

Using machine learning to discover service and host dependencies in networked systems

Aleksandr Simma, Moises Goldszmidt, John MacCormick, Paul Barham, Richard Black, Rebecca Isaacs, Richard Mortier*
Microsoft Research

Extended Abstract

In a modern enterprise network of any scale, dependencies between hosts, protocols and network services are surprisingly complex, typically undocumented, and rarely static. Even though network management and troubleshooting rely on this information, automated discovery and monitoring of these dependencies remains an unsolved problem. The approach we follow in the Constellation project [1] attempts to close this gap by pro-actively inferring a network-wide map of these complex relationships using a combination of machine learning techniques and statistical hypothesis testing. Constellation learns explicit local models of time dependencies between the different services on a given host using little more than the timings of packet transmission and reception. It then applies a recursive algorithm to combine local dependencies and build a graph where nodes represent servers/desktops and edges represent the corresponding service that cause the dependency between the nodes. In this abstract we focus on the probabilistic model that enables the learning of the local dependencies between the services at a given host (i.e., server or desktop computer).¹ The dependency discovery process comprises two steps. First we learn from the observed data a probability model of the interactions between inputs and outputs, and then we use statistical hypothesis testing to determine whether each dependence is statistically significant.

Let a channel be a grouping of communication packets (events for our purposes) belonging to the same service and peer (i.e., that can be logically grouped). Our generative probabilistic model considers a single output channel on a given host and simultaneously analyzes all of this host’s input channels to determine which channel(s) best explain the actual observed occurrence of the output packets. We use o_l to denote the time of an output event. Then the probability that packet k in input channel j generated output packet l is distributed as a Poisson process with intensity $p_k^{(j)}(o_l) =$

$w^{(j)} f_\theta(o_l - i_k^{(j)})$. The parameter $w^{(j)}$ represents the average number of output events that we expect each input event on channel j to be responsible for, and f_θ is the distribution of the delay between an input and the output events caused by it. This function will take as its argument the delay between the time of the output o_l and the time of the input $i_k^{(j)}$. Note that this makes intuitive sense: the probability that a given input packet caused a given output packet depends on both the expected number of packets it generates and the “distance” in time between them. The function $f_\theta(\Delta t)$ provides us with the opportunity of encoding prior information regarding the expected shape of the delay between the input and the output channels. To build the joint probability model of the output packets we recall that given a set of independent Poisson processes (denoted as PP) we can use the sum of their intensities and write $\{o_k\} \sim PP(\sum_j \sum_k p_k^{(j)}(\cdot))$ as the probability of the set of n outputs $\{o_k\}$, $1 \leq k \leq n$. Intuitively, given this independence between input channels, the model considers the packets in the output channel to be caused by the presence of any (a disjunction) of input packets in the input channels (with some uncertainty). We call this model Continuous Time NoisyOr (CT-NOR). Its formal relationship to the NoisyOr gate in graphical models is explored in [3].

We are now ready to write the likelihood function, namely, the probability of observing a set $\{o_l\}$ of outputs, given a set of inputs i on all the channels j . Let $\lambda = \sum_j \sum_k w^{(j)}$

$$P(o_1, \dots, o_n | i) = \lambda^n \cdot e^{-\lambda} \prod_{l, jk} \frac{w^{(j)} f_\theta(o_l - i_k^{(j)})}{\lambda} \quad (1)$$

The parameters $w^{(j)}$ and the ones in $f_\theta(\Delta t)$ are learned using EM [2] maximizing Eq. 1. The derivation of the EM algorithm for this task is described in [3]. In our experiments and characterizations we determined two particularly useful delay distributions ($f_\theta(\Delta t)$). The first is a linear mixture of a uniform distribution on a small window, and a decaying exponential. The uniform component captures almost instantaneous forwarding dependencies (order of a millisecond) while the exponential component captures more distant interactions resulting from queuing or processing. The second useful distribution was a similar mixture, but with a

*John is now with Dickinson College, PA. Work done while a Researcher with Microsoft Research. Aleks is with the University of California, Berkeley, CA. Work done while an intern with Microsoft Research.

¹*Dependence* is used here in the statistical sense to mean that by knowing the inputs we can better predict the outputs.

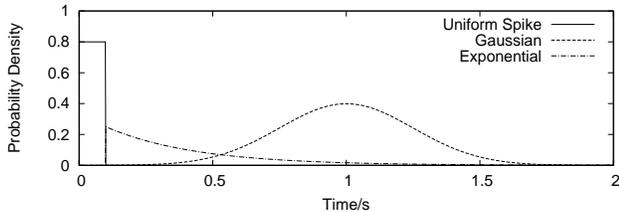


Figure 1: Example delay distribution functions $f_{\theta}(\Delta t)$

Gaussian component for the more “distant” interactions (see figure 1).

To determine which input channels are relevant to the output channel, given the model, we rely on the Likelihood Ratio Test (LRT) [4]. The null hypothesis H_0 is that for a specific $j = J$, that channel is irrelevant (i.e. input J is not responsible for any output packets). To test the hypothesis we maximize Eq. 1 twice, once with the full set of inputs, and once without channel J and compare the log-likelihoods. The result of this test is a “p-value” in the standard sense of statistical hypothesis testing.

We are currently testing the model and the discovery process using a trace of real data comprising headers and partial payloads of around 13 billion packets collected over a 3.5 week period in 2005 at Microsoft Research in Cambridge. This trace covers every packet sent or received by the 500+ nodes on site and captures conversations with over 28000 off-site IP addresses. Our initial results are extremely encouraging. We used forwarded HTTP traffic at a caching proxy server. We extracted ground-truth from a one hour period of the trace (10-11am on a Tuesday) by deep inspection of HTTP packets forwarded by the webproxy machine. The ground-truth data set formed by this HTTP traffic is an example of a particularly challenging environment. The presence of a large amount of multiplexing between traffic to and from the 95 clients and 9 upstream proxies makes relationships difficult to tease out at smaller timescales. Some of these connections transfer only a single HTTP object, whilst others carry many requests from multiple clients.

Figure 2 shows the precision-recall curve for CT-NOR when considering the full hour of the data set, as well as the first five minutes only. For comparison, we also include the results obtained when an heuristic is used checking whether co-occurrence of input and output packets within a window of 10 ms is used to decide correlation. Note that at the knees of these curves, CT-NOR has just over 94% recall and 96% precision, and co-occurrence has about 85% recall and 91% precision. This substantial difference in accuracy is amplified when performing large numbers of hypothesis tests and building the constellation graphs.

There are several avenues of future research we are pursuing. First, we are investigating the use of False Discovery Rate (FDR) techniques to control the number of false

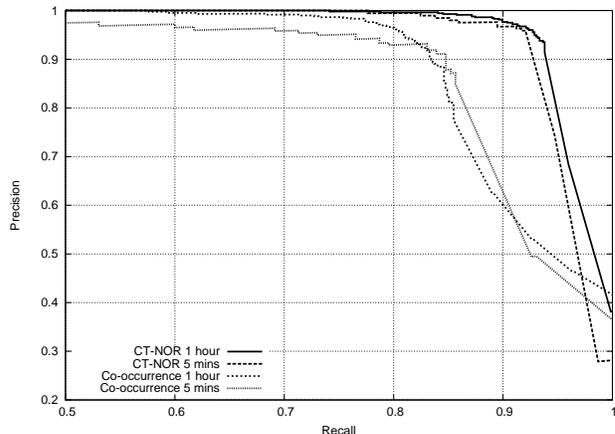


Figure 2: Ground-truth precision-recall for both CT-NOR and co-occurrence over the first 5 minutes of the hour and over the full hour. Note that for clarity the x-axis originates at 0.5 (at less than 50% recall we see almost 100% precision in all cases).

positives [4]. In this domain, ground-truth is very expensive (hence the need for our model). Although the p-value tells us something about the probability that a specific case will be falsely rejected, when we construct a constellation we perform a large number of hypothesis tests and the probability of making an error could rapidly mount up. Our initial results using the Benjamini-Hochberg procedure [4] look promising on selecting a p-value for a specific tolerance on false positives. The second avenue we are pursuing is that of change point detection for monitoring based on the parameter $w^{(j)}$. Finally, we are also looking at other domains where CT-NOR may be useful. The facts that (a) CT-NOR is based on generative models, and (b) it enables the incorporation of expert knowledge on the structure of the expected delay distribution through the function $f_{\theta}(\Delta t)$ renders these models particularly useful for dependency discovery in domains where it is imperative to model events in continuous time.

1. REFERENCES

- [1] P. Barham, R. Black, M. Goldszmidt, R. Isaacs, J. MacCormick, R. Mortier, and A. Simma. Constellation: end-user diagnosis for the enterprise. Technical report, Microsoft Research, 2008. Under submission.
- [2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [3] A. Simma, M. Goldszmidt, J. MacCormick, P. Barham, R. Black, R. Isaacs, and R. Mortier. CT-NOR: learning and reasoning about the dependencies between events in continuous time. Technical report, Microsoft Research, 2008. In preparation.
- [4] L. Wasserman. *All of Statistics*. Springer, 2004.