June 01, 2018

APPLICATION OF DNA-C: A METHOD FOR SHORT TERM LOAD PREDICTION AND LOAD BALANCING USING TECHNIQUES FROM CHAOS THEORY

Jerome Henry

Rajesh Pazhyannur

Robert Barton

APPLICATION OF DNA-C: A METHOD FOR SHORT TERM LOAD PREDICTION AND LOAD BALANCING USING TECHNIQUES FROM CHAOS THEORY

AUTHORS:
Jerome Henry
Rajesh Pazhyannur
Robert Barton

ABSTRACT

Novel techniques are provided for an application built on top of DNAC to predict network load on an access point (AP). These techniques determine a time interval after which each predicted dimension is not accurate, which is reflective of the real load. This allows the system to predict load accurately, without the burden of having to observe all neighboring access points, a computationally expensive process, while being able to determine when new weights and new predictions need to be generated. The result is a computationally-light yet accurate load prediction that can be applied to key Aps (e.g., lobby APs, or other APs), where the load is likely to change rapidly.

DETAILED DESCRIPTION

Load prediction is a key concern for any congestion point, such as an AP. When an AP becomes saturated, the load may need to be shared among several other APs. Although techniques exist to achieve load balancing, load prediction remains difficult.

A simple comparison with past load values is computationally costly, as such comparison needs to take into account a large set of input values. A basic threshold is also insufficient, because alternative APs may have a worse connection. The connection owner needs to balance the quality of connection with predictive capability. Additionally, load prediction is not cyclical. For example, load at 10 p.m. on weekdays is different from load at 10 a.m. on Saturday, and load on a Friday falling on or near a holiday (e.g., Thanksgiving or Black Friday) is different from non-holidays. The amount of data needed for analysis is

linearly proportional to the accuracy needed. Cases with flash-crowd situations are especially unpredictable in nature and difficult to predict.

Recently, machine learning algorithms have been leveraged to increase the capabilities of predictive systems because such systems take into account not only historic values (training data), but also may predict with reasonable accuracy short to mid-term loads by dynamically learning the weight of each contributing element in the training set. Another common technique to predict time series data uses seasonal Auto Regressive Integrated Moving Average (ARIMA). However, while these systems operate reasonably well for mid-term prediction (e.g., over a few hours or days), they commonly display a large error level for short term predictions (e.g., over the next 5 minutes or an hour), when the increment is not linear. For periods of stochastic change (flash crowds and sudden bursts or decreases in load) the model typically displays poor local performance unless it is run at the scale of the entire floor.

Short term Wi-Fi load is non-linear and multi-dimensional. The load is multi-dimensional because several variables, each with changing weights and therefore contributing influences, contribute to the future load. The load is non-linear because variables may display sudden changes (e.g., a client starting a video call or updating a mailbox). As a result, a simple neural network with weights learned from a training set only offer mild efficiency and improvement, because several weights may suddenly change at a fast pace (even if these changes cancel each other out over a longer period of time). To solve this issue, a dynamic learning neural network where weights are recomputed until reaching reconvergence may offer a more efficient solution. However, such reconvergence is computationally intensive (e.g., recomputing back propagation in an artificial neural net is very computationally intensive) with two consequences: (1) re-computation consumes resources that could be employed for real/active services; and (2) recomputation often only completes after the short-term change it was targeted to compute has actually occurred as convergence is slower than stochastic change.

When computation is performed at the scale of the entire floor, accuracy can be maintained at the scale of all APs, but at the cost of expensive computing. However, in most cases, only critical connection and congestion points (e.g., lobby APs) need to be managed for accurate load prediction over a short term. Thus, on one hand, computing at

the scale of the entire floor is accurate but computationally expensive, and on the other hand, computing at the scale of a few individual APs is computationally efficient, but inaccurate. As a result, setting DNAC to efficiently compute AP load is difficult.

To compute accurate local load prediction using machine learning (ML) techniques and also determine when the system needs to be retrained, the present techniques and systems evaluate the time interval during which the current ML-learned model is likely to be valid, and estimates or predicts a point in time (or a time interval) at which the prediction is expected to be too far from the real data, based on the real load value.

Prediction techniques for chaotic systems are well adapted to this type of problem. Chaotic systems are systems where: (1) changes are not linear and therefore are difficult to predict; (2) changes are not random and may depend on well-identified parameters that do not change without cause (For example, contributing events may either reinforce each other, compete with each other, or act as a predator-prey bifurcation diagram, where an increase in one event occurrence generates a decrease in another event, which may then in turn cause a decrease in the first event.); and (3) changes are very sensitive to initial/present conditions, and depending on the initial input values, a trend can spiral out of linearity (For example, the load level of 20% CU at 9 a.m. on Saturday morning in a store has a very different significance from a load level at 30% CU at 8 p.m. on Saturday in the same store. As the initial input changes, even very slightly, the deviation point can change to very different moments and the deviation can move towards very different directions due to attractors.).

As a consequence, chaos prediction tools (e.g., used for climate models, etc.) can accurately predict, based on the local present load input, at what point in time the predicted trend will lose accuracy. An example of this method is shown in *Figure 1* below. The present method leverages such techniques to provide a novel way of estimating, at small scale (one or a few APs), the span of time beyond which the load prediction stops being accurate. At that point, the system is retrained to predict the load for the next time segment.

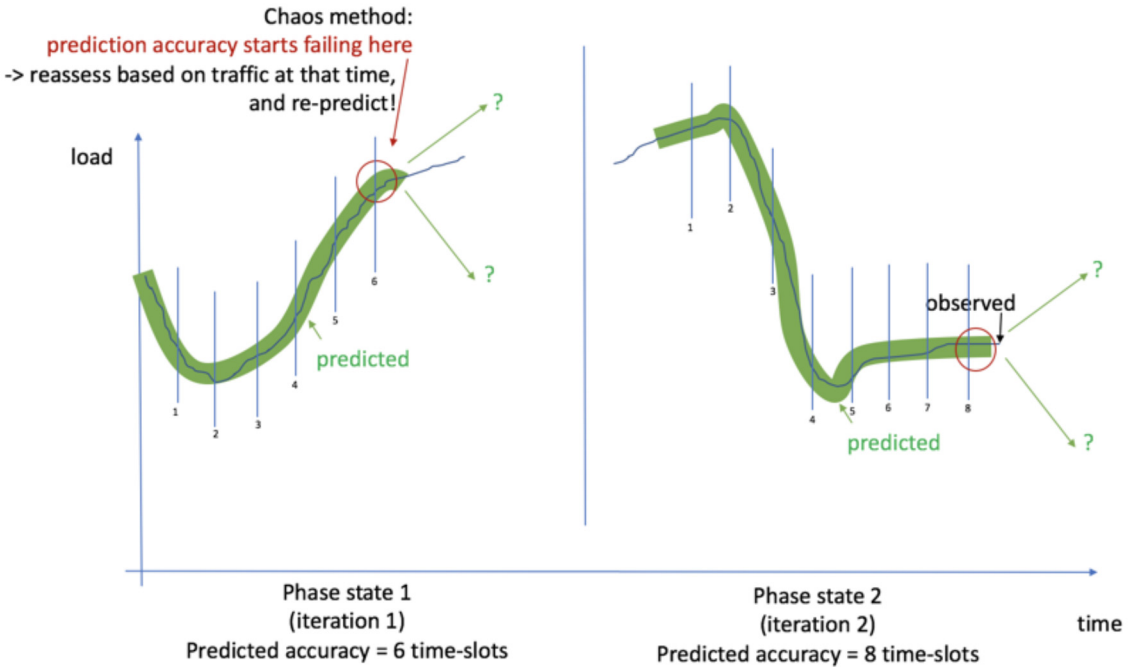

Figure 1

The method is explained in additional detail as follows. The initial inputs are the recent loads for each load prediction system. Based on this initial input, and for each dimension of the system, the delay time (*t*) may be computed (for example, using the Mutual Information method) to determine the number of time-slots after which the load prediction loses accuracy for that dimension.

Next, the embedding dimension *m* is computed (multiple methods are possible, for example, Cao Method, etc.). This allows the optimal duration of each time interval unit (or time-slots) to be evaluated. The Lyapunov exponent *(l)* may also be computed using the load data available above the latest time-slot interval to verify that the space is chaotic. This phase is critical. In "day time", unless the AP operates in a simulated environment, where changes may be linear, it is expected that the Lyapunov exponent will always be positive as system changes are chaotic/non-linear in nature. However, in the "dead of the night", e.g., when no activity is recorded, the system may become quasi-linear, which will allow the load prediction to be skipped, thus saving computational resources whenever possible. At the same time, the value of the Lyapunov exponent, projected at each time-slot computed previously, determines when (after how many *k* time-slots *t*) the system

stops predicting load accurately for each parameter, and when the previous prediction should no longer be used. When the previous prediction expires, a new set of data points should be used and a new computation should be performed.

A load time series $X(i)= [x1(i) ,x(i+t) ,...,x(I+(k-1)t)]$ may be computed, where m is the dimension (m position prediction) of the phase space, and i represents each computed possible position after each time slice t. Values of i may span from 1 to M (where M is the total number of projected positions, i.e., of reconstructed phase spaces). Variables t and m are then used to reconstruct the phase space (i.e., estimate the load within each time slot (t) and at k time-slots in the future).

The result is an estimate of the accuracy of the prediction (i.e. at how many t intervals the prediction would be accurate for each dimension/parameter, and when a new set of data points should be retrieved to perform a new computation. This result is expressed as inputs $w(1)$ to $w(k)$, where w is the estimated load for the next k time slot. This computation may be performed for each dimension parameter, and the last predicted value (index k) may be retained, which represents, for each dimension, the last value and point (in slot-times) before the system becomes chaotic and is no longer reliably predictable. These $w's$ may be used as weights.

The predicted weights may be injected into a neural network: m neural units are set for an initial layer. For example, each chaotic estimated value ($w(1)$ to $w(m)$) is injected as inputs into a neural network initial input layer. A number of units are positioned in one or more hidden layers (e.g., n units in one layer as shown in *Figure 2*), and the system has a single node output which represents the expected load of the overall system. (The number of intermediate layers is not relevant for this concept.) Then, the system may be trained with standard processes, for example, using a sigmoid function (or tanh, softmax or any other suitable neural net function).

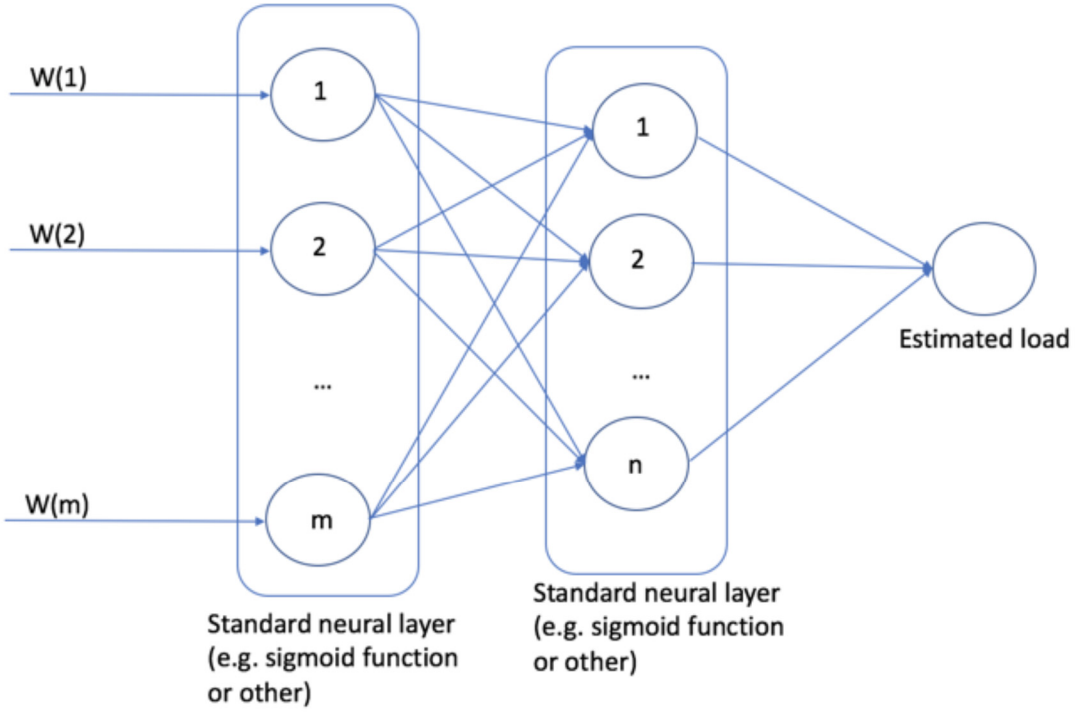

Figure 2

The output of the system is the estimated load after *t * (m-1)* time intervals. The novelty of this method is that (1) the load and (2) the times at which the load will likely stop being accurate may both be predicted. This estimation can be used to determine if the AP is likely to continue to increase its load, and for how long the prediction may be made with accuracy. As a result, the system can determine if an automated load-balancing response should be initiated, or if keeping present connections will likely offer a better quality of experience.

The efficiency of the chaotic input is that *m* will depend on the stochasticity of the expected change. As such, the system will not attempt to compute a load beyond a small number of *t* intervals at times when the stochasticity increases, and will attempt to compute a longer interval when stochasticity reduces. In contrast, standard neural networks with statically computed weights will always compute the same prediction interval, until weights are recomputed.

In summary, chaotic inputs provide a mechanism to dynamically predict when weights will become obsolete. While traditional techniques adapt to the cumulative change

resulting from variation in the initial conditions of each unit after full gradient descent and re-computation, the present techniques due at least in part to its chaotic structure, can predict sudden bursts that would be hidden by standard neural networks using passive weights assigned to each unit. Thus, the present techniques and systems solve the short-term accuracy prediction issue by implementing a novel approach to short term AP load prediction that adds chaotic algorithms to a neural network. These techniques allow for accurate load prediction at the scale of a single AP, thus reducing the computational load needed to anticipate a high capacity event and start load balancing.