Capstone Proposal

Machine Learning Engineer Nanodegree

Andreas Lewitzki
January 4th, 2019

Proposal

Realtime contextual and passive network outage detection

Abstract

I will explore the possibilities of real time network outage detection through anomalies in the sparse IP address renewal (DHCP) signaling. Real time means trigger a detection event at occurrence. The sampling rate in this setup is by the minute, even though the individual events accounted for in the signal analyzed is very sparse. Contextual in this case refers both to the model, but also the data stream. In the data, by aggregating geographical zip code and logical network identifier. But also the method and model will be contextual as many decisions will be due to the, in a sense, narrow and specific use case.

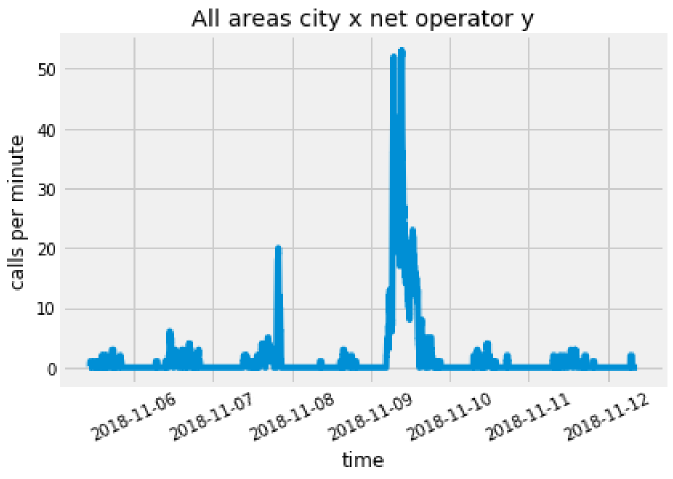
In our use case we want to apply this is on unmanaged open networks where the internet service provider has control over the DHCP. By applying this we could get a sense on local disturbances in open unmanaged networks where we normally don't have any insight or surveillance regarding aggregation or transport network outages. Outages in these cases might be caused by hardware/software failures or fiber cuts.
By generating events for supposed network outage we can more proactively and automatically prepare trouble tickets and trigger customer support call voice prompts linked to the specific autonomous network and geographic area. Having a clear prepared voice prompt regarding the supposed network outage, besides pure goodwill towards customers, will keep most people calling from these areas from having to speak to an human agent. As we're only using already existing data flow, this method will be totally passive towards the network and clients.

Domain Background

I've been working within telecom with network systems for many years. Our accesses today cover everything from cellular mobile systems to cable, DSL and Fiber/LAN. Services includes, besides access transport, multiple internet services.
Later years it has been common with open networks where the transport and network provision is handled by a network operator and the access and IP services is open for any service provider to subscribe. This is called open networks. Even in these open networks it is more and more common consensus to let the service provider handle the addressing of end user clients. In the case of dynamic leasing of IP addresses these clients will have a temporal renewal heart beats. By protocol design these fires within half the time of the lease time.[1] The most common default configured lease time is 3600 seconds. This means that if we get a renewal from a client minute 0 and issue a new lease the next request will come minute 30. So for a given client we can pretty safely assume its gone after 30 minutes of silence. When looking at many clients in discrete aggregates this creates a sparse but pretty predictable pattern.

Problem Statement

Within our own network operations we have access to management for alarm signaling and report disturbances. But for cases where we act service provider in some other external operator networks we are mostly blind. One big problem is that when the operator got outages we will not know about it until numerous customers call in to technical support. Also in these cases they won't get any heads up in the voice prompt as we are not aware of the outage. We can see this as an increasing number of customers call from the same areas within a short timeframe. I have an example of how that might look in the benchmark section. When this happens and we do not have any voice prompt in place customers will get through to human support agents that needs to walkthrough customers to identify the problem. That is in the long run very costly both economically and in customer trust.
Bellow is an example of unmanaged and uncaptured outage generating customer support calls. These are the kind of incidents I will try to mitigate by an anomaly detector.



Datasets and Inputs

This is the biggest and most time consuming task to manage but fortunately I've prepared and worked on this during last autumn. I have already set up a per minute export of DHCP renewal ticks by query to a slave of the in memory database. This data gets correlated to customer data by a service access id and gets aggregated and anonymized as operator and geographic zip code. Every row in the export is in JSON format containing a UNIX timestamp, network and operator identifier and a list of zip code and event counter pairs.

Example :

```
{"ts":1546493280,"netop":"OP CITY NETWORK","zipcnt":[[89493,2],[89540,6],[89491,4],[89430,1
],[89495,3],[89292,1]]}
```

This gets logged and rotated per day on a local Linux system. So it's easy set up a streamer process that can aggregate further.

The second counter I've added to the data stream contains the incoming calls and also gets aggregated by discrete minute, network and zip code. This is collected from a service we maintain that gets queried for every support customer call.

So after accounting for every minute and separate each network we can output simple csv rows containing timestamp, zip code, DHCP count and call count:

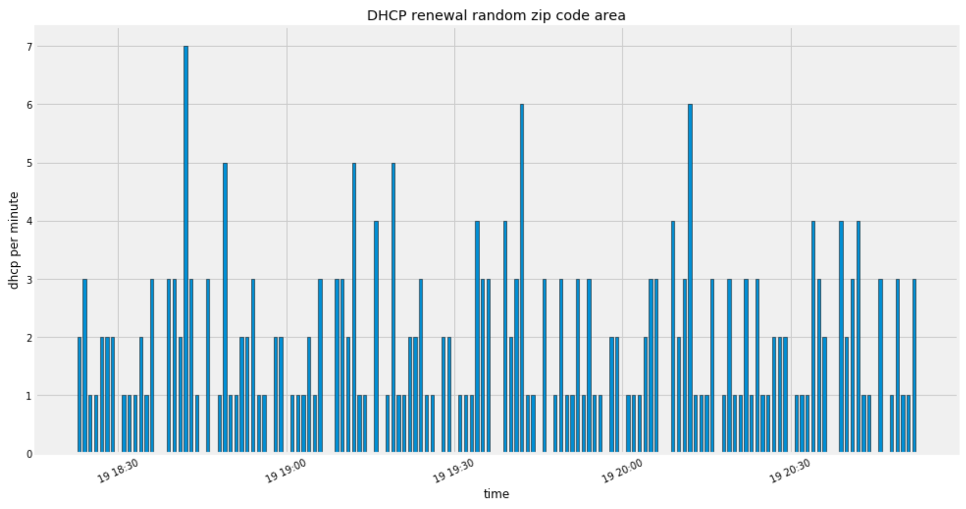
```
ts;zip;dhcp;call
1541155860;22291;0;0
1541155920;22291;3;0
1541155980;22291;5;1
1541156040;22291;3;0
1541156100;22291;2;0
```

The reason I like the zip code as an aggregation parameter is that it's agnostic and generic regarding access technique and service. Also it's a fair sized substitute for nodes in networks where we lack information about equipment normally used to aggregate the client ports.

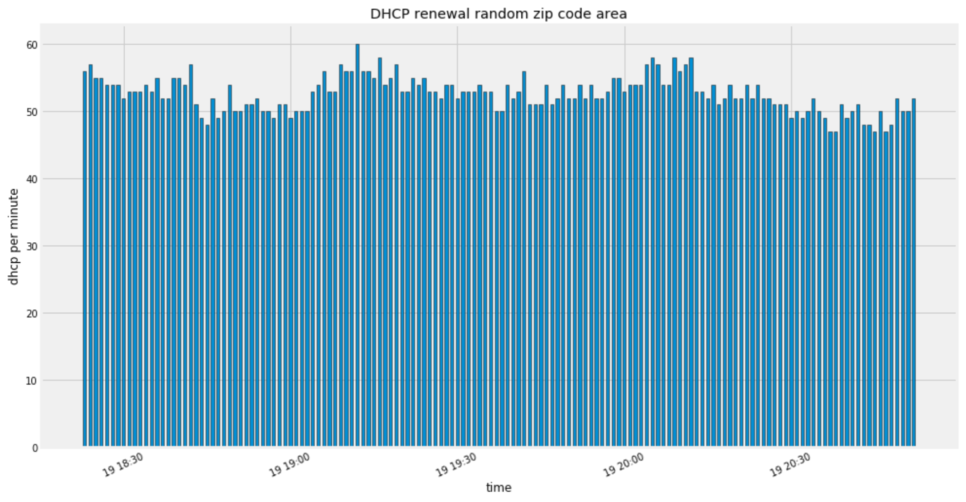
Solution Statement

My idea of solution is to build an anomaly detector that triggers secondary events to activate and clear automatic customer voice response for the specific area. By aggregating the counters and add pad with zeroed time ticks we should be able to get a sense of active leases. As the normal renewal time is 30 minutes, if we suddenly lose an area, we will see a drop minute by minute to a local minimum within 30 minutes. As we know the lease time and the protocol we can assume the device might be active the following 30 minutes. In the simplest naive approach and a perfect world we could predict the same count minute 30 as minute 0. Same count on minute 31 as on minute one etc. In one approach I will project the lease count on the following 30 minutes in the data processing stage. This is also called cumulative summary. By doing this we should go from a rather sparse pattern to a more stable stream. This difference is shown in the graphs bellow.

Original sparse counts :



And here with an added 30 minute cumulative sum window :



In both cases we need to apply a kind of convolution or sliding window to the data. Then use some regression to evaluate distances and create some error function for evaluation.
It might be tempting to go for a supervised learning approach as we could use the incoming calls as target.
The reason I am being somewhat skeptical about that is that the calls are even more sparse and most of them will not be explained by the DHCP signaling.
That's why I think a more simple anomaly detection mechanism might be easier to understand, explain, control and is more optimal in general here.
Depending on time and progress to complete this I might explore more complex models and even deep models as well.
One interesting approach would be to use convolutional neural network (CNN) that accounts for a bigger area where positions represents relations between zip codes. In this case the convolution could slide over the time axis as a window and zip codes gets represented by a 2d array.

Benchmark Model

First of we don't really have any model to compare against. In the case of open networks we go from having nothing to having something. What I would like to measure though is the amount of calls that, if the model where implemented the time of measure, would have gotten a voice prompt.
In this way we can at least compare how different models would perform given the same time frame and data.
What I will try to capture is events generating sudden spike in calls.

I will choose a period of 30days and see how much of these calls we could cover by predicting a gradient drop in DHCP renewals.

Evaluation Metrics

As this is a continuous data stream we also need to evaluate continuously. The only thing we could say is if we detect a gradient sudden drop in DHCP followed by increasing calls that’s a success.
How early we detect an event is more explained by the amount of clients in the area and the spread in DHCP requests.
But this is on a higher level as we also need to measure how good the actual model is in detecting the decrease. In this case we can evaluate the last value against a moving average for the prediction.
So a huge problem here is that we do not have clear true or false positives / negatives.
Even in the case where we have a drop in DHCP requests followed by increase in calls we can’t say that the dip explains all the calls.

What we will measure and evaluate anyway is the number of calls captured by our supposed anomaly window.
So if we evaluate a day where we had outage that showed in the DHCP requests we can always show a relative increase in the percentage of calls that would get a voice prompt compared to the total calls for the period measured.

Project Design

Even though my long term goal and aim is to set up a production system, the fully distributed system layout is outside the scope for this project. This project will analyze and explore the data and methods. Even though it might seem as anomaly detection in time series would be a simple task in these days of deep learning AI, it’s actually a lot of contextual domain specific considerations and explorations that needs to be done, both for methods as well as the evaluation of them.

The part of data collection and streaming has already been set up. I will extract a period of 30days. Ideally this period needs to have at least one disturbance big enough to validate against. This data will then be further anonymized and exported to my laptop for local jupyter notebook labs. My first approach will be to simulate the stream minute by minute for the sample period. These minutes will be recorded and measured in long term and short term frames. You can visualize this approach as a scattered sliding window due to the stochastic From there I will try different methods starting with some naive pattern matching regression by the mean. Then also try to stabilize the signal by a cumulative summary window. The results and findings will be discussed and showed in the final report.

References

[1] Dynamic Host Configuration Protocol (https://tools.ietf.org/html/rfc2131)