
        function used to build the interactions maps.

   Requirements onto the Coordination Function -

        interaction identification: a function in charge of identifying
        interactions

        interaction grouping: a function coping with grouping the
        previously identified interactions, in bundles that can be
        managed independently (for scalability concerns)

        supporting various coordination mechanisms: to have the freedom
        of picking the most appropriate one.

        interaction solving: a function capable of handling an
        independent bundle of interactions by controling the implied
        autonomic functions according to the picked algorithms.

5. External Requirements

   At this stage of the document, this section is merely providing a
   structure of its content.

   In order to achieve the aforementioned goals (detailed in section
   Section 3) a Coordination Functional Block should bring the following
   features:

      a common description of autonomic functions attributes and its
      life-cycle.

      a common command interface between the coordination "agent" and
      the autonomic functions.

      a common representation of information and knowledge (cf.
      interaction maps).

   Guidelines, recommendations or BCPs can also be provided for the
   aspects pertaining to the coordination strategies and mechanisms.

   The coordination function requires a certain set of elements to work
   properly such as the autonomic function descriptor and the
   interaction map(s).

5.1. Autonomic Function Descriptor (AFD)

   The Autonomic Function Descriptor (AFD) should contain the following
   elements:

      actions, metrics, parameters, controlled resources.

5.2. control/command interface of AF

   The Autonomic Function could be guided in its executing of its
   control-loop by the coordination mechanism. The guidance could range
   from preventing the executing of the control loop, to letting run on
   its own. In the middle of the range, coordination mechanism could
   restrain the actions, halt the control-loop at a given state of the
   execution (before enforcement).

   This section can be expanded in conjunction with Section 7.5 of
   [I-D.behringer-anima-reference-model] details more the control loops.

5.3. Interaction/Information Maps

The Autonomic Control Plane(ACP) should be able to provide a view of
the interactions between metrics in order to build the interaction
maps. This functionality is needed to identify that metrics are
coupled. E.g. the capacity of a link and its load ratio are
intimately coupled, and to identify interactions between autonomic
function, having this knowledge may prove instrumental.

6. Specifications

The coordination function can be decomposed in the following sub-
functions:

interaction identification: in charge of identifying interactions

interaction grouping: coping with assigning the interactions to
instances of cooperation mechanisms

interaction solving: coping with various algorithms

TBC.

7. Acknowledgements

This draft was written using the xml2rfc project.

This draft content builds upon work achieved during UniverSelf FP7 EU
project.

The authors thank Dimitri Papadimitriou for his valuable comments.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

TBC

10. References

10.1. Normative References

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119,
            DOI 10.17487/RFC2119, March 1997,
            <http://www.rfc-editor.org/info/rfc2119>.

10.2. Informative References

   [I-D.behringer-anima-reference-model]
             Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L.,
             Liu, B., Jeff, J., and J. Strassner, "A Reference Model
             for Autonomic Networking", draft-behringer-anima-
             reference-model-04 (work in progress), October 2015.

   [RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A.,
             Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic
             Networking: Definitions and Design Goals", RFC 7575,
             DOI 10.17487/RFC7575, June 2015,
             <http://www.rfc-editor.org/info/rfc7575>.

Authors' Addresses

   Laurent Ciavaglia
   Alcatel-Lucent
   Villarceaux
   Nozay 91460
   FR

   Email: laurent.ciavaglia@alcatel-lucent.com

   Peloso Pierre
   Alcatel-Lucent
   Villarceaux
   Nozay 91460
   FR

   Email: pierre.peloso@alcatel-lucent.com