Learning Mixed Traffic Signatures in Shared Networks

Hamidreza Anvari and Paul Lu

Dept. of Computing Science
University of Alberta
Edmonton, Alberta, Canada
{hanvari|paullu}@ualberta.ca

Abstract. On shared, wide-area networks (WAN), it can be difficult to characterise the current traffic. There can be different protocols in use, by multiple data streams, producing a mix of different traffic signatures. Furthermore, bottlenecks and protocols can change dynamically. Yet, if it were possible to determine the protocols (e.g., congestion control algorithms (CCAs)) or the applications in use by the background traffic, appropriate optimisations for the foreground traffic might be taken by operating systems, users, or administrators.
We extend previous work in predicting network protocols via signatures based on a time-series of round-trip times (RTT). Gathering RTTs is minimally intrusive and does not require administrative privilege. Although there have been successes in using machine learning (ML) to classify protocols, the use cases have been relatively simple or have focused on the foreground traffic. We show that both k-nearest-neighbour (K-NN) with dynamic time warp (DTW), and multi-layer perceptrons (MLP), can classify (with useful accuracy) background traffic signatures with a range of bottleneck bandwidths.

Keywords: Classification · TCP · protocol selection · wide-area networks · high-performance network · fairness · shared network

1 Introduction

Knowledge about the state of a data network can be used to achieve high performance. For example, knowledge about the protocols in use by the background traffic might influence which protocol to choose for a new foreground data transfer. Unfortunately, global knowledge can be difficult to obtain in a dynamic, distributed system like a wide-area network (WAN).

Previously, we introduced a machine-learning (ML) approach to network performance, called *optimization through protocol selection (OPS)* [2]. Using local round-trip time (RTT) time-series data, a classifier predicts the mix of protocols in current use by the background traffic. Then, a decision process selects the best protocol to use for the new foreground transfer, so as to maximize throughput while maintaining fairness. We showed that a protocol oracle would choose

TCP-BBR [7] for the new foreground traffic if TCP-BBR is already in use in the background, for proper throughput. Similarly, the protocol oracle would choose TCP-CUBIC [12] for the new foreground traffic if only TCP-CUBIC is in use in the background, for fairness.

However, it was unclear if that result would generalize further. For example, only one bottleneck bandwidth was considered in our first empirical evaluation. On a real network, the bottleneck bandwidth can change dynamically due to different network routing and paths.

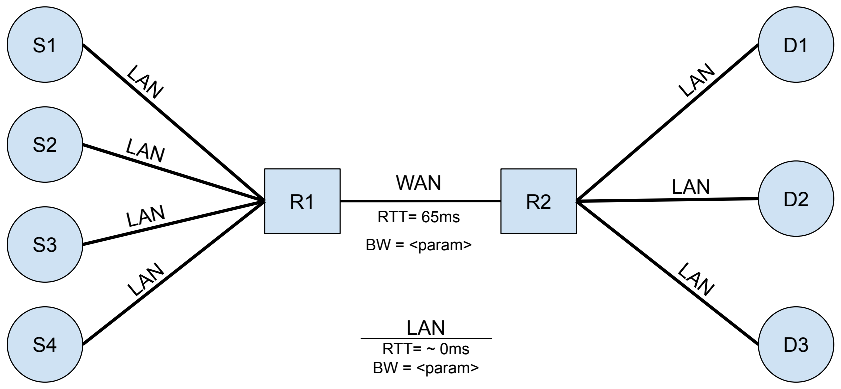
After gathering more empirical signatures with different bottleneck bandwidths (Fig. 1b), we show that the applicability of k-nearest-neighbour (K-NN) with dynamic time warping (DTW) remains intact, even with a mix of training data from different bottlenecks. That result is consistent with the ability of DTW to abstract out scaling differences in time-series data.

In our previous study [2], K-NN and MLP were comparable in accuracy. We advocated for the use of K-NN in practice due to the relative simplicity of K-NNs, and due to the explainability of K-NN's predictions and mis-predictions. In other words, if a K-NN makes an incorrect prediction, one can reason about why K-NN was wrong by examining the training data vectors that were the nearest neighbours of the query vector (a.k.a. time-series). In contrast, if an MLP makes an incorrect prediction, it is harder to explain the mistake. MLPs have a multitude of parameters and weights within the layers and perceptrons that make explanations more difficult.

This empirical study shows that even with multiple and/or changing bottleneck bandwidths (i.e., mixed traffic signatures), OPS continues to be possible with either K-NN or MLP. In the future, we plan to explore the use of ML, whether with K-NN, MLP, LSTM, or some other algorithm, to predict other properties of network traffic. So far, OPS has been about selecting an appropriate CCA at the operating system level. But, if the user had an ML oracle that correctly predicted other network characteristics, then the user might choose different application-level strategies (e.g., different video compression algorithm to make bandwidth-vs-quality trade-offs). Or, in the long term, network administrators might be able to identify situations in which the best optimisation is to selectively move some traffic to a different network route (e.g., an overlay network, or a private network).

2 Experimental Setup

We implemented a controlled testbed to allow us to vary different network configurations (Fig. 1). This controlled network extends the testbed from our previous work at 1 Gb/s speeds [3] by adding 10 Gb/s network interface cards (NICs) and switches, enabling us to vary between 1 Gb/s and 10 Gb/s for the native link rate, while keeping the possibility of emulating varying bottleneck bandwidth, *BtlBW*, and end-to-end latency, *RTT*.



(a) Dumbbell Network Topology (Based on our previous study [3])

Configuration	LAN		WAN	
	BW	RTT	BW	RTT
C1	1 Gb/s	∽ 0 ms	500 Mb/s	65 ms
C2	10 Gb/s	∽ 0 ms	250 Mb/s	65 ms
C3	10 Gb/s	∽ 0 ms	500 Mb/s	65 ms
C4	10 Gb/s	∽ 0 ms	1 Gb/s	65 ms
C5	10 Gb/s	∽ 0 ms	2 Gb/s	65 ms

(b) Testbed Configuration Scenarios

Node(s)	CPU (Model/Cores/Freq.)	RAM
$S1$, $D1$	AMD Opt. 6134 / 8 / 2.30	32 GB
$S2$, $D2$	Intel Ci3-6100U / 4 / 2.30	32 GB
$S3$, $D3$	Intel Ci3-6100U / 4 / 2.30	32 GB
$S4$	AMD E2-1800 / 2 / 1.70	8 GB
$R1$, $R2$	AMD A8-5545M / 4 / 1.7	8 GB

(c) Nodes Configuration

Fig. 1: Testbed Architecture

2.1 Logical View

Our controlled testbed consists of a dumbbell topology with 7 end-nodes grouped into 2 virtual LANs, and 2 virtual edge routers (i.e., `R1`, `R2`) connected over an emulated WAN of bandwidth BtlBW and RTT (Fig. 1a). In the literature, the dumbbell topology is often used as a simplified representation of a shared network.

For the WAN link, the Dummynet [6] network emulator is used on nodes `R1` and `R2` to control the desired bandwidth, delay, and router queuing properties. The emulated configurations we used in this study are summarised in Fig. 1b. The end-to-end propagation delay (base RTT) for all the scenarios is set to 65 milliseconds for the WAN connection, as found on many medium-to-large WANs; the RTT for LAN connections is negligible. The LAN link bandwidth is either 1 Gb/s or 10 Gb/s, both native link rates available to the nodes. The bottleneck shared bandwidth for WAN (BtlBW) varies from 250 Mb/s up to 2 Gb/s.

The router buffer size at R1 and R2 are fixed at 6 MB. This buffer size accounts for about 0.5 BDP for the highest BtlBW, and up to 2 BDP for the lowest BtlBW (BDP=Bottleneck-Bandwidth $\times$ Delay (RTT)). Setting the buffer size to less than 1 BDP, so-called shallow buffering, could result in the underutilisation of network bandwidth. In contrast, setting the buffer to large sizes could result in an effect called buffer-bloat where the users experience extremely long delays due to long queuing delays at the router. However, the existing version of the Dummynet software for Linux platform has an internal limitation that prevents us from setting the buffer to sizes larger than 6 MB. But this fixed size is sufficiently large to enable senders to saturate the BtlBW for WAN in all configurations.

2.2 Physical View

The testbed is implemented as an overlay on top of a physical network. All the nodes in our testbed are physically located in a dedicated cluster, all running Linux distribution CentOS 6.4 using kernel version `4.12.9-1.el6.elrepo.x86-64`. The hardware configuration of the nodes are provided in Fig. 1c. All the nodes are equipped with two network interface cards of 1 Gb/s and 10 Gb/s native rates. There are two network switches connecting all the nodes in parallel, at 1 Gb/s and 10 Gb/s rates accordingly. There is no interference from other traffic because the cluster is isolated from other networks.

2.3 Data-Transfer Scenarios

For all experimental scenarios (Fig. 1b), the sender nodes on the left-hand-side LAN (S1, S2, S3, and S4) act as traffic generators, sending bulk data over the WAN link. The receiver nodes are on the right-hand-side LAN (nodes D1, D2, and D3). We run two data-transfer tasks simultaneously, between S1-D1 and between S2-D2. During the transfers, regular network pings (to measure RTT) are conducted between S3-D3 (§ 5.1). S4 is not normally used for these experiments. Both streams and the pings must travel across the bottleneck link R1-R2. The RTT time series are presented in Fig. 5 and form the basis for our machine-learning experiments in § 5.

Of course, networks often carry multiple streams of traffic, but we start our evaluation with the simpler case of two streams, to make it easier to control interactions.

3 Shared Network: Does Background Traffic Matter?

Networks are not always private. Bandwidth-sharing networks are still the common case for a large number of research and industry users. The workload on shared networks tends to be highly dynamic. As a result, estimating the network condition and available resources (bandwidth, etc.) is one of the challenging tasks in bandwidth-sharing networks. There are a number of studies in this area, investigating the possibility of estimating the available bandwidth in high-speed networks [20, 25].

In addition to bandwidth estimation, estimating the network workload and the type of background traffic on the network could also affect the performance of data transfer tasks. In one study, the effect of background traffic on distributed systems has been investigated [23]. Also, in our previous work we have investigated and shown the counter-intuitive performance of some well-known tools and protocols depending on the type of background traffic in the network [1, 3, 2]. Hence, obtaining knowledge about the background traffic would allow better and more-efficient adjustments of the network configurations, and choosing appropriate data-transfer tools and protocols.

In this section, we further investigate the impact of background traffic on the performance of TCP CCAs, as well as on a few high-performance data transfer tools.

3.1 Case 1: TCP CCAs Interaction as Background Traffic

Here we briefly review the Transmission Control Protocol (TCP) in terms of its congestion control algorithm (CCA). We review CUBIC and BBR, two popular CCA schemes. CUBIC [12] is the the default TCP CCA deployed on most Linux-based hosts. BBR [7] is a newer CCA. Some studies show that both CUBIC and BBR manifest unfair bandwidth utilisation under various circumstances [18, 19].

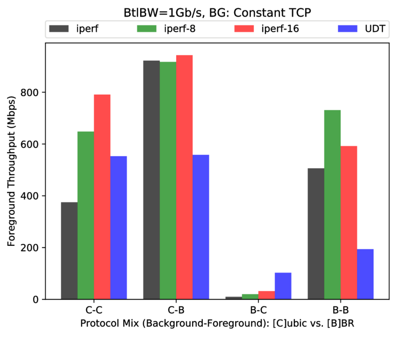
To further investigate the interoperability of CUBIC and BBR, as well as their impact on the other traffic sharing the same bandwidth, we have run an experiments, using CUBIC and BBR over a shared network. The results are provided in Fig. 2. Due to space limit we only present resluts for `C4` configuration.

For each of two possible background traffic, (1) Constant TCP (Fig. 2a), (2) Square-wave pattern TCP Stream (Fig. 2b), we run 4 separate data-transfer tasks as foreground traffic (different bars in Fig. 2):

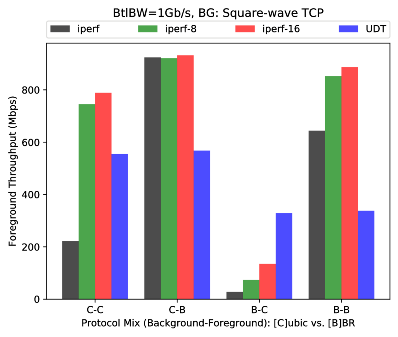
1. **iperf.** A single TCP stream using the standard iperf tool.
2. **iperf-8.** A combination of 8 parallel TCP streams for data transfer.
3. **iperf-16.** A combination of 16 parallel TCP streams for data transfer.
4. **UDT.** A well-known UDP-based high-performance data transfer tool.

For both background traffic and foreground data-transfer tasks we have examined both CUBIC and BBR to study their interaction and impact on each other. For example, the notation *C-C* in Fig. 2 specifies that CUBIC is the CCA for the background traffic, as well CUBIC is the CCA for the foreground traffic (whether it is iperf, iperf-8, iperf-16, but NOT UDT since UDT is based on UDP). Similarly, the notation *C-B* specifies that CUBIC is the CCA for the background traffic, but BBR is the CCA for the foreground traffic (iperf, iperf-8, iperf-16, but NOT UDT).

As shown in Fig. 2, for all the possible mixtures of parameter configurations, there is a significant variation in the observed throughput performance while



(a) Background Traffic: Constant TCP Stream



(b) Background Traffic: Square-wave pattern TCP Stream

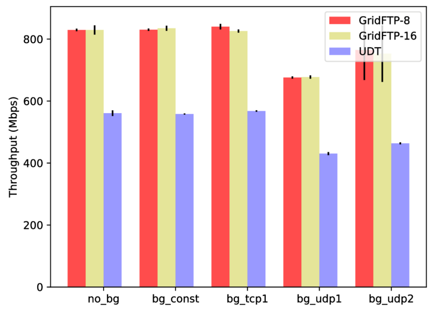
Fig. 2: Impact of different TCP CCAs as background traffic on each other

each tool runs along with another TCP CCA. Both CUBIC and BBR are able to utilize available bandwidth while running in isolation. However, BBR has a significant negative impact on CUBIC in all combinations, regardless of running as a background or foreground stream. CUBIC stream(s) suffer from extreme starvation when running along with BBR streams on a shared networks. In contrast, all-CUBIC and all-BBR scenarios, while not perfect, are considerably fairer in sharing bandwidth compared to heterogeneous combination of the two CCAs.

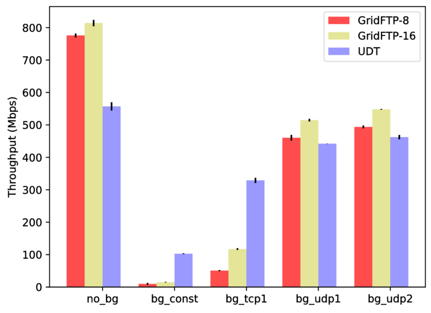
3.2 Case 2: TCP- vs. UDP-based Background Traffic

In last section we investigated the impact of single TCP stream of varying CCA algorithm on the foreground traffic. In this section, we further expand our observations for more complicated patterns of background traffic, where background traffic is either a TCP stream or a UPD-based data transfer task as follows: No background (`no_bg`), constant TCP (`bg_const`), TCP- and UDP-based square-wave pattern cycling on and off for 10 seconds (`bg_tcp1` and `bg_udp1`), and an UDP-based square-wave traffic generated by RBUDP [14] data transfer tool. The results are depicted in Fig. 3.

Similar to previous section, here we have studied the possible combinations of CUBIC and BBR where the data transfer tasks are TCP streams. The results here show that further than TCP CCA, the type of background traffic being TCP or UDP stream will differently impact the foreground data transfer performance. While for simple TCP-based background traffic the GridFTP tool seems to perform better than UDT, when UDP-based streams are added as the background traffic the UDT performs relatively better than GridFTP.



(a) BG=CUBIC, FG=BBR



(b) BG=BBR, FG=CUBIC

Fig. 3: Impact of various TCP- and UDP-based protocols on TCP CCAs. BtlBW=1 Gb/s is the bottleneck bandwidth of WAN link. BG and FG are the CCAs used for TCP-based background and foreground traffic respectively where applicable. (Listed BG CCA is irrelevant for UDP-based background cases.)

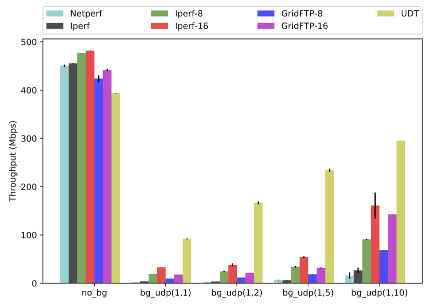
3.3 Case 3: Burstiness of the Background Traffic

In the last section we observed that TCP- and UDP-based background traffic could imply considerably different performance for the foreground data transfer tasks. In addition, we have already seen that for TCP-only mixture of traffic, different CCAs could have different impacts the performance of foreground traffic. In this section we will see that when the background traffic is only formed of UDP-based streams, there are still more attributes that could impact the performance of data transfer tasks. For this experiment, we conduct a bursty UDP stream as background traffic, represented by an square-wave pattern. The burstiness of the UDP stream is varying from 1-second ON and 1-second OFF (denoted by bg_udp(1,1)), to 1-second ON and 10-seconds OFF (denoted by bg_udp(1,10)). The results are provided in Fig. 4. Again, the burstiness of the background traffic could significantly impact the performance of data transfer tasks, where CUBIC is more vulnerable to this burstiness than BBR (Fig. 4a and Fig. 4b respectively)

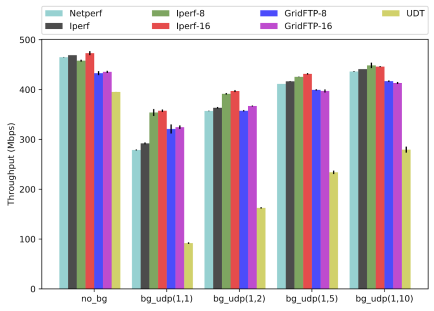
4 End-to-End Traffic Probing: Intrusive vs. Non-Intrusive

In § 3 we investigated and shown the impact of background traffic, from several aspects, on the performance of foreground traffic and data transfer tasks. With that knowledge, here we will review and discuss the possible network signals that might be used at end-nodes to probe the shared bandwidth and identify the type of background traffic. Such a knowledge would later be utilised to decide on an appropriate set of tools and protocols for transferring data over the network.

Estimating the type and mixture of network traffic is performed from a local, non-global perspective, viewing the network as a black box. In such scenarios



(a) Foreground Traffic: TCP CUBIC



(b) Foreground Traffic: TCP BBR

Fig. 4: Impact of bursty UDP-based background traffic on CUBIC and BBR foreground. Background traffic is generated using *iperf* tool in UDP mode.

one well-known technique is to send probing packets to the network and based on the received response form a model as a proxy for the global network view. `Ping` and `TraceRoute` are two widely used tools that utilise this technique.

While end-to-end probing is a flexible and user-level approach, depending on the type of probing to conduct it might face challenges due to its impact on network performance. Sending probing packets to the network would mean that part of network resources will become busy processing non-real data packets for helping an end-node to obtain information about the network. This technique is historically discouraged or blocked by network administrator and infrastructure providers. The challenges for TraceRoute tool and it's numerous variations to deal with imposed restrictions is an evident example of such discouragement.

The two end of network probing spectrum are `Intrusive` and `Non-Intrusive` network signalling.

Intrusive Probing Intrusive probing consists of sending bursts of data traffic to the network in order to estimate the available bandwidth and resources. While this probing policy could result in more accurate predictions, it implies wasting a significant portion of network bandwidth and resources for this probing process. As such, computer networks usually search for such probing workload and apply restrictions, using techniques such as traffic policing[11].

Non-Intrusive Probing At the other end of the spectrum, the most ideal type of network probing does not involve sending any artificial packets to the network. Instead, the end-node would solely rely on the organic signals it receive from network as a result of transferring real user traffic on the network. Despite it benefits and desired behaviour, it is not a practical method for probing network resources in most scenarios. Firstly, solely relying on the organic acknowledgement and signals would limit our exposure to the network and only gives us inconsistent messages which could be very challenging to impossible to draw any conclusion on them. Secondly, such network signals are usually being consumed by the lower layers in the networking stack and are being discarded before handing data over to the user-space. Hence, there is little opportunity to utilise non-intrusive signalling techniques for investigating network characteristics, including the type of background traffic.

A Practical Trade-off: Minimally Intrusive Probing As a practical trade-off between intrusive and non-intrusive methods, we aim to use a *minimally intrusive* probing method. Such a method would enable us to develop sufficient insight about the network, and at the same time would impose minimum impact on the network resources and so will be more probable to be allowed to be carried over and across the networks. In this study, we use RTT as our main network probing facility, studying it variation over time as a potential signature to identify the mixture of background traffic on the network. In particular, in the following section we periodically probe the end-to-end RTT using the `ping` tool to form time-series of RTT values over time. We then use these RTT time-series as a proxy for predicting the mixture of traffic on the network.

5 Learnability of TCP-based Traffic Signatures

In this section, we investigate the feasibility of building classification models to predict the type of background traffic using end-to-end probed RTT time-series. As a proof-of-concept, for this study we investigate the learnability of a mixture of TCP-only traffic on the network (§ 3.1). In particular, we consider the following 6 distinct classes of TCP-based traffic mix to be represented by the prediction model:

1. **No Traffic (B0-C0).** No active data communication on the network.
2. **Single CUBIC (B0-C1).** A single TCP CUBIC stream running on the network.
3. **Single BBR (B1-C0).** A single TCP BBR stream running on the network.
4. **Double CUBIC (C1-C1).** Two TCP CUBIC streams running on the network.
5. **Double BBR (B1-B1).** Two TCP BBR streams running on the network.
6. **CUBIC and BBR (B1-C1).** Two TCP streams running on the network: one CUBIC and one BBR.

Intuitively, for the RTT time-series to be used as the input for prediction, they should hold two qualities:

1. Distinct Patterns between traffic signatures: the RTT time-series for different traffic mixtures should be reasonably distinct in order to be trainable with a reasonable accuracy.
2. Repetitive Pattern within each time-series: in order to be able to train a classifier to generalise well to unseen cases of RTT time-series, repetitive patterns should exist in each traffic mixture signature over time.

We have conducted a series of experiments running the above six classes of traffic mixtures, probing end-to-end RTT in the periods of one second. The sample signature of RTT time-series are provided in Fig. 5. The provided samples, intuitively, hold both desired qualities of interclass distinctive patterns and intraclass repetitive patterns.

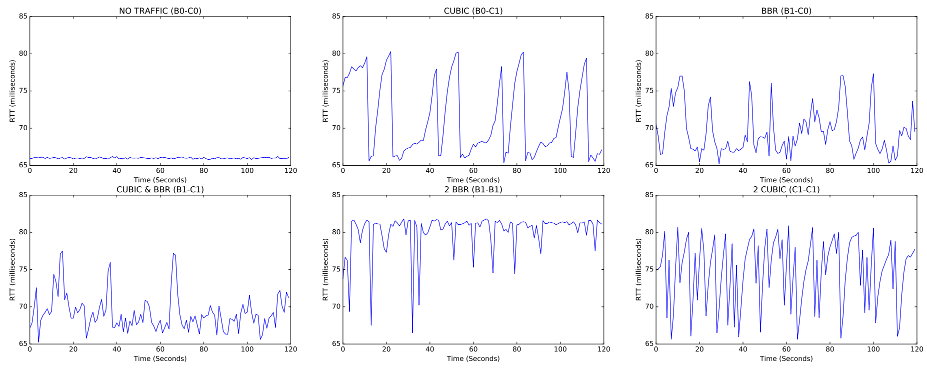


Fig. 5: A Sample of RTT Time-Series Data

In what follows we review the process of building training dataset, training prediction models, and studying the accuracy and usability of the trained models.

5.1 Building a Training Dataset of RTT Time-series

Using our controlled experimental setup presented in § 2, we gathered RTT time-series, with a 1-second sampling rate, over a one hour period of time. To smooth out the possible noise in the measured RTT value, each probing step consists of sending 10 ping requests to the other end-host, with 1 milliseconds delay in between, and the average value is recorded as the RTT value for that second. We repeated this experiment for the six traffic classes listed above. In addition, to further investigate how well the time-series prediction generalises across varying bottleneck bandwidth configurations, we repeated the same experiment for all the configurations listed in Fig. 1b.

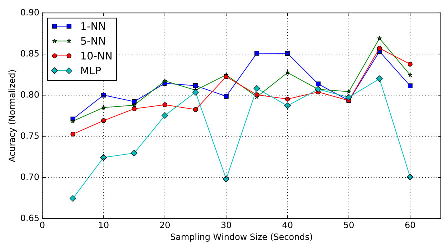
For each probing scenario, we conducted the corresponding CUBIC or BBR traffic stream between pairs of (S1,D1) and (S2,D2) pair of nodes; at the same time, we used the (S3,D3) node pair to conduct the periodic RTT probing and form the RTT time-series for that particular class.

For all data-transfer tasks, we used the *iperf* tool (`http://software.es.net/iperf/`) for generating TCP traffic of the desired CCA.

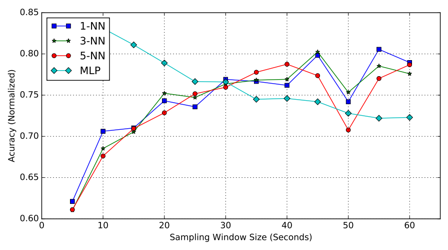
To prepare the gathered data for training process, we partition the one-hour RTT time-series into smaller chunks. For this purpose we re-used the partitioning software we developed as part of our previous study [2], partitioning the data based on a given parameter w, representing the RTT time-series length.

5.2 Training Classification Models

For this study, for training classifiers we use k-nearest-neighbours (K-NN), and a multi-layer perceptron (MLP) neural network [13, 5]. As for K-NN, we use it along with the well-known dynamic time warping (DTW) distance metric, for the time-series data. DTW was originally used for speech recognition. It was then proposed to be used for pattern finding in time-series data, where



(a) Prediction Accuracy for models trained over a single Configuration (BtlBw=500 Mb/s, RTT=65ms)



(b) Prediction Accuracy for models trained over the mixture of all Configurations (Fig. 1b)

Fig. 6: Learnability of TCP-based Traffic Signatures. Accuracy of K-NN w/ DTW and MLP models

Euclidean distance results in a poor accuracy due to possible time-shifts in the time-series [4, 15]. For MLP, we use a network with 2 dense hidden layers of $w * 1.5$ and w nodes, both with ReLU activation function. The output layer applies SoftMax activation function. To avoid overfitting *Dropout* technique is uniformly applied through the layers.

To make an appropriate decision about the parameter value w, length of RTT time-series entries, we did a parameter-sweep experiment where we calculated the classification accuracy for all the classifiers, varying parameter w from 5 seconds to 60 seconds, with a 5 second step.

To better estimate accuracy while avoiding overfitting, we use five-fold cross-validation. The reported results are the average accuracy over the five folds on the cross-validation scheme. Fig. 6 represents the average accuracy per window size w (in seconds), calculated for three variations of K-NN and MLP models.

Fig. 6a presents prediction accuracy for the models trained for a single network configuration. Fig. 6b shows the prediction accuracy when the models are trained using the mixed traffic signatures of all configurations.

For both single-configuration and mixed-configuration scenarios, K-NN with DTW yields a better accuracy in most cases. Since K-NN compares against real data points, it is very efficient in making more accurate prediction on average. 1-NN in particular offers the highest accuracy between tested K-NN variations. In contrast, MLP shows variable accuracy, highly sensitive to the parameter w. On the one hand, in Fig. 6a, MLP has lower accuracy than K-NN across different values of w. On the other hand, in Fig. 6b, MLP has the highest accuracy for w below 25 seconds, but MLP becomes less accurate than K-NN for larger values of w. Recall that, given our methodology, the number of training data instances decreases as w increases.

Although the preference of K-NN vs. MLP depends on the choice of w, the experiment does show that a ML classifier is capable of being accurate in the range of 0.75 to 0.85 for mixed traffic signatures. This accuracy level, being achieved using our relatively simple classification models, confirms our hypothesis on learnability of end-to-end TCP-based traffic signatures using RTT time-series. According to our other study, this level of accuracy is sufficient to improve on the throughput and fairness of the TCP-based data transfers on a shared network, running a mixture of CUBIC and BBR CCAs [2]. Further studying the extensibility of this approach to other mixtures of TCP- and non-TCP-based traffic, as well as adopting more sophisticated classification models such as RNN and LSTM deep neural networks [16], form the next steps of this study to be pursued in the future work.

6 Other Related Work

ML techniques have been used for designing or optimising network protocols. RemyCC uses simulation and ML to create a new TCP CCA, via a decentralised partially-observable Markov decision process (dec-POMDP) [24]. Performance-oriented Congestion Control (PCC) is another recent study where an online learning approach is incorporated into the structure of the TCP CCA. [9]. Online convex optimisation is applied to design a rate-control algorithm for TCP, called Vivace [10]. Another recent approach has been to apply deep reinforcement learning techniques for constructing CCA algorithms [17].

Another line of work includes applying ML techniques to discover network properties. Estimating the available bandwidth in high-speed networks [20, 25], identifying TCP CCA in traffic traces [21, 8], and predicting TCP unique behaviours and behaviour anomalies [22] are among the topics in this category.

7 Concluding Remarks

We investigated the learnability of traffic signatures of background traffic in shared networks. Currently, the signatures are RTT time-series data based on minimally intrusive end-to-end probing. We gathered a labelled training dataset of 6 different classes of TCP-based background traffic, on a testbed that emulates a shared WAN. Different classifier models were trained, using the signatures. We performed a simple parameter sweep of the available bandwidth of the bottleneck in the testbed. Such prediction models could prove useful for a variety of use-case scenarios, including protocol selection for a data transfer, and network tuning and optimisation.

For future work, we will consider adding more complex traffic mixes, including UDP-based traffic, high-performance data transfer tools (e.g., GridFTP), bursty traffic, and more. We hypothesise that by having sufficiently large training datasets, more sophisticated classification algorithms such as deep neural networks and LSTM might be used.

Acknowledgement: Thank you to Jesse Huard for the original implemenation of MLP.

References

1. Anvari, H., Lu, P.: Large transfers for data analytics on shared wide-area networks. In: Proceedings of the ACM International Conference on

Computing Frontiers. pp. 418–423. CF '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2903150.2911718, http://doi.acm.org/10.1145/2903150.2911718

2. Anvari, H., Huard, J., Lu, P.: Machine-learned classifiers for protocol selection on a shared network. In: Renault, É., Mühlethaler, P., Boumerdassi, S. (eds.) Machine Learning for Networking. pp. 98–116. Springer International Publishing, Cham (2019)
3. Anvari, H., Lu, P.: The impact of large-data transfers in shared wide-area networks: An empirical study. Procedia Computer Science **108**, 1702 – 1711 (2017). https://doi.org/https://doi.org/10.1016/j.procs.2017.05.211, http://www.sciencedirect.com/science/article/pii/S1877050917308049, international Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland
4. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining. pp. 359–370. AAAIWS'94, AAAI Press (1994), http://dl.acm.org/citation.cfm?id=3000850.3000887
5. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg (2006)
6. Carbone, M., Rizzo, L.: Dummynet revisited. SIGCOMM Comput. Commun. Rev. **40**(2), 12–20 (Apr 2010). https://doi.org/10.1145/1764873.1764876, http://doi.acm.org/10.1145/1764873.1764876
7. Cardwell, N., Cheng, Y., Gunn, C.S., Yeganeh, S.H., Jacobson, V.: Bbr: Congestion-based congestion control. Queue **14**(5), 50:20–50:53 (Oct 2016). https://doi.org/10.1145/3012426.3022184, http://doi.acm.org/10.1145/3012426.3022184
8. Chen, X., Xu, S., Chen, X., Cao, S., Zhang, S., Sun, Y.: Passive TCP identification for wired and wireless networks: A long-short term memory approach. CoRR **abs/1904.04430** (2019), http://arxiv.org/abs/1904.04430
9. Dong, M., Li, Q., Zarchy, D., Godfrey, P.B., Schapira, M.: PCC: Re-architecting congestion control for consistent high performance. In: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15). pp. 395–408. USENIX Association, Oakland, CA (2015), https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/dong
10. Dong, M., Meng, T., Zarchy, D., Arslan, E., Gilad, Y., Godfrey, B., Schapira, M.: PCC vivace: Online-learning congestion control. In: 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18). pp. 343–356. USENIX Association, Renton, WA (2018), https://www.usenix.org/conference/nsdi18/presentation/dong
11. Flach, T., Papageorge, P., Terzis, A., Pedrosa, L., Cheng, Y., Karim, T., Katz-Bassett, E., Govindan, R.: An internet-wide analysis of traffic policing. In: Proceedings of the 2016 ACM SIGCOMM Conference. p. 468–482. SIGCOMM '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2934872.2934873, https://doi.org/10.1145/2934872.2934873
12. Ha, S., Rhee, I., Xu, L.: Cubic: A new tcp-friendly high-speed tcp variant. SIGOPS Oper. Syst. Rev. **42**(5), 64–74 (Jul 2008). https://doi.org/10.1145/1400097.1400105, http://doi.acm.org/10.1145/1400097.1400105
13. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer-Verlag New York (2009)

14. He, E., Leigh, J., Yu, O., DeFanti, T.A.: Reliable blast udp: Predictable high performance bulk data transfer. In: Proceedings of the IEEE International Conference on Cluster Computing. pp. 317–. CLUSTER '02, IEEE Computer Society, Washington, DC, USA (2002), http://dl.acm.org/citation.cfm?id=792762.793299
15. Hsu, C.J., Huang, K.S., Yang, C.B., Guo, Y.P.: Flexible dynamic time warping for time series classification. Procedia Computer Science **51**, 2838 – 2842 (2015). https://doi.org/https://doi.org/10.1016/j.procs.2015.05.444, http://www.sciencedirect.com/science/article/pii/S1877050915012521, international Conference On Computational Science, ICCS 2015
16. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data Mining and Knowledge Discovery **33**(4), 917–963 (2019). https://doi.org/10.1007/s10618-019-00619-1, https://doi.org/10.1007/s10618-019-00619-1
17. Jay, N., Rotman, N., Godfrey, B., Schapira, M., Tamar, A.: A deep reinforcement learning perspective on internet congestion control. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 3050–3059. PMLR, Long Beach, California, USA (09–15 Jun 2019), http://proceedings.mlr.press/v97/jay19a.html
18. Kozu, T., Akiyama, Y., Yamaguchi, S.: Improving rtt fairness on cubic tcp. In: 2013 First International Symposium on Computing and Networking. pp. 162–167 (Dec 2013). https://doi.org/10.1109/CANDAR.2013.30
19. Ma, S., Jiang, J., Wang, W., Li, B.: Towards RTT fairness of congestion-based congestion control. CoRR **abs/1706.09115** (2017), http://arxiv.org/abs/1706.09115
20. Mirza, M., Sommers, J., Barford, P., Zhu, X.: A machine learning approach to tcp throughput prediction. IEEE/ACM Trans. Netw. **18**(4), 1026–1039 (Aug 2010). https://doi.org/10.1109/TNET.2009.2037812, http://dx.doi.org/10.1109/TNET.2009.2037812
21. Mishra, A., Sun, X., Jain, A., Pande, S., Joshi, R., Leong, B.: The great internet tcp congestion control census. Proc. ACM Meas. Anal. Comput. Syst. **3**(3) (Dec 2019). https://doi.org/10.1145/3366693, https://doi.org/10.1145/3366693
22. Papadimitriou, G., Kiran, M., Wang, C., Mandal, A., Deelman, E.: Training classifiers to identify tcp signatures in scientific workflows. In: 2019 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS). pp. 61–68 (Nov 2019). https://doi.org/10.1109/INDIS49552.2019.00012
23. Vishwanath, K.V., Vahdat, A.: Evaluating distributed systems: Does background traffic matter? In: USENIX 2008 Annual Technical Conference. pp. 227–240. ATC'08, USENIX Association, Berkeley, CA, USA (2008), http://dl.acm.org/citation.cfm?id=1404014.1404031
24. Winstein, K., Balakrishnan, H.: Tcp ex machina: Computer-generated congestion control. In: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM. pp. 123–134. SIGCOMM '13, ACM, New York, NY, USA (2013). https://doi.org/10.1145/2486001.2486020, http://doi.acm.org/10.1145/2486001.2486020
25. Yin, Q., Kaur, J.: Can Machine Learning Benefit Bandwidth Estimation at Ultra-high Speeds?, pp. 397–411. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-30505-9_30, http://dx.doi.org/10.1007/978-3-319-30505-9_30