Using AR(I)MA-GARCH models for improving the IP routing table update

Wouter Tavernier*, Dimitri Papadimitriou†, Didier Colle*, Mario Pickavet* and Piet Demeester*
* Ghent University, IBCN-IBBT, INTEC, Gaston Crommenlaan 8 bus 201, B-9050 Gent, Belgium
Email: {wouter.tavernier, didier.colle, mario.pickavet, piet.demeester}@intec.ugent.be
† Alcatel-Lucent Bell, Copernicuslaan 50, B-2018 Antwerpen, Belgium
Email: dimitri.papadimitriou@alcatel-lucent.be

***Abstract*—When an IP router updates its routing table, e.g., upon failure detection, network traffic is lost as long as the routing entries affected by the failure are not updated. In this paper, we model and predict the network traffic passing through an IP backbone router and define a dynamic heuristic in order to reduce the packet loss resulting from such routing tables entries update events. We use the state-of-the-art AutoRegressive Integrated Moving Average (ARIMA) - Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) model to characterize and predict the network traffic. We benchmarked the resulting models with a heuristic in a simulation environment and show that a significant decrease of packet loss can be obtained for a given computational cost.**

I. Introduction

Many techniques have been developed to help networks reducing their recovery time resulting from network failures. Fast failure detection techniques ensure detection times in the order of milliseconds, for example, using hardware-based loss-of-signal detection or software-based Bidirectional Failure Detection (BFD, see [1]). Recovery techniques such as, Generalized Multi-Protocol Label Switching (GMPLS)-based segment protection and restoration ([2]), or plain IP Fast ReRoute (FRR, [3]), aim at minimizing the duration of traffic delivery disruption caused by link or node failure(s) until the network reconverges on the new topology. Common to all these techniques is their focus on recovery time and availability of a (loop free) backup path.

While the backup path can be known (pre-calculated) at the moment of failure detection, the low-level update of routing entries at the IP level can still take more than 1 second in IP backbone networks (see [4]). The study conducted in [5] was the first to show that in simplified network conditions[1], a significant decrease of packet loss can be obtained when the order of updating the routing table entries in an IP router is optimized based on measured network traffic statistics. The main idea of this work relied on ensuring that higher bitrate traffic flows are updated earlier than lower bit-rate flows (i.e., the corresponding routing table entries are updated earlier). The study was performed under the assumptions that network traffic behaves persistent during the process of the router update (for a period of 1 second) using generated network traffic using Pareto-distributions.

[1]persistent network traffic assumption during the routing table update

In this paper, we show (using MAWI traces, see Section V) that real network traffic of an IP backbone network does fluctuate during periods of 1 second. Therefore, a more accurate network traffic pattern is needed to model and predict network traffic during the process of updating IP routing entries. In addition, heuristics are defined which use the network traffic models to dynamically optimize: i) the update order of the routing entries, and ii) the low-level router process quantum with the goal of decreasing packet loss the resulting packet loss.

The paper is structured as follows. In Section II, we briefly recapitulate the structure of the IP routing table update process. In Section III, we describe AutoRegressive Integrated Moving Average (ARIMA) - Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) models for network traffic modeling and prediction. Next, the first part of Section IV formalizes the concepts of recovery time and packet loss under dynamic network traffic conditions in the context of an IP routing table update. In the second part of this section, we use these concepts to formulate a dynamic heuristic that minimizes the resulting packet loss. In Section V, we analyze the dynamics of IP traffic in backbone networks, and validate the network traffic models of Section III. Next, we measure the resulting packet loss and recovery time using the defined heuristics in combination with modeled network traffic. The computational cost of the proposed scheme is characterized in Section VI. Conclusions together with future work are formulated in Section VII.

II. IP routing table update

We first describe how an IP router updates its routing table in order to determine what aspects can be optimized to reduce the resulting packet loss. In an IP backbone network operating Link-State (LS) Interior Gateway Protocols (IGP) such as OSPF ([6]), a router detecting a link failure originates a LS Update (LSU) message. This message, containing LS Advertisement(s) (LSA) describing the topological link state change(s), is reliably disseminated in the network (flooding). At every router receiving the LSU, the following three-step process executes (see Figure 1):

1) Re-computation of the *shortest path tree* (1) using the topological information stored in the updated LS DataBase (LSDB);

978-1-4244-7286-4/10/$26.00

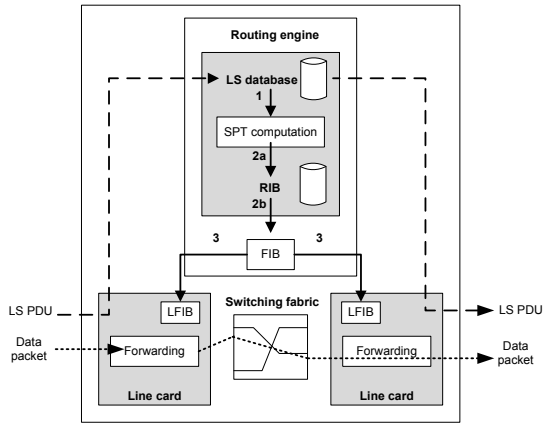

Fig. 1. The router update process

2) Update of the central *Routing Information Base (RIB)* and the central *Forwarding Information Base (FIB)* based on the Shortest Path Tree (SPT) computation (2a and 2b).
3) *Distribution of central FIB* towards the local FIB (LFIB) on line cards (3).

The SPT recomputation is usually performed in its entirety and takes about 30 to 50 μs per IGP destination prefix. Time optimizations can be obtained using incremental SPF (iSPF) algorithms (see [7]). The second step consists of updating the central RIB and FIB, using the calculated shortest paths. This uses about 50 to 100 μs per destination prefix (see [4]). Typically, this step happens in (pseudo-) parallel with step 3, which is about distributing the central FIB entries towards the line cards' LFIB. Running step 2 and 3 in (pseudo-) parallel, means that they both use the central CPU in interleaved time slots, swapping between both processes for updating and distribution. This process can be compared to the usual process scheduling in time-sharing Operating Systems (OS) such as Linux, whereas commercial routers make use of a hard real time OS. The consecutive time the central CPU is spending to perform central RIB/FIB entries update(s) or distribution of FIB entries towards the line card(s) is determined by the quantum of the swapping process. The quantum time can typically be configured between 10 and 800 ms. Similar quantum time values were found in [4]. In practice, the process consists of a series of update-distribution batches, where in a first quantum a fixed set of prefixes are updated towards the central RIB/FIB, followed by a quantum where the same set of prefixes is distributed towards the LFIBs.

By default, the update of RIB/FIB entries (i.e., IGP prefixes) and their distribution are not ordered. Moreover, the size of the batches is determined by the router process quantum, usually a pre-configured constant number over the entire update process. Therefore, the heuristic defined in Section IV-B basically tries to sort the routing entries in decreasing bitrate order on the moment of the update, as well as optimizing the sizes of the series of update-distribution batches.

III. Network traffic models

Using network monitoring techniques, the volume of traffic per IP destination prefix (corresponding to the routing entry) can be measured for a given time interval. Network traffic aggregation thus happens at two levels: i) at the prefix-level (smaller subnets result into more aggregation) and ii) at the time-level (larger time intervals result into more aggregation).

If we want to ensure that the router update process re-orders RIB/FIB entries according to their volume (integrated bitrate), we need to make sure that the associated traffic volumes are correctly estimated. In Section V-B, we show that network traffic heavily fluctuates during periods of 1 second, such that we can no longer assume that traffic behaves exactly as just before the update process as assumed in [5]. Therefore, this section describes two network traffic models which can be combined to model and predict network traffic within short time scale (sub-second). Using these models, we ensure that the technique described in Section IV-B performs the best possible estimation in order to reduce packet loss.

A. Auto-Regressive (Integrated) Moving Average model

The *AutoRegressive Moving Average* $ARMA(p,q)$ model is defined as follows:

$$y_t = \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{j=1}^{q} \beta_j w_{t-j} + w_t$$

This formula combines two techniques: i) autoregression, which reflects the fact that a prediction is based on the signal itself (using p previous values) referring to the first term, and ii) moving averages, reflected by q terms of the white noise series w_t (with $E(w_t) = 0$ and $Var(w_t) = \sigma^2$) which is put through a linear non-recursive filter determined by the coefficients α_i (weighted average). The autoregressive part directly reflects the *Short Range Dependence* (SRD) of a time series. Whereas these models have been successfully applied to model network traffic (e.g., [8]), they are also known to be unable of modeling non-stationary time series. Stationarity implies that the probability distribution characterizing the time series mean and variance remains constant over time.

Some non-stationary time series can be made stationary by one or more levels of differencing[2]. Once the resulting differenced time series is stationary, an $ARMA(p,q)$ model can subsequently be fit and predictions can be made by integrating the predictions back. The resulting model of this approach is called the *AutoRegressive Integrated Moving Average* (ARIMA) model. The resulting model including the lag operator L[3] for $ARIMA(p,d,q)$ is defined as follows (d referring to the number of levels of differencing):

$$(1 - \sum_{i=1}^{p} \alpha_i L^i)(1-L)^d y_t = (1 + \sum_{j=1}^{q} \beta_j L^j) w_t$$

[2]A differenced time series y_t generates a new time series of differences $z_t = y_t - y_{t-1}$

[3]applying lag operator L on y_t generates y_{t-1}, and a power i of L defines a similar recursive process of the order i

B. Generalized Autoregressive Conditional Heteroskedasticity traffic model

The ARIMA model cannot capture the multifractality which has been observed in some network traffic traces. For this reason, the Multifractal Wavelet model (MWM, [9]) has been introduced. However, while the MWM model can capture short timescale multifractality, it cannot predict traffic. For this purpose, the work of [10] introduces the ARIMA - Generalized AutoRegressive Conditional Heteroscedasticity (GARCH) model, a non-linear time series model which combines the linear ARIMA model (with constant variance) with conditional variance GARCH-model (see [11]). The term "conditional" implies an explicit dependence of the variance of the noise/innovation series w_t from the ARIMA model on a past sequence of observations. Formally, a $GARCH(r, s)$ model for the conditional variance σ^2 of the innovation series w_t follows an $ARMA(p = r, q = s)$ model of the following form:

$$\sigma_t^2 = \sum_{i=1}^{r} \gamma_i \sigma_{t-i}^2 + \sum_{j=1}^{s} \omega_j w_{t-j}^2 + w_t$$

where, r and s are positive integers.

IV. Packet loss minimization

Given a network traffic model which is able to capture and predict the traffic behavior during the update of IP routing entries upon recovery, the goal is now to use this information to formulate a packet loss minimizing heuristic. Therefore, we first recapitulate and formalize the concepts of the recovery time and packet loss associated to a routing entry.

A. Recovery time and packet loss

Given the batch structure of the IP router update process (see Section II), all traffic flows directed to a given destination address are recovered when the following conditions are met: i) the IGP routing entry for the prefix including that destination address is updated and stored in the central RIB and FIB, and ii) the batch number b_i that comprises this updated entry is distributed towards the LFIBs on the line cards. Thus, assuming a fixed batch size of x_u (number) routing entries and a given order of entries, the recovery time of a flow f_i is characterized by the following formula:

$$r(f_i) = b_i x_u (t_u + t_d) + (2b_i - 1)t_s$$

Here, t_u refers to the update time for a single IGP routing entry, t_d to the distribution time for a single entry, and t_s to the swapping time interval between an update and a batch distribution operation. In addition, we can define the *Earliest Recovery Time* (ERT) possible for a flow f_j, j referring to the waiting position of the flow at time t, as follows:

$$ERT(f_j, t) = t + j(t_u + t_d) + t_s$$

The ERT is the update time of the smallest batch including all IGP routing entries updated before the considered flow together with the considered flow. Indeed, in this situation only one swapping time interval is needed (separating the update and distribution batch) together with the time needed to update and distribute all prefixes comprised within the batch.

Packet loss occurs for network traffic flows as long as their corresponding routing entries are not updated and distributed on line cards. The loss resulting from the recovery of flow f_i relates to its bitrate, $br(f_i)$, through the following formula:

$$loss(f_i) = \int_0^{r(f_i)} br(f_i)dt$$

B. Packet minimization heuristic

Building further on the heuristic from [5], we can formulate a heuristic that takes into account both flow ordering and batch size so as to minimize the packet losses. For this purpose, consider the following scenario. We denote by $F_n = f_1, \ldots, f_n$ the set of traffic flows affected by the failure, and by $b_{current} = (f_i, \ldots, f_{i+s})$ the set of flows for which the corresponding IGP routing entries still need to be updated[4]. In this context, we have two possible choices when in the middle of the ordered process of updating the routing entries (associated to the set of flows F_n) within our current batch $b_{current}$:

1) *Extension*: extend the current batch with the IGP routing entry associated to the next flow f_{i+s+1};
2) *Splitting*: terminate the current batch and put the IGP routing entry associated to the next flow into a new update-distribution batch.

We can compare the additional cost of extension vs. the additional cost of finishing the update-distribution batch to guide us into the decision above. By defining $t_{bcurrent}$ as the starting time of the current batch $b_{current}$, the extension cost for this batch $ec(b_{current})$ can be formulated as follows:

$$ec(b_{current}) = \sum_{j=i}^{i+s} \int_a^b br(f_j, t)dt$$

with $a = ERT(f_j, t_{bcurrent})$ and $b = ERT(f_{i+k}, t_{bcurrent})$ + $(i + s - j + 1)(t_u + t_d)$. The formula expresses the fact that, by extending the current batch, the recovery time of every IGP routing entry comprised in the current batch will result into an additional delay compared to the minimal delay it can experience (given the position of that entry in the current batch). The minimal delay an entry can experience is determined by the ERT, i.e., the time elapsing for an entry positioned as the last one in an update quantum having no earlier update quanta. This additional delay, when multiplied with the associated bitrate, allows deducing the additional loss caused by the extension of the update-distribution batch. For example, if the routing entry associated to the first flow f_i was already delayed by s update times t_u (as this entry was not directly distributed but put in the same batch as the s next entries ranging from i to $i + s$), extending the current batch

[4]The IGP routing entries for the prefixes corresponding to the flows prior to f_i have already been updated in an ordered manner (from f_1 to $f_{(i-1)}$)

by one element (to reach $i+s+1$ elements) further delays the recovery time of the entry i. On the contrary, the recovery of the last entry $i+s$ of the current batch will only be delayed by one update time t_u in case of extending the current batch.

On the other hand, terminating the current batch has also an associated cost, as it will introduce additional delay for the coming flows, resulting from the additional swapping cost. This termination condition can be formulated as follows:

$$fin_{b_{current}} = \sum_{f_j \notin b_{current}} \int_{ERT(f_j,t_{bcurrent})}^{ERT(f_j,t_{bcurrent})+2t_s.} br(f_j,t)dt$$

Our configuration strategy now consists in identifying the action with the least associated cost. The overall algorithm can be expressed as follows:

1) Add all flows to the working set
2) Sort all flows in the working set in decreasing order of their current cumulative loss $cumloss(f_j, t_{current})$
3) Compute both extension and splitting cost
   - If no current batch exists (routing entry associated to the first flow), then create a batch and add this routing entry into this newly created batch.
   - Otherwise:
     - If the extension cost is smaller than the splitting cost, then add the corresponding routing entry in the sorted list to the current batch;
     - Otherwise, create a new batch and add the corresponding routing entry into this newly created batch.
4) Remove the added flow from the working set
5) Repeat the procedure from step 2 until the working set is empty

V. Experimental results

In the previous section, theoretical foundations have been developed to model: i) the IP router update process, and ii) the network traffic flowing through the router. Both aspects were used to formulate a heuristic that is able to reduce packet loss during the process of updating the routing table entries.

In this section, real network traces are used to evaluated the performance gain (i.e., packet loss decrease) obtained by means of the proposed heuristic. First, we detail the overall methodology and experimental environment; then, we analyze the dynamic properties of network traffic and apply the models of Section III. At last, the gain of the overall procedure is measured.

A. Methodology and environment

A set of PCAP traces obtained in December 2009 was taken from the MAWI project (see [12]). These traces were processed, using Python scripting, at three time interval levels (100 ms, 500 ms and 1000 ms), referred to as time bins, and at three spatial levels (/8, /16 and /24 subnetworks) with respect to their aggregated traffic volume, i.e., aggregated packet sizes per time-bin per prefix. Next, the resulting traffic time series

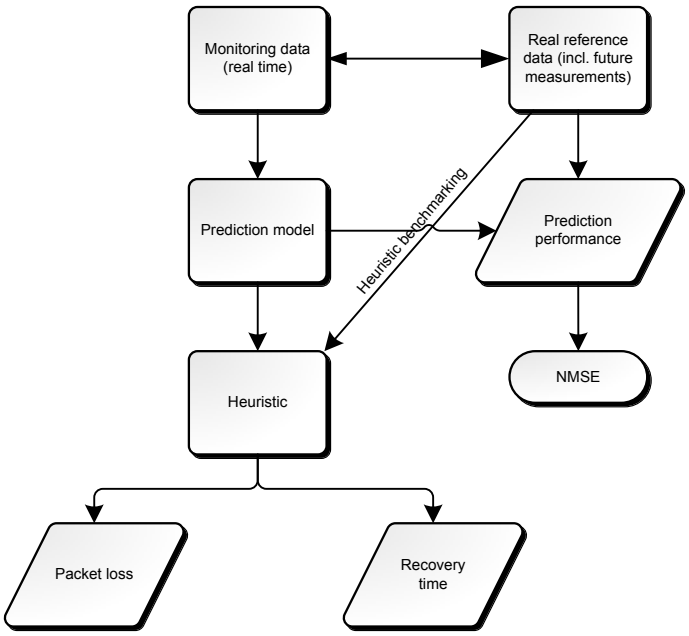

Fig. 2. Benchmarking process

were analyzed and fit to one of the described traffic models using R software ([13]). The resulting modeled data was used as input for the heuristic implemented in Python/C++, such that we could decrease the packet loss resulting from the update of the affected routing entries. All the experiments ran on a regular desktop computer equipped with an AMD Athlon 2.7 GHz CPU and 4 GB RAM. The entire process is shown in Section 2.

B. Network traffic analysis

The output of a processed PCAP file consists of a set of prefixes with associated traffic volume per time-bin (multiple parallel time series). On average, we counted about 140K /24 subnets, or 14K /16 subnets or 0.2K /8 subnets per trace. To analyze the dynamics of the traffic, we measured the persistence of active flows over time. A flow is considered as active when at a certain point in time, a non-zero volume is associated to it. We analyzed the percentage of active flows that remain active over the next time bin for several combinations of aggregation levels i.e., time bin sizes of 100 ms, 500 ms and 1000 ms together with /8, /16 and /24 subnets.

The analysis showed that for /16 and /24 subnets, for all time-bin sizes, the percentage of flows that remain active during two consecutive time bins is about 50 percent on average. Figure 3 shows the number of active flows at every time bin (solid black line), and the number of flows that remain active during the next time bin (dotted line) when using /24 subnets and a bin size of 1000 ms. Using /8 subnets, this percentage is on average 90. Table I gives the average values at other aggregation levels. The consequence of the high fluctuation in activity can be formulated as follows: one can no longer assume that the volume statistics of the moment just before a routing table update will remain the same during

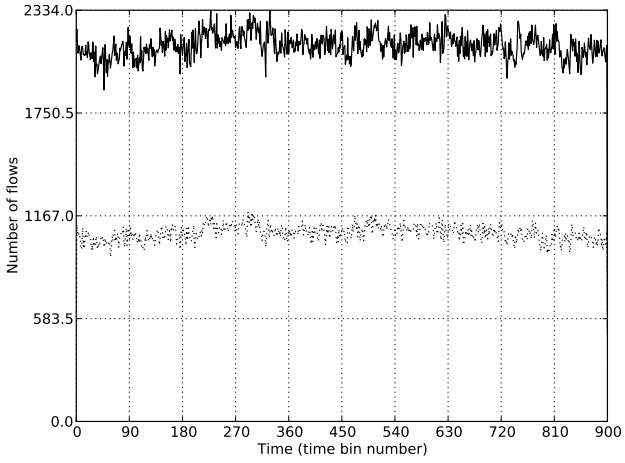

Fig. 3. The number of active and persistent flows using /24 subnets and 1000 ms time bins

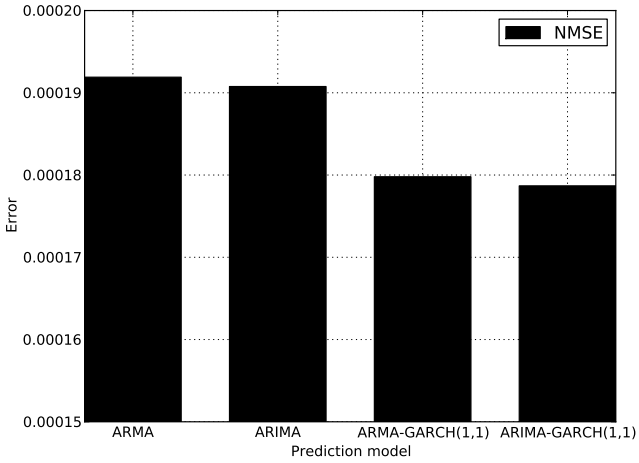

Fig. 4. NMSE of prediction models

the routing table update. For this reason, we fit network traffic to network traffic models discussed earlier and use it to predict their behavior for the next time bin.

C. Network traffic fitting and prediction

Four time-series modeling techniques were tested to fit and predict the referred MAWI-traces: i) ARMA, ii) ARIMA, iii) ARMA-GARCH, and iv) ARIMA-GARCH. It is clear from Section III that these models can have different orders as determined by their parameters p, q, d and r, s.

The main task in automatic ARIMA forecasting is to select an appropriate model order. To automate the process of optimal parameter selection, we used the order selection strategy of [14] for ARMA and ARIMA models restricted to maximum values of 10. The strategy selects those parameters which minimize the Akaike's Information Criterium (AIC) among all fitting models. Once the best parameters are selected, Maximum Likelihood Estimators (MLE) are computed to deduce the best fitting coefficients for the ARMA and ARIMA models (see [14]). The lowest value for the d parameter of the ARIMA model, resulting into a stationary time series, according to the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test for stationarity[5], is then selected (see [14]).

For the GARCH-model, the values of parameters $r = 1$ and $s = 1$ were chosen to reduce the resulting computational complexity, as well as to avoid non-optimizable parameter values due to lack of sufficient information[6]. The GARCH$(1, 1)$-coefficients were estimated using the Quasi-Maximum Likelihood Estimator (QMLE) (see [15]).

For one MAWI trace, the fitting procedure at all aggregation levels, takes about 20 hours on the referred computer. However, in the context of the router update, this process can be executed in background.

Figure 4 shows the average Normalized Mean Square Errors (NMSE) of fitting the models to the set of MAWI traces. The NMSE is defined as follows[7]:

$$NMSE = \frac{\sum_n (\widehat{y}(x) - y(x))^2}{\sum_n (y(x) - \mu_T)^2}.$$

Smaller NMSE values correspond to better fitting models. As expected, the ARIMA-GARCH$(1, 1)$ model shows on average the best performance, followed by ARMA-GARCH$(1, 1)$, ARIMA, and ARMA model. More aggregation, either by using larger time bins or using smaller subnetworks (larger netmasks), leads to better fitting models; however, the relative gain between the techniques remains the same. From Figure 4, one can also observe that all prediction techniques are rather close in predictive performance. Using the same figure, the improvement resulting from the GARCH(1,1) model can also be delimited compared to the AR(I)MA model.

D. Packet loss simulation

Given the data of the prediction models from the previous subsection, the gain of the heuristic described in Section IV-B, can now be evaluated. Figure 5 shows the packet loss resulting from using either the default random update algorithm (unsorted routing entry updates using batches of 100 prefixes) vs. the described optimization heuristic using any of the described prediction models using /24 subnets and time bins of 100 ms. As a simpler benchmark, we also included a simple update described in [5] (mentioned as "*sort on latest RUP*") which relies on the latest measured volume data and updates them according to decreasing volume. This figure makes clear that on average, a decrease of 18 percent packet loss can be obtained using the heuristic with any of the proposed prediction techniques compared to the default random routing table update, and about 8 percent packet loss

[5]The KPSS test assesses the null hypothesis that a univariate time series trend is stationary against the alternative that it is a non-stationary unit-root process

[6]Not all time series corresponding to the volumes of a given prefix are dense enough. Some series have more zero values than others, resulting into non-optimizable parameters due to singular Hessian matrices.

[7]$\widehat{y}(x)$ referring to the predictions, $y(x)$ referring to the actual values

TABLE I
AVERAGE NUMBER OF ACTIVE AND PERSISTENT FLOWS

Subnet length (/x):	Binsize (ms):	Average number of active flows per time bin:	Average number of persistent flows:
8	100	100	90
8	500	118	108
8	1000	125	117
8	15000	204	-
16	100	390	170
16	500	900	537
16	1000	1210	720
16	15000	14412	-
24	100	517	213
24	500	1410	790
24	1000	2013	1005
24	15000	143203	-

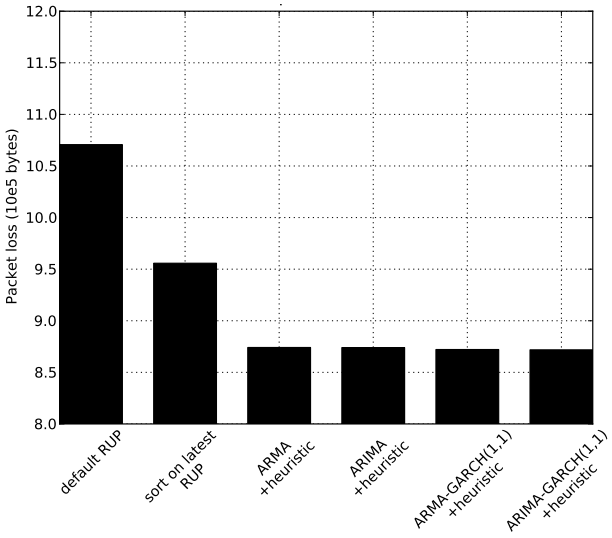

Fig. 5. Average packet loss vs Prediction model and heuristic

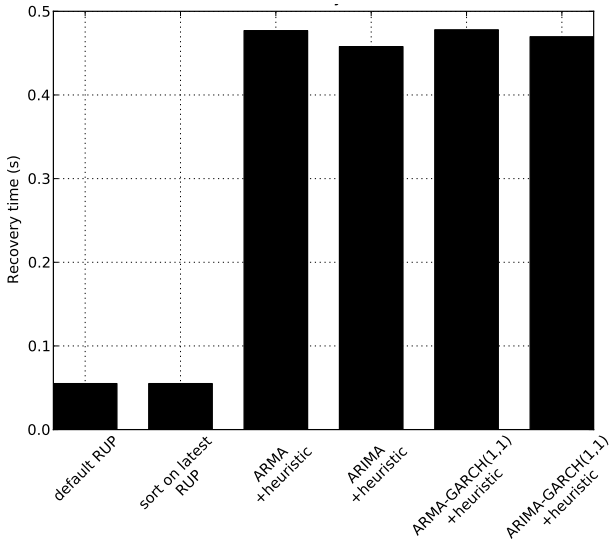

Fig. 6. Average recovery time vs Prediction model and heuristic

compared to *sort on latest RUP*. Whereas it is obvious that the default behavior of updating the routing entries for IGP prefixes in random order is constant independently of the traffic prediction model, it is surprising that the specific choice of the prediction model –in combination with the optimization heuristic– has relatively low influence on the packet loss. This observation can be explained from the fact that the difference between NMSE of the prediction models is rather small.

Whereas previous discussion makes clear that on average[8], significant decreases in packet loss can be obtained, it is important to determine the worst case: is it possible that the default random update shows in some case(s) lower packet loss than the combination of prediction and the optimization heuristic. This question is answered by the results obtained in Figure 8. This figure illustrates that a normal distribution was obtained around the mentioned averages in packet loss, with minimal and maximal improvements of respectively 5 and 35 percent in decrease of packet loss compared to the default routing table update. We found no executions where the default behavior is better than the suggestion solution. Similar distributions were found at other time and spatial aggregation levels for which the average decrease of packet loss is as depicted in Figure 7. The results illustrate that using a prediction model in combination with the described heuristic makes most sense at low aggregation levels, where up to 18 percent (average) decrease in packet loss can be achieved.

Figure 6 makes clear that minimizing the packet loss is not going hand in hand with reduction of the total recovery time (i.e., the time when all routing table entries are updated and loaded on line cards' LFIBs). This can be understood from the fact that the optimization heuristic sometimes introduces more but smaller batches, giving priority to certain routing entries. Smaller batches for the same number of prefixes lead to a larger number of batches, which result into more process swapping, which lead to lower total recovery times. This trend can be observed from Figure 5.

VI. COMPUTATIONAL COST OF THE PROCEDURE

The execution of the proposed procedure can be classified as relatively intensive from a computational perspective. However, many parts can run offline or in background. Fitting ARMA and ARIMA models to time series samples have

[8]over 1000 iterations

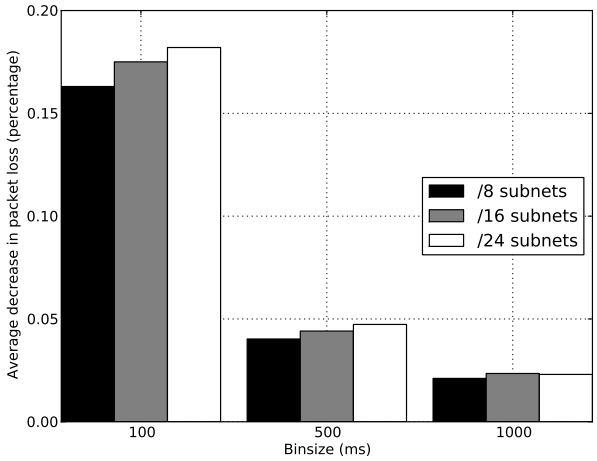

Fig. 7. Average packet decrease vs. default router update process over 1000 iterations

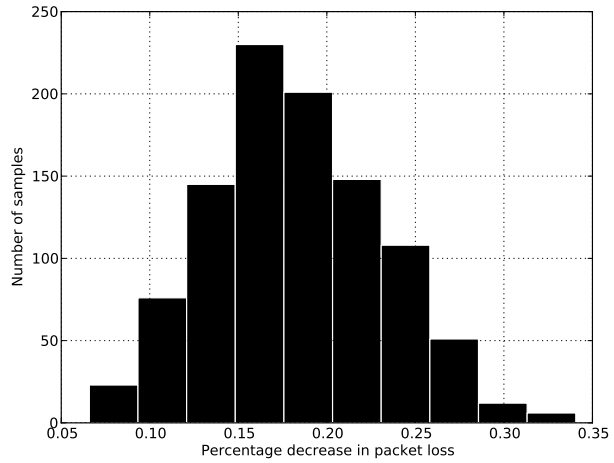

Fig. 8. Distribution of decrease in packet loss (subnet length 24 and binsize 100 ms)

a computational complexity which is bounded by $O(m^3T)$, where T is the length of the time series sample, and $m = max(p, q + 1)$. Fitting GARCH models reduces to hard non-linear optimization problems. To the knowledge of the authors there is no clear bound to the computational complexity of fitting these models for arbitrary (r, s) values.

The computation of the described optimization heuristic involves operations which are bounded by $O(n\ log\ n)$ during every step split/extension-step, caused by the need for real-time sorting. While this results in a heavy procedure, it can be drastically reduced by using hardware-implemented sorting algorithms which reduce to constant time sorting for a given number of values (see [16]).

VII. Conclusion and future work

In current routers, the update of the routing table entries can take up to one second. In this paper, we have shown that a significant amount of packet loss can be avoided using a dynamic heuristic which takes into account a time-dependent model of monitored network traffic. We used AR(I)MA-GARCH models to characterize traffic and showed that significant packet loss decreases can be obtained, varying from 18 percent on /24 subnetworks using 100 ms time bins, to 2 percent using 1000 ms bins. Figure 8 provides an evidence that in some cases even larger gains can be obtained. In general, we may conclude that shorter time aggregation and smaller subnets (larger netmasks) lead to larger improvements.

Acknowledgment

This work is supported by the European Commission (EC) Seventh Framework Programme (FP7) ECODE project (Grant n223936).

References

[1] D. Katz and D. Ward, "Bidirectional Forwarding Detection," Internet Engineering Task Force, Internet-Draft draft-ietf-bfd-base-08, Mar. 2008, work in progress.

[2] L. Berger, I. Bryskin, D. Papadimitriou, and A. Farrel, "GMPLS Segment Recovery," RFC 4873 (Proposed Standard), Internet Engineering Task Force, May 2007.

[3] M. Shand, "IP Fast Reroute Framework," Internet Engineering Task Force, Internet-Draft draft-ietf-rtgwg-ipfrr-framework-08, Feb. 2008, work in progress.

[4] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second igp convergence in large ip networks," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 33–44, July 2005.

[5] W. Tavernier, D. Papadimitriou, D. Colle, M. Pickavet, and P. Demeester, "Optimizing the ip router update process with traffic-driven updates," in *DRCN 2009*, Washington D.C., 2009.

[6] J. Moy, "OSPF Version 2," Internet Engineering Task Force, RFC 2328, Apr. 1998.

[7] J. M. McQuillan, I. Richer, and E. C. Rosen, "An overview of the new routing algorithm for the arpanet," *SIGCOMM Comput. Commun. Rev.*, vol. 25, no. 1, pp. 54–60, 1995.

[8] S. Basu, A. Mukherjee, and S. Klivansky, "Time series models for internet traffic," in *INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, vol. 2, 24-28 1996, pp. 611 –620 vol.2.

[9] R. Riedi, M. Crouse, V. Ribeiro, and R. Baraniuk, "A multifractal wavelet model with application to network traffic," *Information Theory, IEEE Transactions on*, vol. 45, no. 3, pp. 992 –1018, apr 1999.

[10] B. Zhou, D. He, and Z. Sun, "Traffic predictability based on arima/garch model," in *Next Generation Internet Design and Engineering, 2006. NGI '06. 2006 2nd Conference on*, 0-0 2006, pp. 8 pp. –207.

[11] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, April 1986.

[12] K. Cho, K. Mitsuya, and A. Kato, "Traffic data repository at the wide project," in *ATEC '00: Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2000, pp. 51–51.

[13] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2009, ISBN 3-900051-07-0.

[14] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: the forecast package for r," Monash University, Department of Econometrics, Monash Econometrics Working Papers 6/07, Jun. 2007.

[15] T. Bollerslev and J. Wooldridge, "Quasi-maximum likelihood estimation and inference in dynamic models with time-varying covariances," *Econometric Reviews*, vol. 11, no. 2, pp. 143–172, 1992.

[16] Y.-H. Tseng and J.-L. Wu, "On a constant-time, low-complexity winner-take-all neural network," *Computers, IEEE Transactions on*, vol. 44, no. 4, pp. 601 –604, apr 1995.