Improving Transfer Time Prediction of ML Models via Auto-correcting Dynamical Systems Modeling

Venkat Sai Suman Lamba Karanam and Byrav Ramamurthy
School of Computing
University of Nebraska-Lincoln
Lincoln, NE, USA
saisuman@huskers.unl.edu and ramamurthy@unl.edu

2024 IEEE 10th International Conference on Network Softwarization (NetSoft) | 979-8-3503-6958-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/NetSoft60951.2024.10588950

***Abstract*—Machine Learning (ML) is extensively used for predicting transfer times for general purpose Wide Area Networks (WANs) or public Internet applications, but for Research and Education Networks (RENs) two major gaps exist in literature. First, RENs i.e. networks carrying large data flows have received limited attention by the networking community. RENs behave differently compared to the general purpose Internet applications and other network types. Hence, ML models from other network types cannot be used interchangeably for large data transfers. Second, the ML models are used as blackboxes to train on measured network values and then used to predict transfer times or other runtime network parameters. In this paper, we present a dynamical systems model of the large data transfers typical of RENs in the form of a system of Ordinary Differential Equations (ODEs) inspired by the Lotka-Volterra competition model. We present a transfer time prediction component called *Dynamic Transfer Time Predictor* (*DTTP*) which solves the ODEs and predicts the future transfer times. Second we formulate a loss function based on Lyapunov function called *Lyapunov Drift Correction* (*LDC*) that self-corrects the transfer time prediction errors dynamically.**

To design and develop our model, we studied real-world datasets consisting of over *100 million* transfer records collected from platforms such as Open Science Grid (OSG), Large Hadron Collider Optical Private Network (LHCOPN), Worldwide LHC Grid (WLCG), as well as the RENs of Internet2 and ESNet. We integrate our model into well-known neural network models and regressors and present evaluation results.

I. Introduction

Network transfer time prediction systems for general purpose Internet applications or public WANs are well researched. Many of the prediction systems use ML models as blackboxes that learn from the traffic traces/behavior and then make predictions. Little attention is given to the transfer time prediction systems for Research and Education Networks (RENs) that carry large data flows. Example RENs in the continental US include ESNet (Energy Sciences Network) [1] and Internet2 [2]. Traffic on RENs–typically characterized by large data flows–behaves differently from general purpose WANs and hence existing ML models developed for latter cannot be used on the former. We theorize that an accurate understanding of the REN network traffic and mathematical modeling is needed before ML models can be used efficiently. To design and validate our appraoch, we extensively collected and analyzed datasets from the Open Science Grid (OSG) [3], the Internet2 backbone [2], the Worldwide LHC Grid (WLCG) [4] and the ESNet [1].

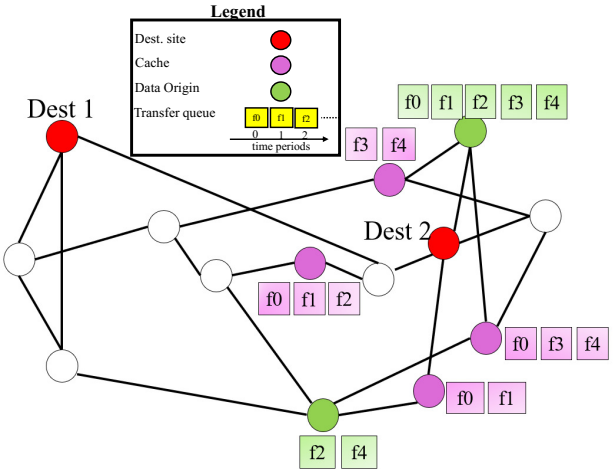

Figure 1. An example scenario of a data-intensive flow (or job) in the network topology shown above. The data transfer queue consists of data chunks $\{f_0, f_1 ...\}$ to be processed in order at sites **Dest 1** and **Dest 2**. There are multiple network paths, and source sites that exhibit different transfer times for each of the chunks. REN data transfer problem can be intuitively modeled as a competition-based dynamical system in the form of ODEs. This motivated us to model the problem using the Lotka-Volterra system.

*a) **Our Approach**:* Our work uniquely tackles transfer time prediction in RENs from a modeling perspective. We model the stochastic processes impacting observed transfer times as dynamical systems using ordinary differential equations (ODEs). We chose Lotka-Volterra system model as it is suitable to model the competing data transfers in RENs. We utilize solutions to the equations for predicting transfer times of a data chunk from its source sites. The site with the best transfer time is chosen as the source. Our approach involves a prediction error minimization mechanism, calculating system instability at each time step and minimizing errors accordingly. Our proposed transfer time prediction model consists of two components: (1) dynamical system representation which is solved and the result being the predicted transfer times (called *Dynamic Transfer Time Predictor* or *DTTP*), (2) correction component (called *Lyapunov Drift Correction* or *LDC*). Details of each are described in Sec. II.

The rest of the paper is organized as follows. Section II

Authorized licensed use limited to: University of Nebraska - Lincoln. Downloaded on December 19,2024 at 09:53:05 UTC from IEEE Xplore. Restrictions apply.

presents the technical details of the components in our prediction system. Section III presents our evaluation setup and Section IV presents the results and the associated discussion. Section V concludes our work and discusses our future directions.

II. Approach

This section covers the technical design of (1) the system modeling and prediction and (2) the correction components. We first begin by understanding how we modeled data-intensive flows as a dynamical system, using one of the well-known competition models called the Lotka-Volterra system [5].

A. Dynamical System Modeling of Large Data Transfers

The Lotka-Volterra model (L-V model), also known as the predator-prey model [5], is widely utilized to model dynamic predator-prey relationships and their influence on population growth. The general form of L-V model equation is expressed as a system of integro-differential equations of the form below.

$$\frac{dN_r}{dt} = \left(\epsilon_r + \sum_{s=1}^{n}\left(A_{sr}N_s(t) + \int_{-\infty}^{t} F_{sr}(t-\tau)N_s(\tau)d\tau\right)\right)N_r(t) \quad (1)$$

Here, $r = 1, ..., n$ and $s \neq r$. n is the total number of species in the environment. We transform the general L-V model from Eq. 1 to the REN scenario by integrating insights from the experiment literature. The L-V model's key principle, independence among entities, aligns with the independent nature of the experiments. Despite this independence, these experiments share finite resources, leading to high levels of competition. Consider two data-intensive flows, D and E,

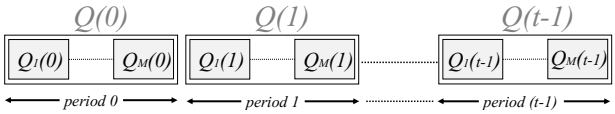

Figure 2. The figure shows the illustration of the dynamical systems representation of our problem statement- transfer time prediction for automating transfers. For simplicity, assume there is only one site r that wants to initiate a data transfer ρ at time period x. Assume there are M possible sites where the data chunk is located. $Q_i(x)$ is the expected data transfer time. $Q = \{Q_i | \forall i \epsilon \{0, 1, 2...M\}$ *and* $i \neq s\}$ is the queue of transfer times for data chunk ρ for each of the M sites. In the figure, $Q(x)$–referred to as a *state*–is a queue of expected transfer times for M sites for site s for time period x, where $x = \{0, 1, 2...(t-1)\}$ are time periods. Essentially this dynamical system formulation of transfers is essential to our problem statement- to automate transfers by predicting transfer times for each data chunk and for each time period. Each of the $Q_i(x)$ are formulated as an ODE and are solved by the prediction component *DTTP* (see Sec. II-B).

exhibiting the same internal resource usage pattern, both utilizing the shared grid resources in a distributed manner for transfers. Let R be a unit of resource that allows D and E to perform their transfer c and produce a unit output o. Both D and E try to maximize their utility function U when transferring the data on the grid. We give the general form of the utility function U^k where $k \epsilon D, E$ as $U^k = \sum_{t=0}^{\infty}\left(\ln o_t^k * (1 - \ln(x_t^k))\right)$. Here o_t^k represents the unit output during time period t and $(1 - \ln(x_t^k))$ represents the minimization of x_t^k, a value that is inversely proportional to the transfer rate.

To express data transfers using the L-V model, we examine their evolution in a discrete time formulation. We explicitly define this evolution as a dynamical system with a queue Q for each discrete time interval in the range $1....t$. Figure 2 illustrates the resulting dynamical system representation. In the following subsection, we adapt the L-V competition model to our problem statement using the dynamical system representation shown in Fig. 2. Following that, we define and explain the prediction model DTTP.

B. Dynamic Transfer Time Predictor (DTTP)

At each time slot t, $Q_i(t)$ is computed using the competition model in Eq. 2 for i^{th} site $i = \{1, 2, ..., M\}$. This means that $\frac{dN_r}{dt}$ computed from Eq. 2 becomes $Q_i(t)$. $Q_i(t)$ values depend on the history $\{Q_i(0), Q_i(1), ..., Q_i(t-1)\}$. From here on, we refer to the result of Eq. 2 as $Q(t)$ instead of $\frac{dN_r}{dt}$. We define the data transfers as a vector $f = \{f_t | t = 0, 1, 2, ...\}$, where f_t is the file or data to be transferred in time period t. In Eq.1, the coefficient of auto-correlation or the intrinsic rate of increase denoted by ϵ is a positive number which represents the rate of growth of the population. Often ϵ is generally modeled as a geometric i.e. exponential growth.For eq. 1, this can be given by $\frac{dN_r}{dt} = N_r(t_0)^{\epsilon t}$, where $N_r(t_0)$ is the initial population of species r. We define ϵ_r at time t as the rate of change in the transfer rate (i.e., N_r) and is proportional to the observed network throughput for that transfer during the time period t. ϵ_r has the limits $[a, b]$, where a and b are two positive integers. The constant a is constrained by the minimum network throughput available to r, ensuring the transfer occurs at the slowest speed. r experiences the highest transfer time at time t when a is the observed network throughput. The constant b is constrained by the maximum network throughput available to r: b represents the optimal scenario for r and yields the lowest transfer time at time t. Algorithm 1 describes the prediction component of our system, *DTTP*, which essentially is a procedure to compute the values of the network state at each time period t as $Q(t) = \{Q_1(t), Q_2(t), ..., Q_(M)\}$. Below we describe important definitions and process of computing $Q(t)$ using *DTTP*.

First, we define n, a positive integer, as the number of transfers, both, opportunistic and non-opportunistic transfers that are running on the sites that host the data, cached or uncached (origin). These n transfers compete for resources at the sites. $N_s(t)$, where $s = 1...n$, is the aggregate information rate (throughput) of the grid as a proportion of the total information rate by the transfer s during the time period t. A_{sr} is the influence of transfer s on r and represents the unitary action of s on r during time period t. We note that this is a continuous process in reality and continues till the lifetime of experiment s or r. Transfer r sees snapshots of A_{sr} in a stochastic fashion at discrete time intervals t. Such discretiza-

Authorized licensed use limited to: University of Nebraska - Lincoln. Downloaded on December 19,2024 at 09:53:05 UTC from IEEE Xplore. Restrictions apply.

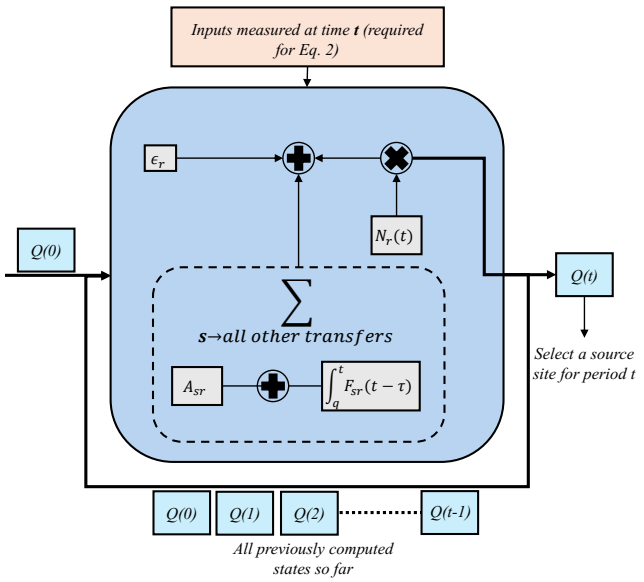

Figure 3. Visual representation of the prediction system *DTTP*, which predicts the state $Q(t)$ for time period t. The state $Q(t)$ is a vector consisting of expected transfer times for all potential sources sites $s\ in\ \{0, 1, 2, 3...M\}$. Say a transfer r is to be scheduled at time period t, the prediction system *DTTP* takes as input all the previous states until t i.e $Q(0), .., Q(t-1)$. The inputs required for Eq. 2 like ϵ_r, the unitary action A_{sr}, the seasonal component $\int_q^t F_{sr}(t-\tau)$ are updated for the current time period t. The output $Q(t)$ is fed back to the system again, to be used for computing the state for next period $(t+1)$.

tion is done to effectively model the evolution of a transfer as a discrete time formulation. For our prediction model, we define this as the proportion of the network bandwidth that s uses out of the set R_{sr}. R_{sr} is the aggregate network bandwidth that are common to s and r during time period t. A_{sr} can be given as the unitary action of transfer s on transfer r during time period t. This is expressed as $A_{sr} = P_{sr}(t) \times u_r(t)$. Here, $P_{sr}(t)$ is the probability of a unitary action by transfer s during time period t that impacts transfer r. For our model, we define this as the proportion of the network bandwidth that s uses out of the set ρ_{sr}. ρ_{sr} is the aggregate network bandwidth common to s and r during time period t. $u_r(t)$ is the utility of the transfer r during time period t. Following this, A_{sr} can now be given by $A_{sr} = \frac{\rho^s_{sr}}{\rho_{sr}}(t) \times u_r(t)$. ρ^s_{sr} is the network usage of s out of ρ_{sr}. ρ_{sr} are the aggregate network bandwidth common to s and r during time period t. By definition, $\frac{\rho^s_{sr}}{\rho_{sr}}(t) \leq 1$. Following the definition of $\frac{\rho^s_{sr}}{\rho_{sr}}(t)$, we could theorize that $0 \leq \frac{\rho^s_{sr}}{\rho_{sr}}(t) + \frac{\rho^r_{sr}}{\rho_{sr}}(t) \leq 1$. Also $u_r(t)$ is the utility function for time period t and is now given by $u_r(t) = \varsigma_r(t)$ where $\varsigma_r(t)$ is the number of concurrent transfer streams during the time period t for the transfer r. $\varsigma_r(t)$ is calculated as the number of I/O streams for transfer r among all the sources during time period t. The integral component $\int_q^{p=t} F_{sr}(t-\tau)N_s(\tau)d\tau * N_r(t)$ in the conventional L-V model represents the seasonal component. The term $F_{sr}(t-\tau)$ is the unitary action of transfer s on transfer r during the time period $(\tau, \tau + d\tau)$ which presents itself at period t. We model this component as any cumulative actions in the period $(\tau, \tau + d\tau)$ pertaining to transfer s that directly affect the runtime execution of transfer r. The lifetime of the transfer is limited and hence the lower limit of the integral is chosen to be the start time of the transfer, r i.e a finite value. Now we rewrite the seasonal component as $\int_q^t F_{sr}(t-\tau) = \frac{\rho^s_{sr}}{\rho_{sr}}(t-\tau) \times u_r(t-\tau) \times N_s(\tau)\Big|_q^t$. Using this redefined seasonal component and the redefined A_{sr} in Eq. 1 gives us the general form of our model.

$$\begin{aligned} Q_r(t) &= \frac{dN_r}{dt} \\ &= \Bigg(\epsilon_r + \sum_{s=1}^{n} \Bigg(\frac{\rho^s_{sr}}{\rho_{sr}}(t) \times u_r(t) \times N_s(t) + \\ &\quad \frac{\rho^s_{sr}}{\rho_{sr}}(t-\tau) \times u_r(t-\tau) N_s(\tau)\Big|_q^t\Bigg)\Bigg) \\ &\quad \times N_r(t) \end{aligned} \tag{2}$$

C. Lyapunov Drift Correction (LDC) for Prediction Error Minimization

To minimize the prediction model error (Eq. 2), we employ the Lyapunov drift-plus-penalty method, which diminishes the distance between predicted and actual trajectories of a dynamic system. The Lyapunov drift $\Delta(t)$ gauges instability at the start of time period t as a scalar, measuring the distance between two probability distributions at discrete intervals. This metric assesses system stability and detects changes over time. Higher the Δ value, higher the instability in the system. In the context of our prediction model in Eq. 2, Lyapunov drift Δ can be used to measure the distance between the predicted $Q(t)$ value and the actual observed value. Higher the Δ value, higher the prediction error of the cache model. The Lyapunov drift is given by $\Delta(t) = [L(t) - L(t-1)]$. $L(t)$ is the Lyapunov function–a scalar value–which defines the state of the dynamical systems at time t. When applied to our prediction model, Lyapunov function is calculated as half of the sum of squares of all predicted values by Eq. 2. Conventional definition of the Lyapunov function $L(t)$ is given by $L(t) = \frac{1}{2}\sum_{i=1}[Q(t)^2]$.

Let $\theta_i(t)$ be the cumulative debt in data transfer time incurred by site i until time t. $\theta_i(t)$ is calculated as $\theta_i(t) = q_i(t-1) - R_i(t-1)$. Here $q_i(t)$ is the sum of all predicted transfer times until time t for site i (using 2) and $\Theta_i(t)$ is the sum of all observed transfer times until time t for site i. $q_i(t)$ and $\Theta_i(t)$ are given by $q_i(t) = \sum_{p=1}^{t} Q(p)$ and $\Theta_i(t) = \sum_{p=1}^{t} \hat{Q}_i(p)$ respectively. $\hat{Q}(p)$ is the actual observed transfer rate for site i at time slot p where $p \leq t$. Below, we describe in detail its technical aspects and steps. Suppose we would like to pick the site from S, where $|S| = M$, that offers the minimum data transfer time. We modify the Lyapunov function by adding the debt term $\theta_i(t)$. The modified Lyapunov function $L(t)$ is given by the following equation.

$$L(t) = \frac{1}{2} * \sum_{i=1}^{M}[Q_i(t)^2 + V * r_i(t)^2] \tag{3}$$

Authorized licensed use limited to: University of Nebraska - Lincoln. Downloaded on December 19,2024 at 09:53:05 UTC from IEEE Xplore. Restrictions apply.

The term V is a weight parameter which adjusts the in-

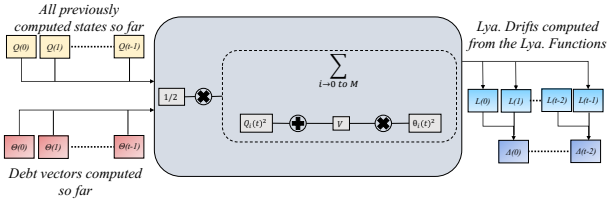

Figure 4. Block diagram of the dynamic drift correction mechanism (*LDC*), which adaptively corrects the prediction error of the transfer time predictions made by *DTTP*. At each time period t, *DC* takes as input (1) the state Q which is a vector consisting of expected transfer times for all potential sources sites $s\ in\ \{0, 1, 2, 3...M\}$ in the period $0, 1.., (t-1)$ and (2) the debt term Θ, which is a vector of cumulative debts for all transfers among the sites in $\{0, 1, 2, 3...M\}$. The LDC then computes the Lya. function $L(t-1)$ and the corresponding drift $\Delta(t-2)$. We must note that the Lya. function and the drift are computed until the *previous* time period (hence $L(t-1)$ and $\Delta(t-2)$), and then they are used to correct the prediction model $DTTP$ for the prediction into the future (i.e. time period t).

fluence of the debt term, denoted as $\theta_i(t)$, representing the cumulative debt in data transfer time for site i until time t. The constraint for minimizing transfer time is expressed as $\Delta(t) + V * R_i(t) \geq \Delta(t) + V * \theta_i(t)$. The constraint $\Delta(t)$ ensures observed transfer times do not exceed their predicted values (Eq. 2). We minimize the constraint $\Delta(t)$ for each time period t to ensure that $L(t)$ is as small as possible. This can be achieved by choosing the input parameters for the prediction model (Eq. 2). To correct and update the model, we chose the minimum of the right-hand side value among all the sites $i = 0, 1, ..., M$ i.e., $min(\Delta(t) + V * \theta_i(t))$. We assume that this minimum occurs at index k, then this index is passed to update the competition model. The site corresponding to the index k is chosen for the transfer during the next period t.

III. Evaluation Setup

This section outlines the setup we used to evaluate our proposed model. Figure 5 shows how we integrate our model into the ML framework/model. The ML model solves the prediction component (defined as a system of ODEs, see Eq. 2) and optimizes over the Lyapunov loss function (see Eq. 3). For evaluation, we chose well-known Neural network based models and regressors. The choice of the ML models was made to accommodate most commonly used architectures for transfer time prediction applications, aided by our experiences [6]. The details of the chosen ML models are given below.

A. Neural Networks

We chose three different Neural Networks (NN), namely, (i) a Deep Neural Network (DNN), (ii) Long Short-Term Memory (LSTM), (iii) and BiDirectional LSTM with Autoencoder-Decoder DNN. Each of the three NNs were modified to fit the requirements of our DTTP and LDC models, the details of which are left for brevity.

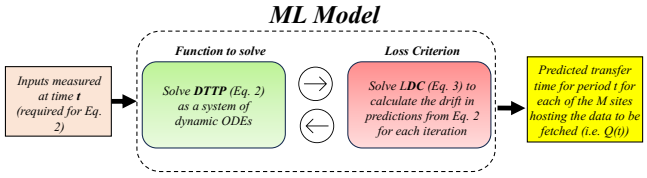

Figure 5. The architecture shows how our proposed prediction (*DTTP*) and correction (*LDC*) are integrated into the ML models.

B. Regressors

Regression-based models are often considered lightweight alternatives to the more computationally expensive neural network-based models. The five regression techniques are: (1) simple linear regression [7], (2) Lasso regression [8], (3) Ridge regression [9], (4) ElasticNet [10] and (5) Lasso-Lars [11] regressions.

IV. Results and Discussion

Model	MSE (%)	RMSE (%)	MAE (%)	MAPE (%)
BiDir_LSTM_Enc_Dec_DNN	0.72	0.85	0.43	0.89
LSTM	0.82	0.91	0.48	0.88
DNN	0.82	0.91	0.48	0.86
Linear	2.13	1.46	1.22	76.04
Lasso	0.72	0.85	0.43	1.27
Ridge	0.72	0.85	0.43	0.94
ElasticNet	0.72	0.85	0.43	1.27
LassoLars	0.72	0.85	0.43	1.27

Table I
Comparison of different ML models integrated with our approach (as shown in Fig. 5).

Table I presents the performance of the neural network-based models (Secs. IV-a, IV-b and IV-c) and the regressors (Sec. III-B). The hyperparameters for each of the models were chosen from the optimal values that were derived via *GridSearch*-based cross validation. The length of time period t was fixed to half-day and the number of days in the dataset was 700 consecutive days. Our results found that all the tested models are able to achieve relatively high accuracy in predicting the transfer times. Linear regression performed the worst, as evidenced by the large $MAPE(\%)$ value (Table I). However, when regularization is applied to the Linear regression, using Ridge/ElasticNet/Lasso-Lars, the performance of the linear regression improves remarkably. We did not find considerable difference in accuracy between any of the regularization methods. Similarly, all three neural network-based models achieve similar accuracy in predictions, with the BiDir_LSTM_Enc_Dec_DNN edging out slightly. Figure 6 shows the Mean Error (%) for each of the time periods t using DNN (top-left), LSTM (top-right), BiDir_LSTM_Enc_Dec_DNN (bottom-left) and Ridge regression model (bottom-left).

While all four models exhibit fluctuations in prediction mean errors, regression-based models, particularly ridge regression, tend to perform worse as the training period increases, evident from wider error lines in Fig. 6.

Authorized licensed use limited to: University of Nebraska - Lincoln. Downloaded on December 19,2024 at 09:53:05 UTC from IEEE Xplore. Restrictions apply.

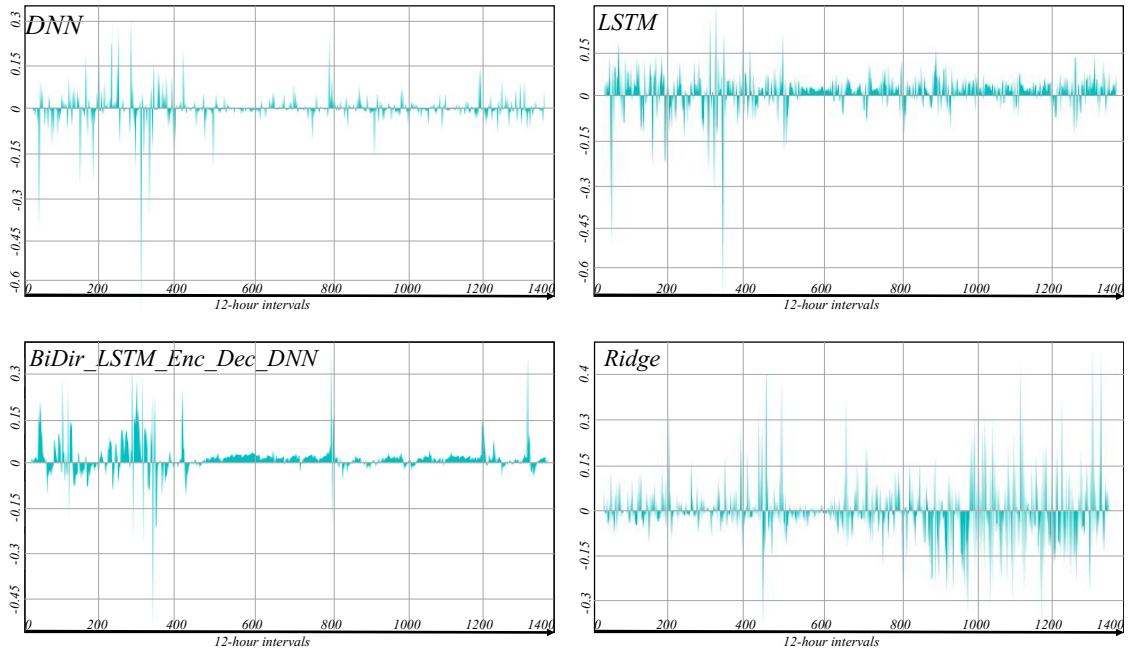

Figure 6. The figure shows the Mean Error (%) over 700 days for each of the three NN models and one regressor (Ridge) that were integrated with our DTTP+LDC model (see Sec. III-A & III-B and Fig. 5). The markers on X-axis start from 0 and end at 1400, representing the 1400 half-day time intervals. The Y-axis is the Mean Error (%) of the predictions made by (top-left to bottom-right): DNN, LSTM, BiDir_LSTM_Enc_Dec_DNN and Ridge regression models. Our approach has shown high, consistent accuracy in predicted transfer times for each interval (12-hour duration). The sporadic large spikes (both *above- and below- zero*) are outliers where our model was not able to predict accurately. We theorize that these sporadic inaccuracies have the following possibilities: (i) due to unknown underlying changes in the network configurations (like maintenance and/or abrupt disturbances of unknown causes), (ii) due to the gaps in the data logging systems leading to incorrect input data values used for training, (iii) occasional resource usage anomalies natural to the grid usage patterns (e.g. sudden availability of a specific dataset from an experiment, leading to an increase in frequency of experiments using that same dataset. This ultimately shows up as anomaly or outlier in the input dataset, which was used to train our model).

V. Conclusion and Future Work

This paper highlighted the importance of modeling the REN network behavior using mathematical representations before ML models can be applied. Specifically, we modeled the data transfers as a dynamical systems problem using a system of ordinary differential equations (ODEs) inspired by the Lotka-Volterra (LV) competition system. LV model was modified and adopted because its underlying theory can be adapted to the REN traffic behavior (validated by our preliminary analysis of a large collection REN traffic).

There are several interesting directions for future work. Our proposed model can be further improved with application layer awareness, where application/experiment type (LHC vs CMS) is used as an additional term in the *DTTP* formulation (Eq. 2). Additionally, other hyperparameters such as alternative prediction time period granularities can be explored. We plan to combine our approach with the SDN paradigm for the automation of data movement across the network, intelligent caching and rerouting.

VI. Acknowledgments

This work is sponsored in part by National Science Foundation (NSF) grant with award number OAC-2322369 and Department of Energy (DoE) grant with award number DE-SC0024648.

References

[1] J. Leighton, "ESNet: The Energy Sciences Network," 1996.
[2] F. Yeung, "Internet 2: Scaling Up the Backbone for R&D," *IEEE Internet Computing*, vol. 1, no. 2, pp. 36–37, 1997.
[3] R. Pordes, D. Petravick, B. Kramer, D. Olson, M. Livny, A. Roy, P. Avery, K. Blackburn, T. Wenaus, F. Würthwein *et al.*, "The Open Science Grid," in *Journal of Physics: Conference Series*, vol. 78, no. 1. IOP Publishing, 2007, p. 012057.
[4] I. Bird, "Computing for the Large Hadron Collider," *Annual Review of Nuclear and Particle Science*, vol. 61, pp. 99–118, 2011.
[5] M.-C. Anisiu, "Lotka, Volterra and their model," *Didáctica mathematica*, vol. 32, pp. 9–17, 2014.
[6] V. S. S. L. Karanam and B. Ramamurthy, "Dycrono: Dynamic cross-layer network orchestration and real-time deep learning-based network load prediction," in *2023 International Conference on Optical Network Design and Modeling (ONDM)*, 2023, pp. 1–6.
[7] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2003, vol. 330.
[8] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, 1996.
[9] A. E. Hoerl and R. W. Kennard, "Ridge regression: applications to nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 69–82, 1970.
[10] C. De Mol, E. De Vito, and L. Rosasco, "Elastic-net regularization in learning theory," *Journal of Complexity*, vol. 25, no. 2, pp. 201–230, 2009.
[11] M. G. Usai, M. E. Goddard, and B. J. Hayes, "LASSO with cross-validation for genomic selection," *Genetics research*, vol. 91, no. 6, pp. 427–436, 2009.

Authorized licensed use limited to: University of Nebraska - Lincoln. Downloaded on December 19,2024 at 09:53:05 UTC from IEEE Xplore. Restrictions apply.