Distributed detection of latency shifts in networks

SICS Technical Report T2009:12
ISSN 1100-3154

December 23, 2009

Rebecca Steinert and Daniel Gillblad

Swedish Institute of Computer Science,
Box 1263, SE-164 29 Kista, Sweden
{rebste, dgi}@sics.se

Abstract. We present the extension of a distributed adaptive fault-detection algorithm applied in networked systems. In previous work, we developed an approach to probabilistic detection of communication faults based on measured probe response delays and packet drops. The algorithm is here extended to detect network latency shifts and adapt to long-term changes of the expected probe response delay. Initial performance tests indicate that detected latency shifts and communication faults successfully can be localised to links and nodes. Further, the amount of network traffic produced by the algorithm scales linearly with the network size.

Keywords: Adaptive probing, distributed fault-detection, anomaly detection, fault-localisation.

1 Background

We have developed a distributed approach to adaptive anomaly detection and collaborative fault-localisation. The statistical method used is based on parameter estimation of Gamma distributions obtained by measuring response delays (or two-way link latency) through probing. The idea is to adaptively learn the expected latency of each link from each node, such that the manual configuration effort is minimised. Instead of specifying algorithm parameters in time intervals and specific thresholds for when traffic deviations should be considered as anomalous, parameters are here specified either as a cost or as a fraction of the expected probe response delay.

In this report we will describe the extension to the existing approach to adaptive fault-handling described in [7]. Apart from detecting communication faults, our model is here extended to include detection of shifts in observed network latencies. Shifts in local network latencies can be symptoms of e.g. malfunctioning equipment, malicious activities, misconfiguration or varying user behaviour.
Being able to capture such events can e.g. increase the efficiency of network management and fault-handling. Further, we will describe the development of a statistical learning approach with palimpsest properties for autonomous adaptation to long-term latency variations. Thus, the extended approach can both detect latency shifts on individual links and adapt to the new 'regime', while gradually forgetting older observations.

2 Approach

The approach is based on probing for two purposes. First, probes are sent between nodes in order to measure the probe response delay and drop rate on each link. Second, adaptive probe tests are performed in each node for detection of anomalous network behaviour.

Based on the expected probe response delay and the expected drop rate, probe tests and probing intervals are autonomously adapted to the current network conditions on individual links. To reduce communication overhead, we use two different intervals for probe tests and individual probes, as described in [7]. For detection of communication faults a probabilistic threshold is used to achieve reliable fault-detection with few false positives. Adaptive probe tests are performed in each node to test the availability of adjacent nodes and links. From the the collection of observed probe response delays, overlapping statistical models are compared to detect and adapt to long-term shifts in the expected response delays.

The approach that we use can find two types of network anomalies; communication faults and shifts in normally observed probe response delays. When a probe test on a connection fails, a communication fault has been detected and a fault-localisation process based on node collaboration is initiated for the purpose of pinpointing the fault to a link or node. Shifts in the normally observed probe response delay on a link is detected if the previous and current latency models differ significantly from each other. The detecting node will in that case report the latency shift on the link and notify the neighbouring node, to reduce control message overhead. In case all links between a node have detected latency shifts on all its connections more or less simultaneously, the node will report an alarm about the current state. The subroutines forming the detection and localisation processes are described in the algorithms 1, 2 and 3 shown in Appendix A.

2.1 Statistical model

The statistical model that we use is based on the probability density function $P(t)$ of probe response delays and the probability of packet drop $P(D)$. We assume independence between the probe response delays and drop rate. Here, $P(t)$ can be any type of distribution that matches the characteristics of the data. Assuming that the probe response delay is mostly a sum of independent exponential transmission delays such as queueing times, we have chosen $P(t)$ to
be Gamma distributed,

\[ P(t; \alpha, \beta) = t^{(\beta-1)} \frac{e^{-t/\alpha}}{\alpha^\beta \Gamma(\beta)}, \tag{1} \]

where \( \alpha \) and \( \beta \) are the scale and shape parameters, respectively. Similar assumptions about traffic latencies (on different network levels) matching Gamma, Weibull, or other exponential distributions have been made in a number of papers, e.g. [1, 2]. The probability of packet drop \( P(D) \) is estimated by counting the number of probe responses relative the total number of sent probes.

### 2.2 Parameter estimation

Observations of the probe response delay obtained by probing are continuously collected, forming a distribution from which the Gamma parameters \( \alpha \) and \( \beta \) are estimated. In order to reduce computational demands we use a simple method of moments approach, estimating \( \alpha \) and \( \beta \) from the first and second sample moments \( s_1 = \frac{1}{n} \sum t_i \) and \( s_2 = \frac{1}{n} \sum t_i^2 \) (e.g. [3, 4]). Given that \( \alpha \beta = s_1 \) and \( \alpha^2 \beta(\beta + 1) = s_2 \), the estimates \( \hat{\alpha} \) and \( \hat{\beta} \) are

\[
\hat{\alpha} = \frac{s_2 - s_1^2}{s_1}, \quad \hat{\beta} = \frac{s_1^2}{s_2 - s_1^2}. \tag{2}
\]

This approach produces parameter estimates with less precision than the maximum likelihood estimation approach. These estimations are frequently performed in each node. Since the computational capacity of the nodes may vary, we have chosen the method of moments approach in favor of computational efficiency.

To account for long-term variations in the network, each node models probe response delays as overlapping Gamma distributions, using the previous model as prior to the next model (fig. 1). Here the priors are:

\[
s_1^{(i+1)} = \frac{\sum_j t_j^{(i+1)} + s_1^i}{n + 1}, \quad s_2^{(i+1)} = \frac{\sum_j (t_j^{(i+1)})^2 + s_2^i}{n + 1}. \tag{3}
\]

The learning scheme is circular with \( M = \frac{N}{T} \) models, each based on \( N \) observations and the degree of 'forgetfulness' \( T \). The degree of overlap \( T \) controls the temporally palimpsest properties (i.e. forgetting models over time). By using previous model as prior input to the next model, a smooth transition between models is achieved while previous observations have smaller impact on the current parameter estimations. The benefit that this learning scheme offers is faster adaptation to new network 'regimes', caused by e.g. software upgrades and change of network equipment.

### 2.3 Detecting communication faults and latency shifts

In the following model, we assume that the probability of receiving a probe response \( R_{\Delta t} \) is

\[
P(R_{\Delta t}) = (1 - P(D)) \int_0^{\Delta t} P(t)dt. \tag{4}
\]
Anomalous related to communication faults on either links or nodes are detected using the following decision model:

\[
P(\neg R|\Lambda_1, \Lambda_2, \ldots, \Lambda_i) = \prod_i 1 - P(R^{(i)}_\Delta) < \psi
\]  

Assuming independence between probes \(R^{(i)}_\Delta\), the probability of not receiving any probe response \(P(\neg R)\) gradually decreases for each sent probe, until a probe response is obtained or until the probability of not receiving a response has reached below threshold \(\psi\). The number of probes needed to reach below the detection threshold \(\psi\) is adapted based on \(P(R_\Delta)\). Hence, the detection performance and the detection confidence \(\psi\) is a trade-off between the communication overhead and the amount of false alarms.

To detect latency shifts, the current model \(M_{i+1}(\alpha_{i+1}, \beta_{i+1})\) is compared to the previous latency model \(M_i(\alpha_i, \beta_i)\) (fig. 1) using the symmetric Kullback-Leibler (KL) divergence \(D\) as a metric, where \(KL(M_i||M_j)\) is the divergence metric for Gamma distributions [5]:

\[
D(M_{i+1}||M_i) = KL(M_{i+1}||M_i) + KL(M_i||M_{i+1}) > \eta,
\]  

\[
KL(M_i||M_j) = \psi(\beta_i)(\beta_i - \beta_j) - \beta_i + \log \frac{\Gamma(\beta_j)}{\Gamma(\beta_i)} + \beta_j \log \frac{\alpha_j}{\alpha_i} + \frac{\alpha_i \beta_i}{\alpha_j}.
\]  

Changes in the observed latency on the link are detected when the \(D(M_1M_2)\) is higher than a certain threshold. Examples of the circular learning scheme of overlapping models and detection of latency shifts are shown in figure 2. Here, a stepwise latency shift is temporarily induced on a connection in a scale-free network, by multiplying the simulated probe response delays by five. This corresponds to multiplying the scale parameter while maintaining the estimated value of the shape parameter (fig. 2). To reduce the effect of the somewhat bursty alarms caused by variations in the KL-divergence metric, the algorithm is not allowed to report the latency shift more than once until reaching a lower threshold of the KL-metric (see Appendix A).
3 Experiments and results

We have investigated both algorithm performance and adaptability with respect to varying network conditions. For this purpose we have performed the experiments using the OMNET++ simulation environment [8]. Further, we have tested the scalability of the algorithm. The algorithm performance was tested using parameters $\psi = 10^{-6}$ while varying the mean two-way packet drop rate between $\xi = \{0.025, \ldots, 0.5\}$ (drawn from a Gaussian distribution with $\mu = \xi$ and $\sigma = 0.2\xi$) and the rate of anomalous events $\lambda = \{10, 20, 40, 60, 80\}$. In all the experiments we assumed that in each period of 8 hours the expected number of $\lambda$ events was generated from a Poisson distribution on uniformly selected network elements. The type of event (i.e. latency shift or communication fault) was randomly decided. The latency changes were simulated as temporary stepwise shifts based on simulated probe response delays multiplied by a random number between 1 to 10. Further, the fault duration was randomly set up to 1 hour. Simulated link latencies (in milliseconds) were set symmetrically based on ran-
domly drawn parameter values from a Gaussian distribution, with $\mu = 2.5^{-3}$, $\sigma = 5^{-4}$ for the scale parameter $\alpha$ and $\mu = 30$, $\sigma = 6$ for the shape parameter $\beta$. The interval $\theta$ between individual probes was set to $\theta = 1.0 f_{cdf}^{-1}(0.9)$. The interval between probe tests were set $\tau = 10^5 f_{cdf}^{-1}(0.9)$. The first two experiments were performed on a synthetically generated scale-free network. The synthetic network consists of 30 nodes and 81 undirected links, and was generated using the Barabási-Albert method [6]. The scalability tests were performed on synthetically generated scale-free networks of increasing size from 40 to 300 nodes. All networks were generated starting with 5 nodes and adding 3 links in each iteration. To obtain networks with small amounts of single connections, 10% of the links were randomly removed in each case. For statistical significance, all results are based on 4 days of simulated time and shown as the mean of 10 runs.

4 Results

The metrics used for measuring the performance are based on the localisation rate for links and nodes, detection rate, false positives rate and the probe rate.

![Graph](a) (b)

Fig. 3: Performance rates obtained when increasing the packet drop rate for a) detection of communication faults and b) detection of latency shifts.

The results from the first two experiments generally show fixed detection rates for both communication faults and latency shifts when the drop rate is increased (fig. 3a, b). Further, we see that the detection rate for communication faults is fairly fixed (fig. 4a), whereas in the case of latency shifts the detection rate decreases with the number of anomalous events (fig. 4b). Moreover, we see that the localisation rates of communication faults on both links and nodes are over 70% for small amounts of packet drop and anomalous events (fig. 3a, 4a). We observe that the localisation rates of latency shifts for nodes and links are over
80% for increasing drop rates (fig. 3b), whereas it decreases with the number of anomalous events (fig. 4b), as a result of overlapping communication faults and latency shifts. In addition, we see that the probing rate is autonomously adapted to the increasing drop rate as described in section 2.3 (fig. 3a). In combination with a low setting of the detection threshold $\psi$, the number of false alarms can be kept small. Finally, due to the distributed nature of the approach, it should

Fig. 4: Performance rates obtained when increasing the number of anomalous events for a) detection of communication faults and b) detection of latency shifts.

scale well to the network size in terms of generated traffic. In figure 5, we see that the number of packets (including both control messages and probe traffic) indeed scales linearly with the number of connections.

Fig. 5: Scalability tests.
5 Concluding remarks

We have extended our approach to distributed anomaly detection to take into account deviations in observed probe response delays. This is achieved by using overlapping statistical models and comparing these models in between. This learning mechanism includes temporally palimpsest properties, and allows for smooth adaptation to long-term changes, while gradually forgetting earlier observations. Initial performance tests indicate that link and node anomalies caused by either shifts in expected latency or communication faults can be detected and localized with fairly high accuracy. Further, we have observed that the algorithm scales well to the network size in terms of communication overhead.

Future work includes refinement of the current model and further algorithm performance tests. The detection of latency shifts generates a burst of alarms until the learning model has converged to the new latency regime. The algorithm currently use simple thresholds to prevent sending more than one alarm per detected latency shift. For improved reliability, this alarm-mechanism could possibly be improved using e.g. Poisson distributions for individual network elements. In addition we believe that such an approach also can be used to detect abnormal behaviour for small populations of links and nodes in local regions of the network.

References

A Algorithm pseudo-code description

Let \( n \) be a node in the network. Each node needs to keep track of the set of neighbouring nodes, \( N_n \), as well as the sets of neighbours to each of these neighbours \( i, N^i_n \). Let each node \( n \) store an error state \( S^i_n \) related to communication faults for each neighbour \( i \). Each \( S^i_n \) represents the current state of \( n_i \) as viewed from \( n \), and can be assigned one of four different error states, namely no fault, link fault, node fault, link or node fault (a fault was detected, but the origin is undecidable). Further, let each neighbor store a similar array of states related to detection of latency shifts \( L^i_n \), taking the values no shift and link latency shift. In case latency shifts on all connections of a node have been detected the node reports itself as node latency shift.

\begin{algorithm}
\caption{Confirm fault of \( \hat{n} \) for \( n \) in \( \hat{n} \)}
\begin{algorithmic}
\Require \( \hat{n} \in N_{\hat{n}}, n \in N^0_{\hat{n}} \)
\State \( t \leftarrow \text{Test node } \hat{n} \text{ from } \hat{n} \)
\State Report \( t \) to \( n \)
\end{algorithmic}
\end{algorithm}

\begin{algorithm}
\caption{Test node \( \hat{n} \) from \( n \)}
\begin{algorithmic}
\Require \( \hat{n} \in N_n \)
\Repeat
\State Send test transaction to \( \hat{n} \)
\State Wait \( \theta \) s
\Until \( \text{Any response or } \prod_i (1 - P(R^{(i)}_{\Delta t})) < \psi \)
\If \( \text{Any response} \)
\State \text{return Success}
\Else
\State \text{return Fault}
\EndIf
\end{algorithmic}
\end{algorithm}
Algorithm 3 Monitor node $\hat{n}$ from node $n$

Require: $\hat{n} \in N_n$

repeat
    if Test node $\hat{n}$ from $n$ fails then
        if $S_{\hat{n}}^n = \text{No fault}$ then
            for all $\tilde{n} \in N^n_{\hat{n}}$ do
                Confirm fault of $\hat{n}$ for $n$ in $\tilde{n}$
            end for
            if Any $\tilde{n} \in N^n_{\hat{n}}$ report success then
                $S_{\hat{n}}^n \leftarrow \text{Link fault}$
                Report failed link from $n$ to $\hat{n}$
            else if All $\tilde{n} \in N^n_{\hat{n}}$ report fault then
                $S_{\hat{n}}^n \leftarrow \text{Node fault}$
                for all $\tilde{n} \in N_n$ do
                    $S_{\tilde{n}}^{\hat{n}} \leftarrow \text{Node fault}$
                end for
                Report failed node $\hat{n}$
            else
                $S_{\hat{n}}^n \leftarrow \text{Link or node fault}$
                Report link or node fault $\hat{n}$
            end if
        end if
        else if $S_{\hat{n}}^n \neq \text{No fault}$ then
            $S_{\hat{n}}^n \leftarrow \text{No fault}$
            if $S_{\hat{n}}^n = \text{Node fault}$ then
                for all $\tilde{n} \in N_n$ do
                    $S_{\tilde{n}}^{\hat{n}} \leftarrow \text{No fault}$
                end for
                Report working link from $n$ to $\hat{n}$ and node $\hat{n}$
            end if
            if $D(M_i+1, M_j) > \eta_{\text{upper}}$ and $L_{\hat{n}} = \text{No shift}$ then
                $L_{\hat{n}} \leftarrow \text{Link latency shift}$
                Report latency deviation on link from $n$ to $\hat{n}$ and notify node $\hat{n}$
            end if
            if All links from $n$ to $\hat{n}$ have been reported as shifting then
                Report Node latency shift in $n$
            end if
        else if $D(M_i+1, M_j) < \eta_{\text{lower}}$ and $L_{\hat{n}} \neq \text{No shift}$ then
            $L_{\hat{n}} \leftarrow \text{No shift}$
            Report normal link $n$ to $\hat{n}$
        end if
    end if
    Wait $\tau$ s
until $\hat{n}$ disconnects