Networking 2013 1569702439

Adaptive Failure Detection Timers for IGP Networks

Bruno Vidalenc
Alcatel-Lucent Bell Labs, Nozay, France
Bruno.Vidalenc@alcatel-lucent.com

Ludovic Noirie
Alcatel-Lucent Bell Labs, Nozay, France
Ludovic.Noirie@alcatel-lucent.com

Samir Ghamri-Doudane
Alcatel-Lucent Bell Labs, Nozay, France
Samir.Ghamri-Doudane@alcatel-lucent.com

Eric Renault
Telecom SudParis, Evry, France
Eric.Renault@it-sudparis.eu

***Abstract*—Guaranteeing a high availability of communication services to customers is a day-to-day challenge for Carrier networks. In the context of IGP routing, when a failure occurs, a re-convergence process is initiated in order to re-establish a consistent view of the network. During this process, the latency of the failure detection, which is realized by the Hello protocol, is responsible for an important unavailability. Indeed, quick failure detection would require the use of fast Hello exchange which, in turn, would cause false detection and instability. However, there are often forewarning signs that a network device is about to stop working properly. Based on an embedded and real-time risk-level assessment, one can adapt in a real-time manner the Hello message frequency of sick nodes and thus reduce unavailability while maintaining the routing stability. This papers details and evaluates a mechanism for adaptive failure detection timers in IGP networks. The impacts in terms of availability and quantity of Hello messages have been estimated based on an analytical model and then simulated to measure the benefits of the proposed proactive self-healing function.**

I. Introduction

IP networks now represent a large portion of networks deployed worldwide, and more and more operators are investing in full IP networks. Indeed, they have the advantage to be fast and easy to carry out, through their various autonomic mechanisms [1], to be inexpensive, and to allow a high flexibility. However, the IP protocol suffers from some defects that lead operators to associate the MPLS protocol for their critical networks. Besides traffic engineering, IP protocol lacks effective fault management which is an essential factor for the Quality of Service (QoS) provided by operators to their customers. Both topology and fault management rely on the Hello protocol which is not flawless. The Hello protocol is largely responsible for the duration of service interruptions caused by a breakdown that may last for several seconds. Such unavailability is not acceptable for QoS sensitive traffics. Unfortunately, the Hello protocol that detects failures by sending control messages at regular intervals allows fast detection only by increasing the frequency of sending these messages at the expense of network stability. To solve this problem, this paper proposes an Adaptive Failure Detection Timers (AFDT) mechanism that exploits real-time failure prediction to automatically adjust the Hello protocol frequency rate. Many works have been done on the failure prediction task but none of them interest in how properly exploit such information. We differ by proving a realistic way to take advantage of a failure prediction, and evaluating the expected benefit for the operators, in order to justify the interest of implementing failure prediction functions into the network equipments. The paper contribution aims to answer the following questions:

- What mechanisms can efficiently exploit failure predictions ?
- What are the gains expected by such predictions?
- What performances the failure prediction has to offer to improve network management?

After a brief description of IP restoration and its related works in Sec. II, the paper describes the self-healing mechanism in Sec. III and explains its integration with regards to the Hello protocol in Sec. IV. Section V defines the analytical model developed to measure the effect of the mechanism on the network availability and the routing stability. Afterwards, the instantiation of this model network examples and its comparison with simulation experiments are analyzed in Sec. VI. Finally, conclusion remarks of Sec. VII close the paper.

II. Context and related work

In IP networks, intra-domain routing is performed by using Interior Gateway Protocol (IGP) such as OSPF and IS-IS. With OSPF, each router in a routing area discovers and builds a complete view of the network topology. When topology changes, typically due to a failure, the convergence process is triggered. This process is composed of four main phases [2]: a failure detection with t_D as the failure detection time, the Link State Advertisement (LSA) flooding with t_F as the LSA flooding time, the Shortest Path First (SPF) computation with t_{SP} as SPF computation time, and the routing table and forwarding table update with t_U as table update time. Consequently, convergence time t_C, during which the routing topology is not consistent, is given by $t_D + t_F + t_{SP} + t_U$. As it stands,

the downtime of flows affected by a failure due to the convergence process is of the order of several seconds, which is not acceptable for premium traffics. Compared to failure detection, the other steps are almost negligible in terms of time since they are less than a hundred milliseconds, with 0.03 seconds for t_F [3] and 0.2 seconds for t_U [4]. The computation time of the shortest paths depends on the number of nodes in the network which, according to [4], is equal to $t_{SP}(N) = 2.47.10^{-6} * |N|^2 + 9.78.10^{-3}$ where N is the set of routers in the network. For clarity reason, we will omit the dependence on N by using t_{SP} in the following formulas. Although subsecond failure detection had been considered as resolved by using link layer detection or hardware implemented BFD protocol in the data plane, these mechanisms are not always available and, most of all, do not allow to detect software failures present in the control plane. Unfortunately, software failures have become so important [5] that disregarding such failure is not possible anymore. That is why the behaviour of Hello P2P failure detection protocol of both OSPF and IS-IS, where each router periodically sends keep-alive messages to all its neighbors, is crucial. For each router, the period between consecutive Hello messages sent is defined by the *Hello Interval* parameter. This parameter has a default value of 30 seconds but can drop down to 1 second. However, it is usually of the order of tens of seconds. After *Router Dead Interval* seconds, a neighbor's node that has not received these Hello messages removes the router from the topology. The *Router Dead Interval* parameter is commonly set to 3 or 4 times the value of *Hello Interval*. This ensures a fairly reliable failure detection by waiting for the non-reception of 3 or 4 Hello message before declaring it.

Although the *Hello Interval* can be as short as one second, operators prefer to use higher values, *i.e.* between 3 and 10 seconds [6]. In fact, the use of a high frequency for sending Hello messages generates false detection of failures, and hence creates oscillations in the routing process [7]–[9]. Indeed, the need to send and receive Hello messages on a few milliseconds basis creates an additional workload that is not harmless for the main controller or the control processor and may disturb other tasks performed by the processing unit. Moreover, the shorter the *Hello Interval*, the higher the probability a network congestion leads to the loss of several consecutive Hello messages.

In many works dedicated to the improvement of the convergence time [10], [11], a large part deals with failure detection. The proposed solutions consist in reducing the *Hello Interval* to sub-second values [4], [12] to speed up the convergence of IGP protocols of one order of magnitude. But this raises the problem of stability [8] that has been the subject of numerous studies advocating for the dynamic adaptation of the Hello rate to the observed network congestion [13]. Nevertheless, this solution considers that the sensitivity of the Hello frequency to the congestion is similar on each device, which is not the case in today's networks with heterogeneous routers having different vendors, qualities and ages. Another strategy proposes to automatically adjust the frequency to the number of routing flap [14], but this solution waits to observe oscillations before to take action, limiting the problem without eradicating it. Finally, the Bidirectional Forwarding Detection (BFD) protocol [15] has been defined to allow a much faster detection by using a much faster frequency of sending Hello messages. It can be hardware implemented on line cards to support sub-second rates without generating instability. However, this implies not detecting faults that occur in the controller or routing software that is problematic with regards to the importance of such failure [5].

Despite efforts by the community to define the optimal frequency for sending Hello messages, operators usually use values in the order of few seconds [6]. Indeed, these studies are based on some specific network configurations, which are unrepresentative of the extent of operational networks. In practice, the *Hello Interval* is never under one second, but around three seconds [6] and up to ten seconds, depending on the characteristics of each network. Although subsecond failure detection is possible in some specific cases this does not concern all equipments, all protocols and becomes less and less frequent with the raise of the software failures [5]. That why, improving the Hello protocol is still a present concern.

III. Self-healing mechanism using risk-assessment

The reliability of network infrastructures has continuously evolved over the years. However, failures remain. The origin of failures has somewhat evolved too [5]. Fortunately, many of them are predictable thanks to the monitoring. On large scale systems, on-line failure prediction techniques like failure tracking, symptom monitoring, error reporting or undetected error auditing have already been tested [16]. Moreover, today's network managers generate thousands of warning messages that can be monitored to extract a multitude of indicators. This includes the temperature of the processing unit, the power supply voltage information available via the ACPI (Advanced Configuration and Power Interface) or the hard-drive status indicated by the SMART (Self-Monitoring, Analysis, and Reporting Technology) and all other indicators based on hardware sensors. Network based parameters like the bit-error rate or the packet-loss rate can also be integrated. However software malfunction, which become more and more important [5], are detectable by checking log entries, system errors, unreleased file locks, file descriptor leaking, data corruption, memory leaking, etc. Communications with external systems, like Intrusion Detection System (IDS) or Network Management System (NMS) are also envisioned. At last, machine learning techniques [16] using Bayesian networks, time series analysis, Support Vector Machines or Semi-Markov Processes can be used to make prediction more accurate.

Unfortunately, there is a lack of appropriate tools inside network elements to report failure detections in advance, to provide a risk-level assessment in a timely manner, and to supply relevant decisions with these information. In Generic Autonomic Network Architecture (GANA) [17], this role is devoted to the Risk assessment module to create a time window wherein the autonomic mechanism would have time to dynamically change the Hello timers. One of the propositions of the present work, in accordance with the Hello protocol, is to configure all timers for Hello messages to a slow rate most of the time and move to a sub-second rate timers for which the risk-assessment predicts an upcoming failure on the interface. This increases the responsiveness to failure when warning signs occurred while maintaining a stable network. Similarly, the impacted timers for Hello messages return to a slow value when the probability of risk of failure is low. Considering that risky periods are usually short and limited to very few interfaces, the network should remain stable while the availability for predicted failures increases. However, failure predictions can also be wrong. This results in sending extra Hello messages fortunately limited to prediction periods and only concerned interfaces. Considering the gain in terms of network availability as a whole, this inconvenience is acceptable.

Many studies concentrated on the evaluation of failure prediction. This study sets apart failure prediction performances and focuses on the proactive action part. Indeed it is essential to ensure that failure prediction permit a real QoS improvement. We answer this question by providing a pragmatical proactive resilience mechanism, evaluating its benefits and characterizing the constraints on the failure prediction task.

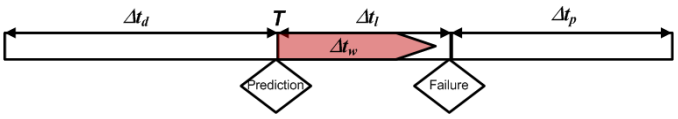


Figure 1. Time relation in online failure prediction.

The following details the characteristics of failure predictions at the different time periods involved in the online failure prediction process (see Fig. 1):

- T: the present time.
- Δt_d: the data window time in which the failure predictor keeps the data used to predict upcoming failures.
- Δt_l: the lead time, *i.e.* the minimal period between the prediction and the failure event.
- Δt_w: the warning time is the time required to set up proactive self-healing actions. It is always smaller than Δt_l.
- Δt_p: the prediction period for which the prediction remains valid. Most competitive failure prediction methods consider values up to few minutes [18]. But, in order to consider the widest set of failure prediction methods, we take a margin by setting this parameter to one hour.

We could have chosen to use a subset of existing failure prediction mechanisms but we chose to use failure prediction performance metrics in order to be more exhaustive, i.e. to give an overview of the performances of our mechanism with any failure prediction mechanism. Regarding performance, both unpredicted failures (FN) and wrong predictions (FP) characterize failure predictors. In this paper, *Precision* and *Recall* are the considered metrics: *Recall* to evaluate the ability to detect a failure using the correctly predicted failures (TP) over the total number of effective failures ratio; *Precision* takes into account false predictions using the correctly identified failures over the total number of predicted failures ratio. As advances in the failure prediction field allow a large ratio (greater than 90% for *Recall* and 80% for *Precision* [18]), in this paper, a value of 80% is considered as as the most competitive value for both metrics.

$$\text{Recall} = \frac{TP}{TP + FN} \qquad \text{Precision} = \frac{TP}{TP + FP}. \tag{1}$$

Recall, *Precision* and Δt_p are the inputs considered in Sec. V and VI to analyze the failure prediction performance which enable the adaption of failure detection timers. Then, *Recall* and *Precision* are also used to take into account the uncertainty of failure predictions.

IV. Applicability to Hello protocol

The proposed idea aims to dynamically adapt the *Hello Interval* when a predictive failure is detected. Many studies concentrate their efforts to put forward failure prediction mechanism but neglect that how exploiting this information is just as important. In consequence, we choose to repair this deficiency by proposing a pragmatic and extremely fast to activate proactive mechanism to exploit failure prediction and by evaluating the expected gains, as well as the constraints with regards to the failure prediction task.

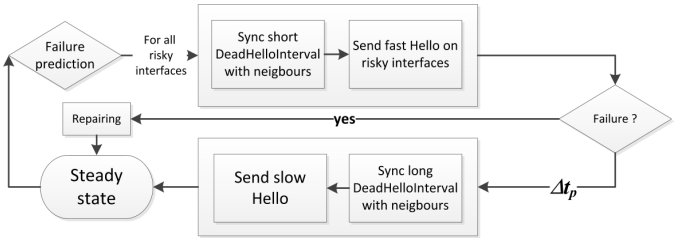


Figure 2. AFDT state diagram.

Thanks to the prediction provided by the Risk Assessment Module (RAM), the module that manages the frequency of Hello messages, located within the routers, is in charge of accelerating the Hello rate of all interfaces concerned by a failure prediction (see Fig. 2). Symmetrically, the Fig. 2 shows that AFDT module is also in

charge of restoring a less-aggressive *Hello Interval* value that is considered as extremely stable when the prediction expire.

Changing the Hello rate requires two steps. First the remote neighbour need to be aware of the new timer value thanks to a synchronization step to update its *Router Dead Interval* consequently. Then, the risky node can send fast Hello from the risky interfaces.

The solution described above requires some specific changes to the Hello protocol of OSPF or ISIS. The first required extension is to enable the dynamic change of both *Hello Interval* and *Router Dead Interval* values, which is currently not possible. These settings are currently configured to init the service and are not intended to be modified. Secondly, the configuration of these parameters must be specific to each neighbor, so that it becomes possible to assign a rapid rate to a single router or a single link. The values for *Hello Interval* and *Router Dead Interval* are currently global to an entire area.

Finally, it is also necessary to introduce the ability to specify values of *Hello Interval* and *Router Dead Interval* that are smaller than one second. BFD is already capable of maintaining individual timers and changing them dynamically without noticeable overhead. The real issue is to be capable of handling fast Hello rates. BFD can resolve this problem with an implementation in the data plane. This solution has the disadvantage of not considering the numerous errors happening in the control plane. For this reason, we prefer to modify the hello protocol even if BFD could indeed be an alternative to deploy the mechanism without standardization effort.

The proposed mechanism combines all advantages of *Hello Interval* configuration strategies: rapid detection of failures when the environment is risky and a smooth and stable behaviour when the environment is safe. With a *Hello Interval* greater than one second during normal periods, stability is ensured. Then, at a failure prediction, the Hello burst is confined to the risky network elements during the prediction period Δt_p. A router usually receives Hello messages on only one interface during a limited period without comparison with the use of a rapid rate full time across the whole network.

V. Analytical modelling

A. Notations

Let N be the set of nodes (or routers) in the network and E be the set of directed edges (or links). The network can be modelled as directed graph G=(N,E). Let F be the set of traffic flows in network G. Each traffic flow $f \in F$ is associated with its ingress node $In(f) \in N$, its destination node $Out(f) \in N$, its throughput $\mu(f)$ and the ordered list of transit routers which is the shortest path provided by the routing protocol (OSPF or IS-IS). In this study, the impact of the failure of ingress or egress nodes is not considered as there is no way to protect from or restore the traffic flows in such a case. In real networks, such a case is handled by multi-homing.

Each node $n \in N$ is mainly characterized by its Mean-Time-Between-Failure $MTBF(n)$ and the Mean-Time-To-Repair $MTTR(n) << MTBF(n)$. For the risk-awareness modelling, let $Recall(n)$ and $Precision(n)$ values be defined for each node n (Cf Sec. III). Changes during time for these parameters are not considered. For a stationary ergodic process, the probability that a node is in a failure state is given by:

$$P_{node}(n) = \frac{MTTR(n)}{MTBF(n) + MTTR(n)} << 1 \qquad (2)$$

When considering the case of identical routers, with regards to failure probability, one can omit the dependency in n in the notations for all these parameters. For each flow $f \in F$, when a transit node $n \in sp(f)$ is failing, the flow is only restored after the convergence process which duration t_C is define in Sec. II. In order to evaluate the AFDT mechanism, it has to be compared to the standard use of the Hello protocol with both fast and slow timers. While t_F and t_U are constant and t_{SP} depends on the network size, t_D depends on the *Hello Interval* noted t_{HI} and the *Router Dead Interval* noted t_{RDI}. If one considers that failure occur uniformly between two Hello messages, and that $t_{RDI} = 4 * t_{HI}$, the average detection time is :

$$t_D = (t_{RDI} - t_{HI}) + \frac{t_{HI}}{2} = (3 * t_{HI}) + \frac{t_{HI}}{2}. \qquad (3)$$

Nevertheless, all three strategies (slow timer (S), fast timer (F) and AFDT) require the definition of t_{FHI} (resp. t_{SHI}) to represent the fast *Hello Interval* (resp. the slow one) and of t_{FRDI} (resp. t_{SRDI}) that represent the short *Router Dead Interval* (resp. the long one). As a consequence, there are two convergence times, one using fast timers t_{FC} and the other one using slow timers t_{SC} such as:

$$\begin{aligned} t_{FC} &= (t_{FRDI} - t_{FHI}) + \frac{t_{FHI}}{2} + t_F + t_{SP} + t_U \\ t_{SC} &= (t_{SRDI} - t_{SHI}) + \frac{t_{SHI}}{2} + t_F + t_{SP} + t_U. \end{aligned} \qquad (4)$$

B. Unavailability computing

For the AFDT case, the conditional probability part can be split in the sum of the two disjoint probabilities about successful risk detection ($Recall$) or unsuccessful risk detection ($1 - Recall$), which finally gives:

$$\begin{aligned} U_{AFDT}(f,n) = Recall(n).t_{FC}/(MTBF(n) + MTTR(n)) \\ + (1 - Recall(n)).t_{SC}/(MTBF(n) + MTTR(n)). \end{aligned} \qquad (5)$$

Note that the formula for the fast timer case (resp. slow timer case) is given by Eq. (5) by setting $Recall(n) = 1$ (resp. $Recall(n) = 0$). The following relationship stands:

$$\begin{aligned} U_{AFDT}(f,n) =& (1 - Recall(n)).U_S(f,n) \\ &+ Recall(n).U_F(f,n). \end{aligned} \qquad (6)$$

Thus, $U_S(f,n) \geq U_{AFDT}(f,n) \geq U_F(f,n)$. Assume that router failures are independent events, with $X = AFDT$, S or F, the network unavailability for flow $f \in F$ is:

$$U_X(f) = 1 - \left(\prod_{n \in sp(f)} (1 - U_X(f,n)) \right) \approx \sum_{n \in sp(f)} U_X(f,n). \quad (7)$$

The approximation being valid as of Eq. (2), which means we neglect the case with simultaneous failures of two or more routers in the network. Then one can give a weight for each flow f, e.g., their throughput $\mu(f)$, in order to define the average network unavailability:

$$U_X(G,F) = \frac{\sum_{f \in F} \mu(f).U_X(f)}{\sum_{f \in F} \mu(f)}. \quad (8)$$

C. Quantity of Hello messages

As we said, the routing instability generated by routing flaps is the major drawback of short Hello timers. This well know issue [7]–[9], [14] is unfortunately hard to model because of the heterogeneity of equipment with multiple control packet management architectures, the various traffic engineering policies, queues management and so on, make it very difficult to quantify, precisely and in a general way, the impact of Hello timer values on routing instability, especially on the number of false failure detection. In consequence, we prefer to avoid approximation and measure the cause if its instability evaluated by the measure of the Hello message rate that each router have to process and let operators evaluate if it is compatible with their own specific infrastructure. To calculate the number of Hello messages received per second, it is necessary to average the messages received for all nodes when they are not failed. With d^- and d^+ the in and out degree, the average Hello rate by node for the standard strategies (*i.e.* S and F) is:

$$H_{IGP}(G) = \frac{\sum_{n \in N} d^-(n) - (P_{node}(n).d^+(n))}{|N| * t_{HI}} = \frac{|E|.(1 - P_{node})}{|N| * t_{HI}}. \quad (9)$$

The AFDT case needs to take into account the impact of true positive (TP) and false positive (FP). The corresponding probabilities for node n are:

$$P_{FP}(n) = P_{node}(n).Recall(n).\left(\frac{1}{Precision} - 1\right).\left(\frac{\Delta t_p}{MTTR(n)}\right),$$
$$P_{TP}(n) = Recall(n).\left(\frac{\Delta t_p/2}{MTTR(n) + MTBF(n)}\right). \quad (10)$$

The above formulas in Eq. (10) take into account that:

- at a wrong prediction (FP), the n node sends Hello messages with the fast frequency ($1/T_{FHI}$) for the prediction period Δt_p;
- when a failure is predicted in advance (TP), the node n sends Hello messages with the $1/T_{FHI}$ frequency only before the failure occurrence. Considering that the occurrence of failures is uniformly distributed over Δt_p, the average time of sending Hello messages at the fast frequency is $\Delta t_p/2$;
- the remaining time Hello messages are sent to the slow frequency $1/T_{SHI}$, except during a failure where no message is sent.

It is then possible to calculate the average of received Hello per second and per node. With a unique failure probability P_{node}, meaning that we consider identical routers, the equation can be simplified as the following:

$$H_{AFDT}(G) = \sum_{n \in N} \frac{d^+(n)(1 - (p_{node}(n) + P_{TP}(n) + P_{FP}(n))}{|N| * t_{SHI}} + \sum_{n \in N} \frac{d^+(n).(P_{TP}(n) + P_{FP}(n))}{|N| * t_{FHI}}$$
$$H_{AFDT}(G) = \frac{|E|}{|N|} * \left(\frac{1 - (P_{node} + P_{TP} + P_{FP})}{t_{SHI}} + \frac{(P_{TP} + P_{FP})}{t_{FHI}} \right). \quad (11)$$

The next section is dedicated to the AFDT mechanism evaluation with concrete examples.

VI. Case study analysis

A. Network topologies and traffic matrices

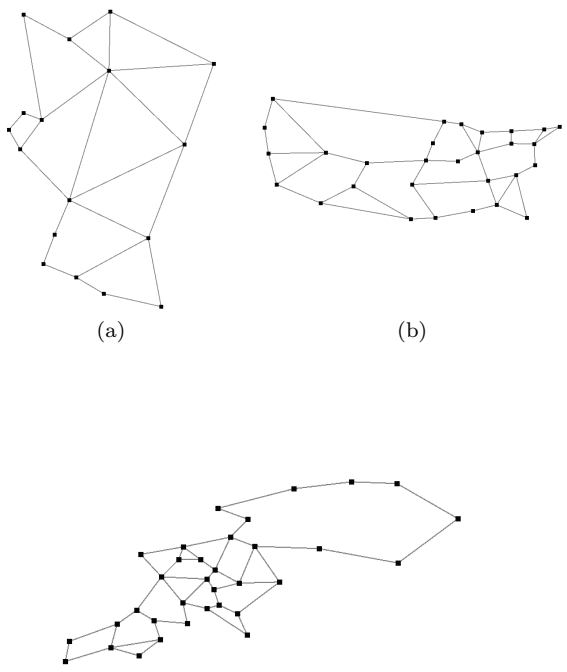
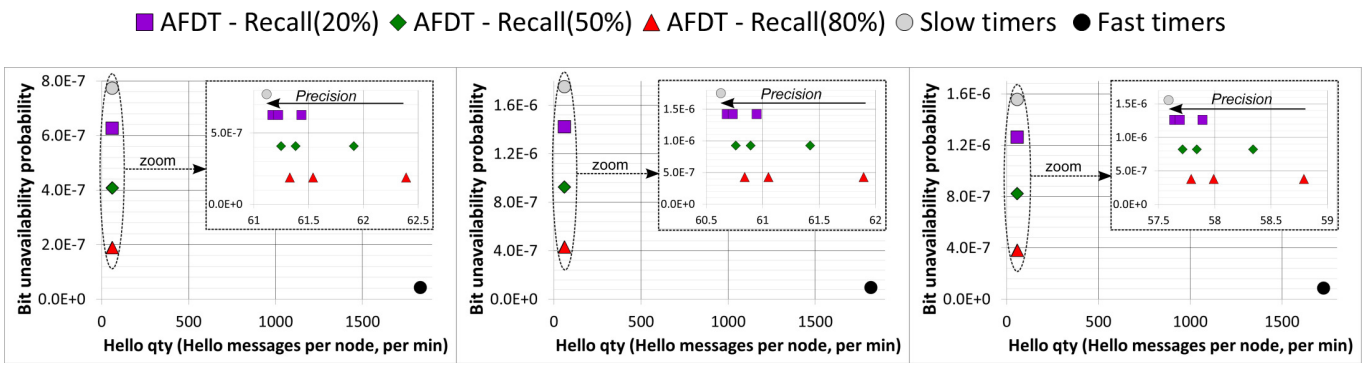


Figure 3. Network topologies.

The real-core network topologies presented in Fig. 3 have been used to illustrate the aforementioned models: a)



(a) (b) (c)

Figure 4. Hello quantity and unavailability estimation ($MTTR = 5000h$, $MTTR = 5h$, $\Delta t_p = 1h$) for the 3 networks.

a national network in Europe, b) a US network and c) a Pan-European network. Table I details the characteristics of these networks: $|N|$ represents the number of routers, $|E|$ the number of links, $\bar{d}$ the average degree and $d(G)$ the diameter of the network. Additionally, IGP metrics are based on the distance between each node (*i.e.* between cities), and the flow distribution for traffic matrices is proportionally balanced with city populations. The number of flows ($|F|$) and the sum of these flows (Traffic) can be found in Tab I.

G	$\|N\|$	$\|E\|$	$\bar{d}$	Density	$d(G)$	$\|F\|$	Traffic (Gbit/s)	#IF
a	17	26	3.06	0.19	8	242	1363	626
b	29	44	3.03	0.11	9	812	485	440
c	34	49	2.88	0.09	14	1122	1554	1244

Table I
TOPOLOGY CHARACTERISTICS.

B. Theoretical comparison between AFDT and standard Hello

The purpose of this section is to compare the performance of the Adaptive Failure Detection Timers mechanism vs. the classical Hello behaviour with both fast and slow timers using the previously defined analytical model. In the analysis, the convergence time use the aforementioned values given in Sec. II and V and with a short *Hello Interval* set to 100 miliseconds (t_{FHI}) and a stable *Hello Interval* (t_{SHI}) of three seconds. Mean Time Between Failure (MTBF), Mean Time To Repair (MTTR) and Δt_p are respectively assigned to 5000 hours, 5 hours and 1 hour as reference values. Nevertheless, for exhaustivity reason, we mind to vary the MTBF from 1000 to 10000 hours, the MTTR from 1 to 10 hours and Δt_p from 5 minutes to 10 hours on the three topologies but for space reason we will only show the most valuable results.

Since the main benefit of the AFDT solution is a lower unavailability probability than the slow timers case, and a much lower quantity of Hello to process than the fast timers configuration, we plot for each network x (x = a, b and c) on Fig. 4 what gives slow, fast and AFDT cases for the Hello rate (X-axis) and the bit unavailability probability (Y-axis). Moreover, for AFDT, three different levels of performance are considered for both *Recall* and *Precision* of the online failure predictor: low level at 20%, medium level at 50% and a high level at 80%, giving nine points for the AFDT cases.

Thanks to the AFDT mechanisms we observe on the three topologies (see Fig. 4), a unavailability gain of more than 4 for *Recall* of 80%, almost 2 for the 50% and 1.4 for *Recall* of only 20%. The AFDT availability performances are not similar to the 100ms timers performance, but allow intermediates performances, up to a factor 4 close to fast instable timers, while providing a much better availability than the stable slow timers. When we analyze

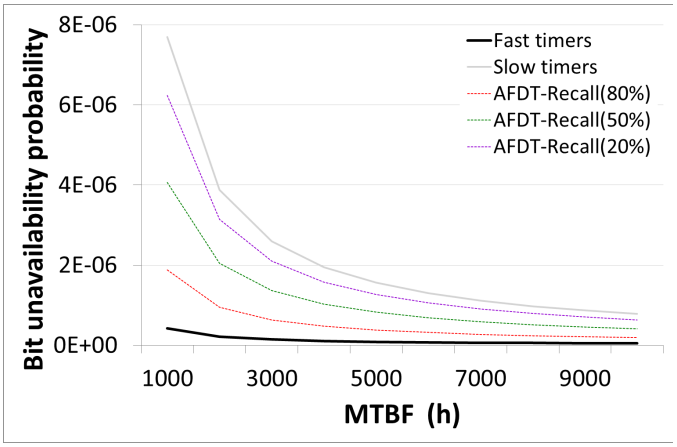


Figure 5. MTBF impact on network c availability.

the Hello quantity that have to be processed on each router, we observe on Fig. 4 that AFDT and normal slow timers have a similar impact on processing unit, which is more than 30 times lower than with short *Hello Interval*. The only differences between the slow timers and the AFDT cases are limited in time and space to the (true or false) prediction period. But this interlude is negligible

in comparison to the full fast timer. Concerning the failure predictor performance, its impact on the Hello quantity is very low and we need to zoom in Fig. 4 to make it visible. While *Recall* values have an important impact on the availability performance of the AFDT scheme, *Precision* impact is negligible. Indeed, the Hello burst due to false predictions is spatially and timely very contained.

After seeing the references condition results, it is interesting to look into the AFDT of one topology (c) behaviour when failure rate evolves. The Fig. 5 show how MTBF influences unavailability of each strategy. Since availability is only function of *Recall*, the AFDT mechanism is represented by only three curves. Although the differences of performance increase with failure rate, the ratio between each mechanism remains constant.

The Fig. 6 is rather focus on the Hello quantity evolution resulting from the variation of failure rate. The MTBF decrease associated with the different configurations of failure prediction performance show a relatively reduced increase of Hello rate to process compare to the about 1700 messages that the short *Hello Interval* configurations have to handle.

It is also useful to notice that the failure prediction performance play an important role on the Hello quantity but it is mostly the quantity of whatever prediction (TP and FP) which is responsible of the Hello burst. Indeed, one can see that the two most Hello expensive configurations are the ones with the lowest *Precision* with 80% and 50% for *Recall*. The third is a configuration having a *Precision* of 50% and *Recall* of 80% because this configuration generates more predictions that the one having 20% for both *Precision* and *Recall*.

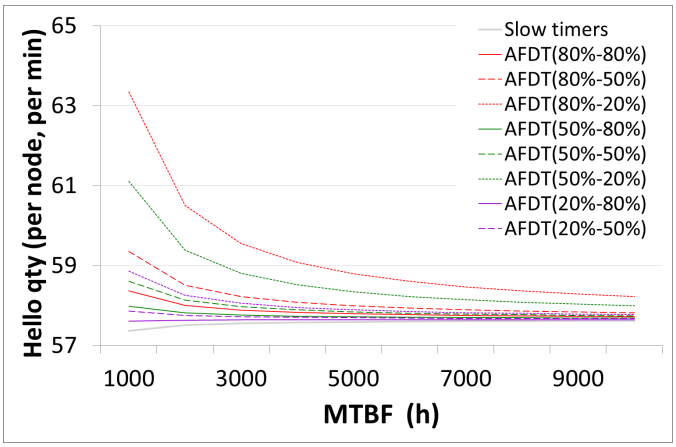


Figure 6. MTBF impact on Hello quantity with network c.

But the most impacting parameter on the Hello quantity is the prediction period Δt_p as illustrated by the Fig. 7. With long Δt_p of several hours, even if we are far away from the 1727 messages of the fast timers, the high average increase of Hello quantity might be symptomatic of major Hello burst that could generate routing flaps. The AFDT simulations describe in the next section allow to study that particular point.

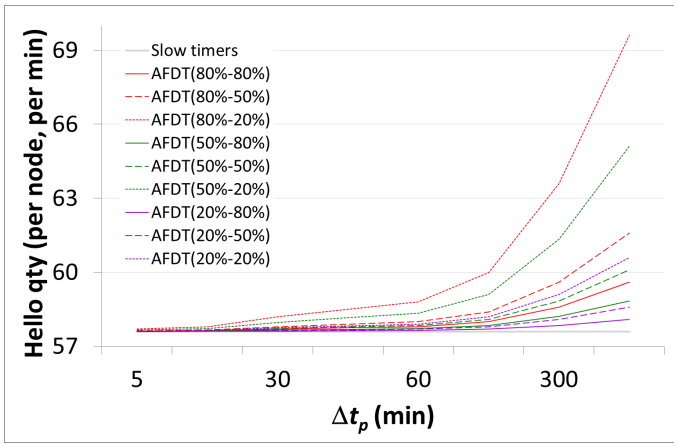


Figure 7. Δt_p impact on Hello quantity with network c.

C. Simulation experiments

In order to verify the analytical hypothesis, the AFDT mechanism has been implemented in the NS3 discrete-event simulator [19]. The general reliability theory [20] has been applied to generate failure events using an exponential distribution ($\lambda = 1/MTBF$) for time between failures and lognormal distribution $\ln N(\mu, \sigma^2)$ with $\mu = \log(MTTR) - ((0.5) * \log(1 + ((0.6 * MTTR)^2/MTTR^2)))$ and $\sigma = \sqrt{\log(1 + ((0.6 * MTTR)^2/MTTR^2))}$ for time to repair. Concerning failure prediction, anticipated failures have been uniformly chosen following the *Recall* ratio and the false predictions have been uniformly generated during the simulated time to reach the targeted *Precision*. Using the same network topologies and traffic matrices than in the analytical evaluation, the impacts of MTBF, MTTR, *Recall*, *Precision* and Δt_p have been analyzed based on 7 simulation runs with a simulated time equals to $5 * (MTBF + MTTR)$.

Using the same configurations as the analytical experiments of Fig. 4, the simulation results with confidence interval of 99% are presented in Fig. 8. The adequacy of these results with the analytical results illustrates the global observation of simulation outcomes, which confirms the validity of the proposed analytical model.

On top of the same configurations that the ones analytically studied, the simulations allow a more precise analysis of Hello bursts. First, they show the limited distribution in space, by validating that Hello bursts never append on more than one interface. A Hello burst leads to the processing of an average of 643 Hello messages by minute within network c, while the fast timers strategy generates three times this value. Then, it is important to ensure that burst are also time limited. Fig. 9 shows the total time of the Hello burst period function of Δt_p. We can see the burst duration accounts for less than 0.5% of the total time until Δt_p of two hours, but then can reach

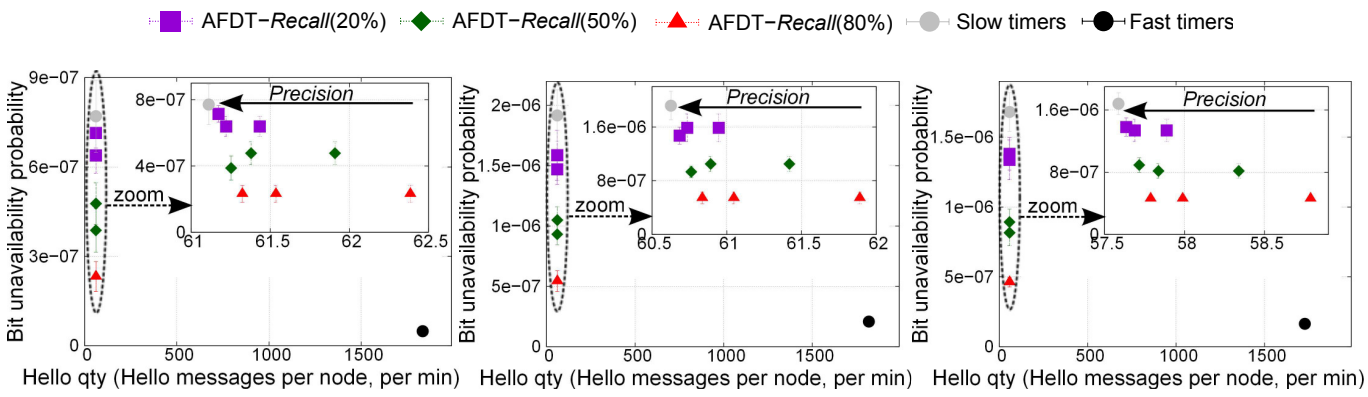


Figure 8. Hello quantity and unavailability simulation ($MTTR = 5000h$, $MTTR = 5h$, $\Delta t_p = 1h$) for the 3 networks.

up to 2%. Although 2% is quite low, it is possible that some operators become reluctant to tolerate such value; it is therefore preferable to only use failure predictors with Δt_p of 2 hours at maximum, in order to ensure an utmost stability.

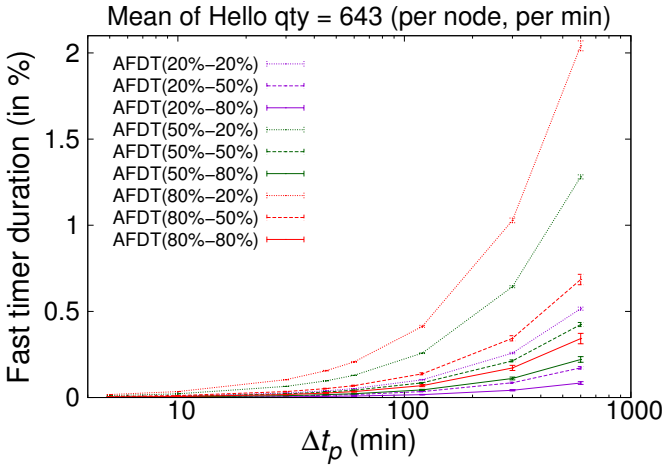


Figure 9. Δt_p impact on Hello burst duration with network c.

VII. Conclusion

The paper present a new proactive mechanism exploiting failure prediction, its evaluation and the gain quantification whatever the failure prediction performances, i.e. for any existing or future failure prediction mechanisms, as well as the impact quantification of the failure prediction performances on the proactive mechanism behavior. This paper differs from common proactive self-healing contribution by focusing on the resilience action and its evaluation instead of the failure prediction aspect. The adaptation of failure detection timer to the risk of failure proposed in this paper is intended to improve the convergence time of IGP. Failure detection is the most critical step during IP restoration and can be accelerated by increasing the frequency of sending messages at the expense of system stability. Observation of router health allows to use the risk of failure information in real time to speed up the Hello rate during a limited period and for a small subset of equipment. The behaviour of this mechanism has been analytically modelled in order to quantify its impact on the availability and on router workload that is responsible of instabilities. The results of this study were then confronted with the simulation results. The AFDT mechanism improves availability with few drawbacks. Gains on availability are proportional to the percentage of failures that were anticipated by the failure prediction module of the RAM. Impacts on routing stability are minor regardless of the network topology, as the Hello rate acceleration only concerns the unsafe equipments. However, extreme conditions, including both low *Precision* and high *Recall* (*i.e.* a large number of predictions), as well as a high failure rate or a very long failure prediction period (*i.e.* Δt_p) show potential limitations that can deter some very conservative operators to use the mechanism in such conditions. The interest of this proposition is that it leverages a proactive risk of failure information to autonomously adapt a parameter that improves network availability but that is currently used in a static way. The consequences of this intervention do not alter the traffic flows thereby limiting the impact of false predictions on the quality of service.

References

[1] G. Rétvári, F. Németh, A. Prakash, R. Chaparadza, I. Hokelek, M. Fecko, M. Wódczak, and B. Vidalenc, "A guideline for realizing the vision of autonomic networking: Implementing Self-Adaptive routing on top of OSPF," in *Formal and Practical Aspects of Autonomic Computing and Networking*. IGI Global, May 2011.

[2] S. Poretsky, B. Imhoff, and K. Michielsen, "Terminology for benchmarking Link-State IGP data plane route convergence," 2011.

[3] A. Shaikh and A. Greenberg, "Experience in black-box OSPF measurement," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, San Francisco, California, USA, 2001, pp. 113–125.

[4] M. Goyal, K. Ramakrishnan, and W. chi Feng, "Achieving faster failure detection in OSPF networks," in *IEEE International Conference on Communications*, vol. 1, 2003, pp. 296–300.

[5] A. Medem, R. Teixeira, N. Feamster, and M. Meulle, "Joint analysis of network incidents and intradomain routing changes,"

in *Proceedings of the International Conference on Network and Service Management, CNSM*, 2010, pp. 198–205.

[6] C. Alaettinoglu and S. Casner, "Detailed analysis of ISIS routing protocol on the qwest backbone," 2002.

[7] Y. Ohara, M. Bhatia, N. Osamu, and J. Murai, "Route flapping effects on OSPF," in *Proceedings of the Symposium on Applications and the Internet Workshops*, 2003, pp. 232–237.

[8] A. Basu and J. Riecke, "Stability issues in OSPF routing," in *ACM SIGCOMM Computer Communication Review*, vol. 31, New York, USA, Aug. 2001, p. 225–236.

[9] A. Shaikh, A. Varma, L. Kalampoukas, and R. Dube, "Routing stability in congested networks: experimentation and analysis," in *ACM SIGCOMM Computer Communication Review*, vol. 30, New York, USA, Aug. 2000, p. 163–174.

[10] S. Rai, B. Mukherjee, and O. Deshpande, "IP resilience within an autonomous system: current approaches, challenges, and future directions," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 142–149, 2005.

[11] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second IGP convergence in large IP networks," *SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 35–44, 2005.

[12] C. Alaettinoglu, H. Yu, and V. Jacobson, "Towards milli-second IGP convergence," *NANOG*, 2000.

[13] A. Siddiqi and B. Nandy, "Improving network convergence time and network stability of an OSPF-Routed IP network," in *Proceedings of NETWORKING*, Waterloo, Canada, 2005, p. 469–485.

[14] F. Wang, S. Chen, X. Li, and Y. Li, "A route flap suppression mechanism based on dynamic timers in OSPF network," in *Proceedings of the 9th International Conference for Young Computer Scientists, ICYCS*, 2008, pp. 2154–2159.

[15] D. Katz and D. Ward, "Bidirectional Forwarding Detection (BFD)," RFC 5880, Jun. 2010.

[16] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Computer Survey*, vol. 42, no. 3, pp. 1–42, 2010.

[17] N. Tcholtchev, M. Grajzer, and B. Vidalenc, "Towards a unified architecture for resilience, survivability and autonomic fault-management for self-managing networks," in *Proceedings of the 2009 international conference on Service-oriented computing*, Stockholm, Sweden, 2009, pp. 335–344.

[18] F. Salfner, M. Schieschke, and M. Malek, "Predicting failures of computer systems: a case study for a telecommunication system," in *International Parallel and Distributed Processing Symposium*, 2006.

[19] http://www.nsnam.org.

[20] M. Ohring and J. R. Lloyd, *Reliability and Failure of Electronic Materials and Devices*. Academic Press, 2009.