Technical Report: Virtual Explicit Congestion Notification

Tingting Lu
Tandon School of Engineering
New York University
New York, 11201
Email: tl984@nyu.edu

Abstract—**Fifth generation (5G) milimeter wave (mmWave) communication provides high speed, high capacity wireless communication networks to future application. 5G mmWave channel is sensitive to obstructions, which result in high fluctuations. This requires a fast reacting TCP congestion detection scheme. In this paper, we present a new end-to-end TCP congestion detection scheme for high speed network: Virtual Explicit Congestion Notification (Virtual ECN), which makes the design of fast reacting congestion control to be promising. ECN is available in most commercial routers today. The ECN bit of a packet is enabled and marked if the occupied buffer queue in the router exceeds a threshold on the arriving of the packet. This marked ECN bit indicates the congestion happened in the router. The receiver piggybacks the marked ECN bit to the sender. Therefore, end user can respond to the congestion and reduce their sending speed without generating extra traffic. In this paper, we develop a Virtual ECN scheme which enables the end user to determine if congestion happened in the intermediate network without involving any other network components (routers, etc.). The end user predicts the Virtual ECN based on the data it can collect, such as packet sending time, ACK receiving time, etc. In this paper, we demonstrate the Virtual ECN scheme can predict the congestion with high precision in a relatively short time period.**

I. Introduction

The explosive demand of mobile data motivated the development of 5G communication network, which fully exploited the spectrum of mmWave channel. mmWave channel provides huge amount of capacity and relatively low delay while it suffers high propagation loss and sensitivity to blockage. Simulations and measurements showed that the capacity gain is tremendous comparing with current cellular systems in [1, 2]. The last hop wireless channel capacity is not a bottleneck problem anymore. Meanwhile, mmWave channels are prone to variations due to the blockage from walls, trees, or even human body [3–5]. This feature brings high capacity variations to the channel. The channel frequently switches between Line-of-Sight and Non-Line-of-Sight. Currently, the upper layer channel provided by mmWave communication network is not standardized yet. But generally, researchers agreed that 5G mmWave channel will provide high capacity and low latency with high fluctuation. For 5G mmWave wireless communication network, the traditional TCP congestion control won't work due to the conservative slow congestion detection. Therefore designing a new TCP congestion control scheme without involving any intermediate network device is necessary. In this paper, we are addressing the initial step to design such scheme: design an end-to-end fast data driven congestion detection algorithm.

In recent years, several promising TCP congestion control protocols are developed: BIC TCP [6], CUBIC TCP [7], Compound TCP [8], etc. Most of these TCP congestion control schemes rely on 3 Duplicated ACKs or Time Out to detect the congestion and packet loss. This means TCP needs to wait for at least 3 ACKs or time out timer before entering into recovery mode or slow start mode. The slow congestion detection will deteriorate the network performance for a high capacity, low latency and high variation mmWave channel. First, for high capacity link, during this time interval, more congestion and packet loss might occur and cause high payoff to recover from the loss. Second, for a high variation network whose link capacity is fluctuate tremendously, the slow congestion detection scheme might provide out of date information of the congestion status, which makes the TCP congestion control protocol extremely conservative.

Due to the particular behavior of mmWave channel, a fast end-to-end congestion detection scheme is necessary. In this paper, we design such a scheme of congestion detection which does not rely on duplicated ACKs or time out. The paper is organized as follows: Section II gives the related work on TCP congestion control and mmWave channel. We also state our motivations to design this scheme. Section III demonstrates our design of the fast end-to-end congestion detection scheme in detail including network simulation setup, data process and congestion prediction. Section IV summarize the paper and propose future work can be accomplished based on this scheme.

II. Related Work

In this section, we present the related work and motivations of our work. The design of a fast and accurate congestion detection algorithm is necessary to the next generation mmWave wireless communication system.

A. mmWave Channel

In [9][10], researches on the feasibility of using mmWave to provide wireless communication service are presented. [1] investigated the mmWave propagation loss, penetration loss, interference etc. The author provided a detailed statistical mmWave channel model based on outdoor measurements. This model shows that mmWave system can provide an order of magnitude increase in capacity and provides the probability of outage and non-outage. Coverage design technology and beamforming scheme are studied in[11] [12], which convince that mmWave can be used for future wireless communication network design. Although mmWave channel can provide high capacity and low latency, it has a challenging issue: mmWave is sensitive to the obstructions caused by buildings, trees and even nearby pedestrian or vehicles. These obstructions may cause sudden, short and severe channel downgrade. How to react fast to the channel fluctuations and fully use the channel capacity is the main issue for TCP congestion control protocol design. This paper focus on solving this problem.

B. Explicit Congestion Notification (ECN)

Legacy TCP congestion control protocols considered duplicate ACKs as indication of congestion. With the addition of active queue management and ECN [13] to the network infrastructure, routers are capable to detect congestion before queue overflows. An additional Congestion Experienced (CE) byte is added to the IP header of the packet to support this function. Routers running RED queue management can mark the CE codepoint with certain probability when the average queue length is between minimal threshold and maximal threshold. When the average queue length exceeds the maximal threshold, all upcoming packets' CE byte will be marked with probability 1. The packet with CE marked will be sent to the receiver. When the receiver received the packet, it will piggyback an ACK with CE marked to notify the sender that there is congestion in the router. No extra traffic will be generated to overload the network. The sender will reduce the congestion window to avoid queue overflows and packet loss. Using ECN to detect congestion requires the participation of routers. To mimic the congestion detection using ECN without involving any network devices, we developed a Virtual ECN.

C. Remy

Remy [14] is proposed by Keith Winstein in 2013. It designed congestion-control schemes based on prior knowledge of the network and the traffic model. It was the first computer generated congestion control protocol. There a RemyCC tracks just three state variables of the end user:

- An exponentially-weighted moving average (EWMA) of the interarrival time between new ACKs received (ack_ewma).
- An EWMA of the time between TCP sender timestamps reflected in those ACKs (send_ewma).
- The ratio between the most recent RTT and the minimum RTT seen during the current connection (rtt_ratio).

The algorithm does not require the support from intermediate network devices. The end user can adjust its sending speed based on the above three data collected by itself. This indicates these three variables together might reflect the congestion state of the network.

Remy convinced us that designing a data driven congestion detection algorithm using end user data is possible. Since ECN gives the accurate information of whether there is congestion in the network or not, we decide to use end user data to predict ECN, thereby detect the congestion. The main contribution of this paper is developing a data driven fast congestion detection algorithm (within one ACK interarrival time) purely using the end user data without support from receiver or router or other network devices.

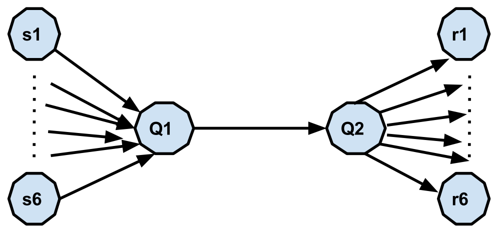


Fig. 1. Network Topology

III. Congestion Detection Algorithm

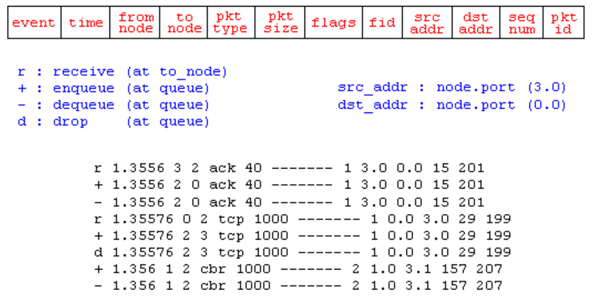
In this section, the design of the fast congestion detection algorithm is illustrated in detail. The key idea is to predict the congestion in network within one interarrival time interval of ACKs from the end user data. The data can be collected from end user includes packet sending time, ACK arriving time and the Round-Trip-Time (RTT) calculated from these two time stamps. To collect these data for a particular network topology, we did simulations using discrete event based Network Simulator 2 (NS2).

A. Network Topology

A typical dumbbell model is studied in this paper. The network topology is as in Fig. 1. There are 6 traffic senders and 6 receivers, sender i send data to receiver i through the bottleneck link $Q1 \rightarrow Q2$. Both $Q1$ and $Q2$ maintain a DropTail Queue. The queue limit is set to be bandwidth-delay product to ensure the fully utilization of the link capacity. Congestion happens when the occupied queue length exceeds some threshold T on the arriving of the incoming packet. To generate traces with congestion identifiers, we tag every packet as Congestion Experienced in the queue when the queue length is greater than T. Typical NS2 TCP traces are collected for future machine learning as in Fig. 2 [15].

From the trace format we can identify the packet sending time, arriving time, packet size, unique ID, etc. Matching the packet unique ID, RTT can be calculated from packet sending time and corresponding ACK receiving time. For each unique packet, if congestion happens when it arrives, then the unique ID was recorded to be congestion experienced.

The drop tail queue behavior is displayed as follows. Fig. 3 and Fig. 4 are two different queue behavior patterns. Fig. 3 shows how queue changes when part of the traffic sources are running Cubic TCP with UDP competing traffic while Fig. 4 shows the case one part of traffic sources are running NewReno TCP with competing UDP traffic. In these two



event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id

```
r : receive  (at to_node)
+ : enqueue  (at queue)            src_addr : node.port (3.0)
- : dequeue  (at queue)            dst_addr : node.port (0.0)
d : drop     (at queue)

         r 1.3556 3 2 ack 40 ------- 1 3.0 0.0 15 201
         + 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
         - 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
         r 1.35576 0 2 tcp 1000 ------- 1 0.0 3.0 29 199
         + 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
         d 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
         + 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
         - 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
```

Fig. 2. NS2 TCP Trace Format

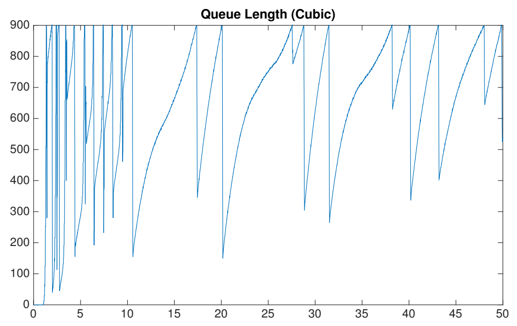


Fig. 3. Queue Length Dynamics (Cubic)

cases, when queue length is greater than 450 packets, the incoming packets will be marked as congestion experienced.

B. Data Collection

To avoid losing generality of the analysis, we simulated several different traffic scenarios to collect end user data. We investigated following scenarios:

- Scenario 1: All traffic sources are running TCP NewReno with stationary bottleneck link capacity.
- Scenario 2: All traffic sources are running TCP Cubic with stationary bottleneck link capacity.
- Scenario 3: One half of traffic sources are running TCP NewReno while the other half are running UDP with stationary bottleneck link capacity.

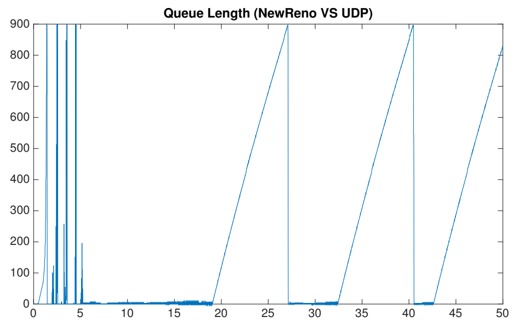


Fig. 4. Queue Length Dynamics (NewReno VS UDP)

TABLE I
SIMULATION PARAMETERS

Access Link Capacity	1Gbps
BottleNeck Link Capacity	1Gbps
Access Link Delay	5ms
BottleNeck Link Delay	10ms
Queue Capacity	Delay-Bandwidth Product

- Scenario 4: One half of traffic sources are running TCP Cubic while the other half are running UDP with stationary bottleneck link capacity.
- Scenario 5: Repeat scenario 1 and 2 for a periodically On-Off bottleneck link.

The bottleneck link for those scenarios is link $Q1 \rightarrow Q2$. Detailed simulation parameters are showed in TABLE I.

We collect data from end user. The data which is available from end user are packet sending time t_s and ACK receiving time t_r. From these two time stamps, we can calculate packet sending interval $T_{s_i} = t_{s_i} - t_{s_{i-1}}$, ACK interarrival time $T_{r_i} = t_{r_i} - t_{r_{i-1}}$ and Round Trip Time (RTT) $RTT_i = t_{s_i} - t_{r_i}$. From the above three available time intervals, we develop 5 features that are used in the congestion prediction.

- Sending time interval between two consecutive packets T_{s_i}.
- Receiving time interval between two consecutive ACKs T_{r_i}.
- An exponentially-weighted moving average (EWMA) of T_{s_i} (send_ewma).
- An EWMA of T_{r_i} (ack_ewma).
- The ratio between the most recent RTT and the minimum RTT seen during the entire connection period (rtt_ratio).

These above five features are used to train the machine learning algorithm and predict the congestion.

C. Congestion Prediction

Each sample of the data collected from above five scenarios includes the five features and a congestion identifier: Virtual ECN (0 stands for no congestion, 1 stands for congestion). The problem to be solved is a binary classification problem: train an algorithm to classify the sample data into two categories: congestion experienced and no congestion. First we ran Logistic Regression, a popular binary classification algorithm, on the sample data. The area under ROC is 0.89 as showed in Figure. 5. The performance of Logistic Regression is good $(0.80 - 0.90)$. The decision boundary is $-2.31936102 * f_1 - 33.36046689 * f_2 - 12.65365902 * f_3 - 15.29961927 * f_4 + 2.0344486 * f_5 - 2.69202498 = 0$, where f_1, f_2, f_3, f_4, f_5 are the five corresponding features.

Next we applied Decision Tree algorithm and regulated the maximum depth of the tree to be 5 and the minimum samples in each leaf node to be 10000 to avoid overfitting. Figure. 6 gives the ROC curve of Decision Tree Classifier. The area under ROC curve is 0.98. This indicates excellent $(0.90 - 1)$ prediction result. The feature importance are given as [0.011703, 0.00085, 0.000000, 0.03985, 0.94760] for [T_{s_i},

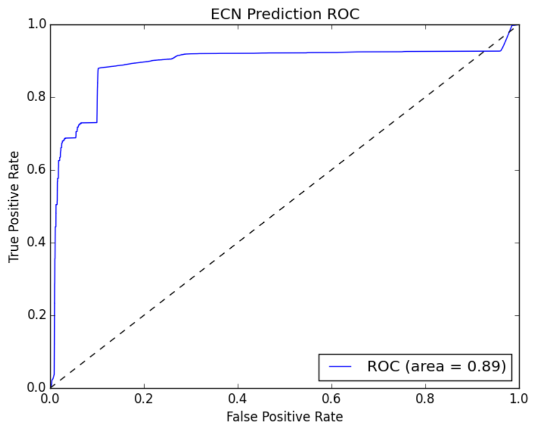


Fig. 5. ROC curve of Logistic Regression Classifier

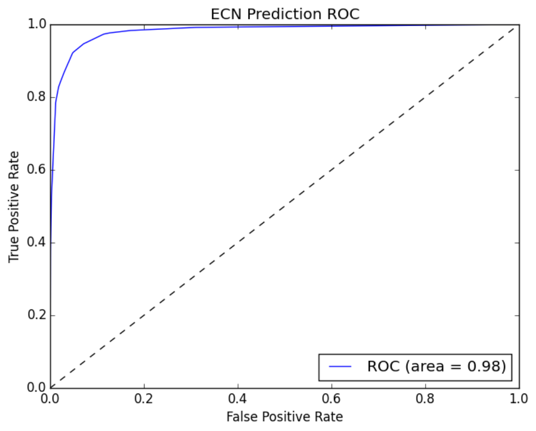


Fig. 6. ROC curve of Decision Tree Classifier (max_depth = 5, min_samples_leaf = 10000)

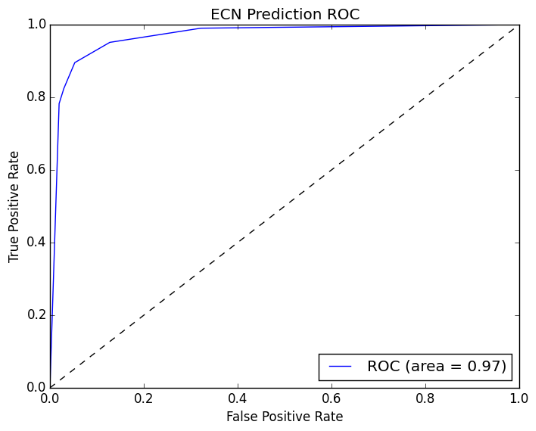


Fig. 7. ROC curve of Decision Tree Classifier (max_depth = 3, min_samples_leaf = 100000)

TABLE II
SIMULATION PARAMETERS

Access Link Delay	5ms, 10ms, 20ms
BottleNeck Link Delay	10ms, 20ms, 40ms
Queue Capacity	Delay-Bandwidth Product

T_{r_i}, send_ewma, receive_ewma, rtt_ratio], from which we can conclude that rtt_ratio is the most important feature to predict congestion. Further we changed the maximum depth to 3 and minimum samples of each leaf node to 100000, the ROC is displayed in Figure. 7. The feature importance becomes [0, 0, 0, 0, 1]. This indicates the decision tree is doing the binary classification merely based on rtt_ratio. All other features are not contributed to the binary classification process. The decision tree is showed in Figure. 8. In Figure. 8, x[4] is rtt_ratio, the first layer is classified by this feature. The cumulative density function (CDF) of rtt_ratio is given in Figure. 9. The blue line is the CDF curve of rtt_ratio when congestion experienced while the red line is when congestion not experienced. Take 1.13 as an example, the probability of rtt_ratio to be less than 1.13 is 89.3% when there is no congestion However, that probability is just 11.7% when congestion experienced. This is consistent with the result of feature importance generated from Decision Tree algorithm.

The above results are generated from a network with fixed propagation delay. To study how this prediction algorithm works when network condition changes, we investigated Scenario 1 and 2 with variations of propagation delay. We simulated networks with access delay and queue capacity as in TABLE II. The Logistic Regression algorithm generated the ROC curve as in Figure. 10. The area under ROC curve is 0.654497, which indicates poor performance $(0.60-0.70)$. The decision boundary is $-0.46605323*f_1-7.33485574*f_2-0.34620622*f_3-1.41154115*f_4+5.20171832*f_5-5.53557771=0$. The coefficient of each feature follows the same pattern as before. Decision Tree classifier is then applied to the collected dataset. The ROC curve is as in Figure. 11. The area under ROC curve is 0.701188, which is on the boundary of poor performance. The feature importance is [0.000955787581, 0.000389816495, 0.00489484823, 0.0636899225, 0.930069625]. This is consistent with the previous result that rtt_ratio is the most important feature. Those two figures indicates that when network condition changes, the predictability of congestion from the aforementioned five features decreases. Figure. 12 shows the CDF of rtt_ratio with different link propagation delay. The figure indicates near 30% overlap of the CDF curve of Congestion Experience packet and No Congestion packet. This again verifies the prediction results.

IV. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a data driven machine learning congestion detection algorithm. Datasets are collected using NS2 for a dumbbell model with five different traffic scenarios. Five features are formatted from end user data. When the network condition is not changing, which means the propagation delay of each link is stable, our algorithm can detect congestion with high precision from the five features. In contrast, if

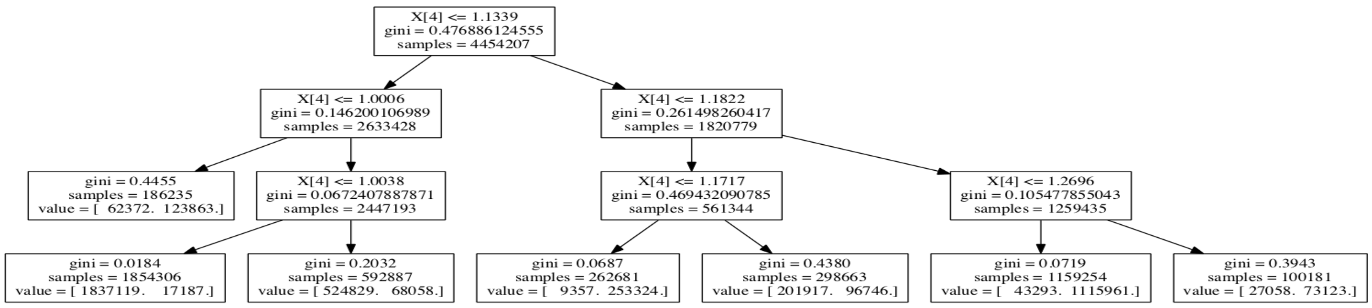


Fig. 8. Decision Tree (max_depth = 3, min_samples_leaf = 100000)

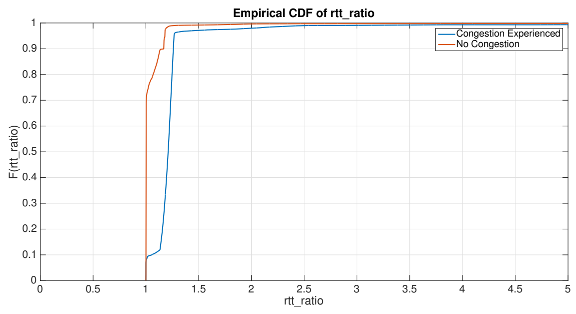


Fig. 9. CDF of rtt_ratio

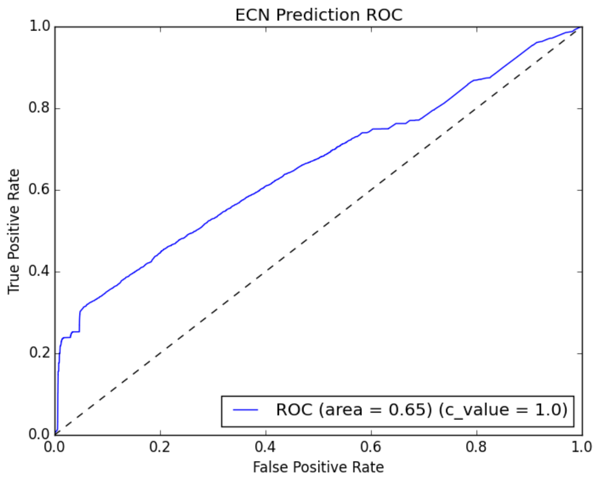


Fig. 10. ROC curve of Logistic Regression Classifier with Different Propagation Delay

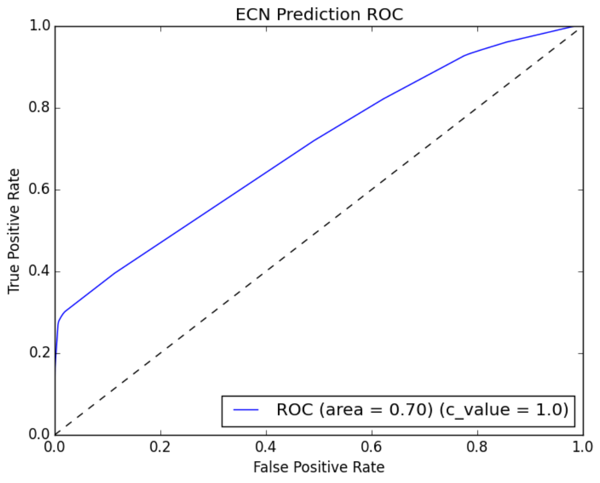


Fig. 11. ROC curve of Decision Tree Classifier with Different Propagation Delay

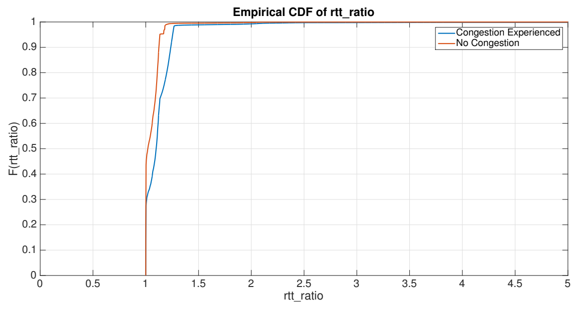


Fig. 12. CDF of rtt_ratio with Different Propagation Delay

the network condition changes, the algorithm fails to work. In both cases, rtt_ratio is the most important feature to predict congestion. Other four features are of super less useful when doing the binary classification. The area under ROC curve is consistent with the CDF of rtt_ratio. In the future, to find an algorithm works for a network with time varying delays, more useful features like rtt_ratio need to be developed and more datasets need to be collected. Besides, other intelligent machine learning algorithm or binary classification algorithm can be investigated to check if there exists a machine learning congestion detection algorithm for a time-varying network. For stable network, a fast reacting congestion control algorithm can be designed based on our congestion detection algorithm for 5G mmWave communication network.

References

[1] M. R. Akdeniz, Y. Liu, M. K. Samimi, S. Sun, S. Rangan, T. S. Rappaport, and E. Erkip, "Millimeter wave channel modeling and cellular capacity evaluation," *Selected Ar-*

eas in Communications, IEEE Journal on, vol. 32, no. 6, pp. 1164–1179, 2014.

[2] T. Bai and R. W. Heath, “Coverage and rate analysis for millimeter-wave cellular networks,” *Wireless Communications, IEEE Transactions on*, vol. 14, no. 2, pp. 1100–1114, 2015.

[3] J. Lu, D. Steinbach, P. Cabrol, and P. Pietraski, “Modeling the impact of human blockers in millimeter wave radio links,” *ZTE Commun. Mag*, vol. 10, no. 4, pp. 23–28, 2012.

[4] H. Zhao, R. Mayzus, S. Sun, M. Samimi, J. K. Schulz, Y. Azar, K. Wang, G. N. Wong, F. Gutierrez, and T. S. Rappaport, “28 ghz millimeter wave cellular communication measurements for reflection and penetration loss in and around buildings in new york city,” in *Communications (ICC), 2013 IEEE International Conference on*, pp. 5163–5167, IEEE, 2013.

[5] A. V. Alejos, M. G. Sánchez, and I. Cuiñas, “Measurement and analysis of propagation mechanisms at 40 ghz: Viability of site shielding forced by obstacles,” *Vehicular Technology, IEEE Transactions on*, vol. 57, no. 6, pp. 3369–3380, 2008.

[6] L. Xu, K. Harfoush, and I. Rhee, “Binary increase congestion control (bic) for fast long-distance networks,” in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4, pp. 2514–2524, IEEE, 2004.

[7] S. Ha, I. Rhee, and L. Xu, “Cubic: a new tcp-friendly high-speed tcp variant,” *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[8] K. Tan, J. Song, Q. Zhang, and M. Sridharan, “A compound tcp approach for high-speed and long distance networks,” in *Proceedings-IEEE INFOCOM*, 2006.

[9] Y. Niu, Y. Li, D. Jin, L. Su, and A. V. Vasilakos, “A survey of millimeter wave communications (mmwave) for 5g: opportunities and challenges,” *Wireless Networks*, pp. 1–20, 2015.

[10] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez, “Millimeter wave mobile communications for 5g cellular: It will work!,” *Access, IEEE*, vol. 1, pp. 335–349, 2013.

[11] S. Sun, G. R. MacCartney, M. K. Samimi, S. Nie, and T. S. Rappaport, “Millimeter wave multi-beam antenna combining for 5g cellular link improvement in new york city,” in *Communications (ICC), 2014 IEEE International Conference on*, pp. 5468–5473, IEEE, 2014.

[12] W. Roh, J.-Y. Seol, J. Park, B. Lee, J. Lee, Y. Kim, J. Cho, K. Cheun, and F. Aryanfar, “Millimeter-wave beamforming as an enabling technology for 5g cellular communications: theoretical feasibility and prototype results,” *Communications Magazine, IEEE*, vol. 52, no. 2, pp. 106–113, 2014.

[13] K. Ramakrishnan, S. Floyd, and D. Black, “Rfc 3168,” *The addition of Explicit Congestion Notification (ECN) to IP. The Internet Society*, 2001.

[14] K. Winstein and H. Balakrishnan, “Tcp ex machina: Computer-generated congestion control,” in *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 123–134, ACM, 2013.

[15] “Trace analysis example.” http://nile.wpi.edu/NS/analysis.html. Accessed: 2016-4-5.