ΔQ Generative Models: Modeling Time-Variation in Network Quality

1st Bjørn Ivar Teigen
Department of Informatics
University of Oslo
Oslo, Norway
bjornite@ifi.uio.no

2nd Neil Davies
Predictable Network Solutions Limited
Stonehouse, United Kingdom
neil.davies@pnsol.com

3nd Peter Thompson
Predictable Network Solutions Limited
Bristol, United Kingdom
peter.thompson@pnsol.com

4rd Kai Olav Ellefsen
Department of Informatics
University of Oslo
Oslo, Norway
kaiolae@ifi.uio.no

5th Tor Skeie
Department of Informatics
University of Oslo
Oslo, Norway
tskeie@ifi.uio.no

6th Jim Torresen
Department of Informatics
University of Oslo
Oslo, Norway
jimtoer@ifi.uio.no

***Abstract*—This work introduces a class of network performance models designed to capture variations in network quality on diverse timescales. By explicitly modeling how quality changes over time, the proposed models enable computation of performance metrics that are beyond the scope of steady-state methods such as Markov chains. We use the quality attenuation (ΔQ) metric to quantify network quality, and ΔQ generative models specify how quality attenuation varies over time. Variation over time is modeled using a finite state machine with timed state transitions. We show how the models can be used to shed light on practical problems by presenting novel results for the problem of buffer sizing. In addition to the buffer sizing results, this work presents the ΔQ generative model structure and the basic algorithms needed to work with the models.**

***Index Terms*—Computer network performance, ΔQ, Quality attenuation, Buffer sizing**

I. Introduction

Internet protocols such as TCP [1] and QUIC [2] are designed to adapt to network performance variations. However, the response time of these protocols is limited by the delay of feedback signals. Therefore, it is important to develop an understanding of performance variations on timescales that are too small for end-to-end adaptation to react, but still large enough to affect queuing behavior and application performance. In the literature, there are two main approaches to model variations in network performance: *Steady-state analysis*, which captures system behavior as an average over all of time, and *simulation* which captures performance in specifically modeled scenarios with great accuracy. This work proposes a way to interpolate between steady-state methods and simulation by defining a class of models where the timescale of performance variations is itself variable.

Internet throughput has improved much over the last few decades, and in some places, it has now reached a point where more bandwidth does not significantly improve the end-user experience [3]. Perhaps because of the diminishing returns of increased bandwidth, research on improving internet quality is increasingly focused on how to deliver reliable low latency. Much of the work on how to deliver reliable low latency address issues related to the design and configuration of queuing algorithms. Queuing, also known as *buffering*, is known to be a significant source of latency variation in the Internet [4]. Queues are necessary to smooth out variations in the relative magnitude of supply and demand of network resources. Some buffering is helpful, but too much buffering, often referred to as *bufferbloat*, is undesirable because it introduces excessive latency [4]. However, what *excessive latency* means in this context is not fully understood. The answer undoubtedly depends on the situation, but it is nevertheless possible to derive theoretical descriptions of the required trade-offs. Recent theoretical advances [5], [6] have shed some much needed light on this issue, but the methods for sizing buffers are still mostly based on experimentation. The introduction and deployment of new congestion control algorithms such as BBR [7] and L4S [8] makes buffer sizing even more complicated because the interactions between queuing algorithms and congestion control algorithms are becoming more complex and varied.

There has been increasing interest in novel queuing algorithms in recent years [9]–[11]. Several of these algorithms, such as CoDel [9], PIE [10], and CAKE [11], rely on a configured set-point for the desired amount of queuing. Finding the best set-point is non-trivial and dependent on the specific behavior of the link. Our results shed new light on the trade-offs of buffer sizing decisions.

We propose ΔQ generative models in this paper. This class of models was developed while investigating the problem of buffer sizing over links with varying capacity. The contributions of this paper is to define a model structure and to motivate the level of complexity by showing how the proposed structure can help shed light on the buffer sizing issue. We believe this class of models is also well suited to machine learning and, as such, represents a step toward data-driven methods for the development and design of network performance models. A ΔQ generative model learned from a data set of observed network behavior may pick up on patterns of performance variation that are not typically implemented in network simulation work. To keep this paper focused on the model definition and why and how the models are useful, the challenge of efficient learning with ΔQ generative models will be explored in future work.

II. Background and related work

This section describes selected literature related to time-varying network quality and the ΔQ network quality metric.

Queuing theory is the mathematical background for analysis of queuing systems. Kendall first described how queuing systems can be solved by analyzing embedded Markov chains [12]. The power of this method lies in the ability to rapidly and precisely compute the long-term average performance of complex queuing systems. This method of analysis is not directly applicable to systems where the service rate varies as a function of time. Liu and Whitt [13] study a fluid model of a general class of time-varying queues and describe the performance functions. ΔQ generative models further generalize the description of the server compared to [13] by adding the possibility of packet loss.

Much work has gone into studying the behavior of TCP over the internet [14]. Studying the behavior of TCP, especially when looking at the interaction of multiple flows, is to study the effects of variable network capacity because TCP dynamically varies the load it places on the network in somewhat predictable ways. Research on congestion control over variable links like 5G [15] is also exploring similar issues by empirically studying how different congestion control algorithms deal with the capacity variation patterns specific to 5G networks. Srivastava et al. [16] explore the performance of several congestion control algorithms over an emulated 5G mmWave link. Their method, replaying the recorded performance of a link, is similar to how ΔQ generative models can be used in simulation and emulation studies. Our models generalize the method by allowing the description of link behaviour to be stochastic.

Goyal, Alizadeh, and Anderson [6] explore an optimal end-to-end congestion controller for links with varying capacity using mathematical modeling. Their central assumption is that capacity changes can be modeled as a Markov process evaluated every round-trip time. Our models of link behavior differ in how the link capacity changes, where our proposed model is the most general of the two.

Quality attenuation is a network quality metric that captures the latency and packet loss performance of packet-switched networks. A method for measuring quality attenuation has been standardized by the Broadband Forum [17]. The quality attenuation metric has been developed through several decades of academic work, but it is not widely known in the research community. Therefore, we give a brief description here. The quality attenuation metric combines latency and packet loss into a single variable where packet loss is modeled as infinite latency. Equation 1 formally defines a quality attenuation value as a probability density function $P(t)$ paired with a real number $P(\infty)$ such that $0 \leq P(\infty) \leq 1$. $P(t)$ describes the probability density over all possible latency values, and $P(\infty)$ describes the probability of packet loss (we can think of packet loss as infinite latency). We use "ΔQ" to abbreviate quality attenuation. A ΔQ value can be plotted as a cumulative density function (CDF) with a maximum value of $1 - P(\infty)$.

$$\Delta Q := [P(t),\, P(\infty)],\; t \in \mathbf{R}^+ \tag{1}$$

Haeri et al. [18] present a framework for reasoning about the timeliness of outcomes. They also use the notion of quality attenuation (ΔQ). Their methods are suitable for packet-switched networks where traffic can incur both losses and delays when traversing a link. The authors describe one of the most useful features of ΔQ; the fact that ΔQ values can be composed both sequentially and in parallel. Haeri's framework captures the distributed nature of networked applications and provides tools for reasoning about how communication delays affect application outcomes. In Haeri's model, there can be many paths between two outcomes, and each path is modeled as a stochastic process, but there is no notion of a single path varying over time. That is to say, all samples of the same path are assumed to be independently selected from the same distribution, regardless of the timing between subsequent samples. The main difference between our work and [18] is that our class of models includes variation as a function of time.

III. A class of generative models for time-varying network performance

This section defines our proposed class of models and describes some basic algorithms needed to work with models from the class. The class of models has been designed with a few key properties in mind. We want the models to be able to:

- Include delay *and* packet loss as possible observations
- Capture temporal structure at any timescale
- Model any sequence of observations
- Reduce to an expected value description when there is no temporal structure
- Facilitate learning from recorded network traces

The proposed model is our best attempt at finding a minimal structure with all of these properties.

A. Model definition

A ΔQ generative model is a 5-tuple $(\mathbf{S}, \Delta Q^{\mathbf{S}}, D_{loss}^{\mathbf{S}}, P, T)$ where $\mathbf{S}$ is a finite set of states called the state space. Each state $s \in \mathbf{S}$ has an associated ΔQ value labeled $\Delta Q^s \in \Delta Q^{\mathbf{S}}$. $\Delta Q^{\mathbf{S}}$ is the set of quality attenuation values for all states. $D_{loss}^{\mathbf{S}}$ is the set of D_{loss} for all states, where D_{loss} is a probability distribution describing the amount of time it takes to lose

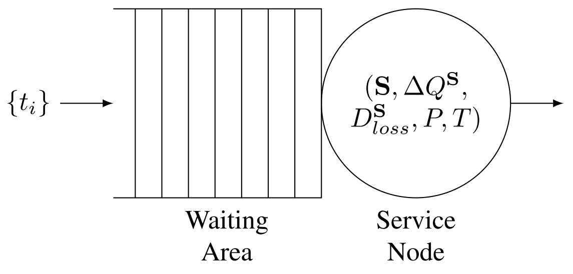

$\{t_i\}$
$(\mathbf{S}, \Delta Q^{\mathbf{S}}, D^{\mathbf{S}}_{loss}, P, T)$
Waiting Area
Service Node

Fig. 1: The ΔQ generative model describes how the service node changes over time

a packet. P is a transition matrix describing the probability of moving from state s to state s' for each pair of states $s, s' \in \mathbf{S}$. We use the notation $P(s, s')$ for the probability of moving from state s to state s'. $T(s)$ is a function describing the distribution of time spent in state s before moving to a state $s' \in \mathbf{S}$. This class of models allows us to generate arbitrary patterns of ΔQ variation as a function of time. The models allow us to capture the appropriate amount of temporal structure because very high-frequency variation can be averaged out and summarized by the ΔQ values of each of the states s. Capacity variation on the timescales we care about can be modeled as time-dependent state changes. The models describe the service pattern of a network interface. Figure 1 shows a FIFO queue where the server is described by a ΔQ generative model.

The sequence of states, $s(t)$, is a discrete function of the continuous variable t describing the state of the system at each point in time. The sequence $\{t_0, t_1, ...\}$ is a discrete set of times at which observations are made, and the sequence $\{o_{t_0}, o_{t_1}, ...\}$ are the observed quality attenuation values at each of the times $\{t_0, t_1, ...\}$. Figure 2 shows the relationship between ΔQ generative models, observations, and expected value descriptions of observations.

Given a model $\mathbf{M}$, we can choose to sample the model at any set of times $\{t_0, ..., t_n\}$. Suppose the goal is to capture the performance of a specific application across the network. In that case, it makes sense to select a set of sampling times that reflect the pattern of packet arrivals generated by the application in question. Having decided on a set of sampling times, generating a set of observations from $\mathbf{M}$ amounts to running the model forward from a starting state. We can generate trajectories through the model by randomly selecting which edge to transverse according to the probabilities of P and dwelling in state s according to a random sample from $T(s)$. When t_0 amount of time has been simulated, we generate a latency (or loss!) value o_{t_0} according to the ΔQ of the state the model happens to be in at time t_0, $s(t_0)$.

B. Computing the average service time

Given a model $\mathbf{M}$, what is the average service time? Here, Kendall's methods come to the rescue [12]. The rationale for using a ΔQ generative model is to get higher fidelity descriptions of network performance than what can be achieved with time-average methods. However, knowing the long-term average delay can still be helpful. The average service time can be used to decide what the utilization factor (the value of ρ) is for a given arrival pattern, and it can be used to find the maximum long-term average throughput of the interface. The transition matrix P contains the information we need to evaluate which state the system is in immediately after changing state because P defines an embedded Markov chain. We find the steady-state of the state distribution of the embedded Markov chain by computing the eigenvectors of the transition matrix and choosing the (normalized) eigenvector with the largest eigenvalue. Because the time spent in each state is not equal, we need to weigh the steady-state distribution of the embedded Markov chain by the average time spent in each state (this is given by T) to find the probability of finding the system in each state at a randomly sampled point in time.

Having calculated how likely it is to find the system in each of its possible states at a random point in time, we can use the probabilistic choice operator for ΔQ values [18] to compute the steady-state ΔQ. We denote the long-term average quality attenuation by $\Delta Q^{\mathbf{M}}$. We can think of this as the quality attenuation we expect to observe if we do not have any information about the state of $\mathbf{M}$. The average of a ΔQ value is infinity if there is a non-zero chance of packet loss. However, we can also think of a ΔQ value as describing the time it takes a network interface to process a packet. From this perspective, it makes sense to specify the duration of a loss, that is, the amount of time it takes to fail to transmit a packet. We define the average of a ΔQ value in equation 2, where D_{loss} is the latency distribution over processing times for lost packets.

$$\mathbf{E}[\Delta Q] = avg(P(t)) * (1 - P(\infty)) + P(\infty) * avg(D_{loss}) \quad (2)$$

The average of $\Delta Q^{\mathbf{M}}$, $\mathbf{E}[\Delta Q^{\mathbf{M}}]$, is the average service time of the model $\mathbf{M}$. The long-term average throughput of $\mathbf{M}$ is, therefore, $\frac{1}{\mathbf{E}[\Delta Q^{\mathbf{M}}]}$ times the average payload size of a packet. Given a specific arrival pattern, we can use the average service time to compute the utilization ρ.

IV. An example model: Computing necessary queuing delay

In this section, we work through a concrete example showing both how ΔQ generative models can be used and how they provide value. We describe the simplest model that adds more temporal structure than a steady-state approach, and show how the chosen example can inform buffer sizing decisions in real-world scenarios. We analyze the model both mathematically and using Monte Carlo simulation. The Monte Carlo simulation method applies to all ΔQ generative models. In contrast, the mathematical analysis only works in this case because the chosen model is simple enough to facilitate tractable mathematics. We show that both methods of analysis produce the same results.

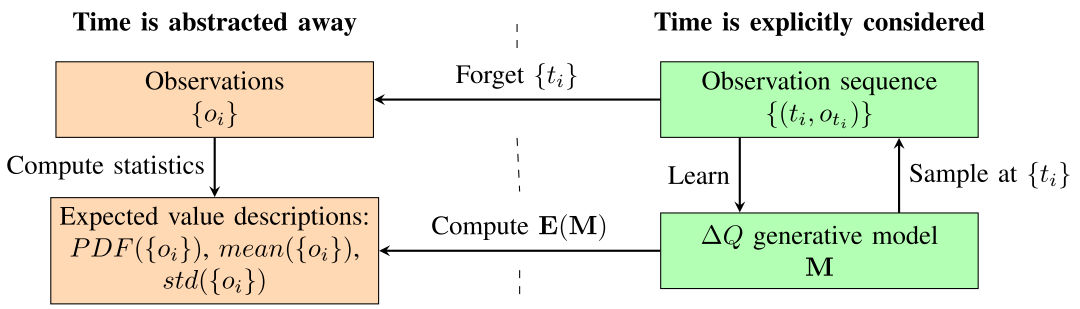

Time is abstracted away
Time is explicitly considered
Observations $\{o_i\}$
Forget $\{t_i\}$
Observation sequence $\{(t_i, o_{t_i})\}$
Compute statistics
Learn
Sample at $\{t_i\}$
Expected value descriptions: $PDF(\{o_i\})$, $mean(\{o_i\})$, $std(\{o_i\})$
Compute $\mathbf{E}(\mathbf{M})$
ΔQ generative model $\mathbf{M}$

Fig. 2: The relationships between ΔQ generative models, expected value methods and observations of network performance.

A. Problem statement

This section develops a simple example model, which we call the *square wave approximation*, and shows how ΔQ generative models can be used to reason about the relationship between queuing delay and throughput. This section shows that the temporal information captured by the functions T of the proposed model class can be crucial for good modeling of network behavior. The mathematical analysis in this section assumes a fluid-flow approximation of network traffic, whereas the Monte Carlo simulation uses packetized traffic.

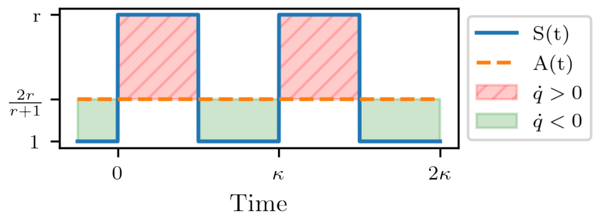
Consider a communications interface capable of sending information at a maximum rate of C Mbit/s. Periodically, the interface is disturbed so that capacity drops by a factor $1/r$ to C/r Mbit/s. We let the capacity of the interface vary as a square wave. One reason for choosing a square wave is that it serves as a bounding worst-case for capacity variations. The square wave approximation treats capacity changes as instantaneous step changes and therefore captures the kind of capacity variation that is most difficult to adapt to. In addition, a square wave makes the mathematics relatively simple.

We also assume that the interface spends an equal amount of time at each capacity before the capacity changes again (so the square wave's duty cycle is 50%). We normalize all service and arrival time values by treating them as multiples of the service time at the maximum rate. The described service pattern is labeled $S(t)$ in Figure 3(a), where normalized service time is on the y-axis, and time is on the x-axis. We denote the period of the square wave by κ and the frequency of the square wave as $f = \frac{1}{\kappa}$. An important observation is that the long-term average latency distribution for the proposed service pattern is the same for all values of κ. Figure 4 shows the model as a state transition diagram.

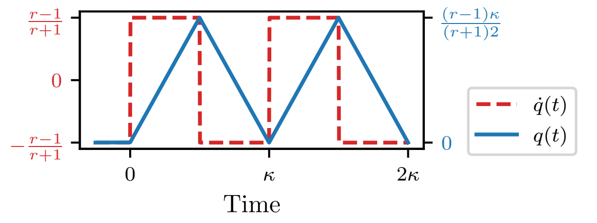
We assume a traffic flow is attempting to utilize the varying capacity interface fully. A reasonable strategy for achieving 100% utilization is to send packets at a constant rate equal to the average rate of the link. This strategy maximizes link utilization while avoiding the need to phase-match sender rate variations with the variations of the bottleneck interface.

B. Analysis

The average service rate, C_{avg}, of the square wave is the average of the two rates C and C/r, calculated as shown in equation 3.



(a) Time per arrival and per service



(b) Queue depth and queue growth rate

Fig. 3: A square-wave service pattern with constant arrival rate at the average service rate, and plots of the resulting $q(t)$ and $\dot{q}(t)$.

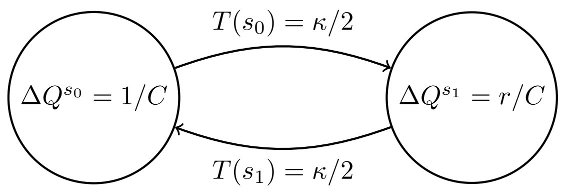


Fig. 4: The example model. The probabilities $P(s_0, s_1) = P(s_1, s_0) = 1$ have been left out to improve readability.

$$C_{avg} = \frac{C + \frac{C}{r}}{2} = \frac{rC + C}{2r} = \frac{(r+1)C}{2r} \quad (3)$$

The average service time when the interface is running at 100% capacity is proportional to $1/C$, as shown by equation 4. For a general ΔQ generative model, the method described in section III-B can be used to calculate the average service rate and consequently also the constant arrival rate that achieves 100% utilization.

$$A(t) = avg(S(t)) = \frac{1}{C_{avg}} = \frac{2r}{(r+1)C} = \frac{2r}{r+1}\frac{1}{C} \quad (4)$$

Figure 3(a) shows per-packet arrival and service time for the case where time per arrival is $2r/(r+1)$, which is the average value of the service time at 100% utilization. Given that the values for the arrival and service times are expressed as multiples of $1/C$, we can compute values for the length of the queue at the interface. The queue grows or shrinks at a rate defined by equation 5. Because the packet arrival rate is kept constant, the length of the queue measured in seconds is the integral of equation 5 over time, with the caveat that queue delay cannot become negative. Equation 6 shows how to compute the value of $q(t)$ assuming $q(0) = 0$ and taking into account that the queue depth cannot be negative. Figure 3(b) shows values for the derivative of the queue depth, $\dot{q}(t)$, and the queue depth, $q(t)$. Because the factor $1/C$ is present in both the numerator and denominator of equation 5, the growth rate of the queue is independent of C. However, for this analysis to be a good approximation to real networks where data is transmitted as packets where each packet uses a single rate, we require that the duration of single packet transmissions at the slowest rate is much smaller than $\frac{\kappa}{2}$.

$$\dot{q}(t) = \frac{\text{Service time} - \text{Time per arrival}}{\text{Service time}} = \frac{S(t) - A(t)}{S(t)} \quad (5)$$

$$q(t) = \int_0^t \frac{S(t) - A(t)}{S(t)}\, dt - \min(0, \min \int_0^t \frac{S(t) - A(t)}{S(t)}\, dt) \quad (6)$$

For the case of a square wave service rate and 100% link utilization, the maximum queue size is equal to the size of the green area (or $1/r$ the size of the red area) in figure 3(a), as expressed in equation 7.

$$max(q) = \left(\frac{2r}{r+1} - 1\right)\frac{\kappa}{2} = \frac{(r-1)\kappa}{(r+1)2} \quad (7)$$

Figure 5 shows the peak delay at 100% utilization ($\rho = 1$) for different values of $f = 1/\kappa$ and r. The plot represents the minimum delay that must be accepted for a constant-rate arrival pattern to achieve 100% utilization of the modeled interface with no packet losses. It is possible to achieve lower delays, but that requires either lowering utilization, accepting packet losses, or a combination of these.

If we select a target peak delay, $\lceil q \rceil$, and insist on no packet loss, how much must we reduce utilization to achieve our goals? Inspecting figure 6(b) we can see that in the interval from time 0 to $\kappa/2$, our target peak delay means we must limit the slope at which the queue grows. Assuming the queue is empty at time 0, we can derive equation 8. Observe that $q(t)$ is a straight line in the interval 0 to $\kappa/2$, and that $S(t) = r$ in the interval 0 to $\kappa/2$. Since the derivative $\dot{q}(t)$ is given by $\frac{S(t)-A(t)}{S(t)}$, equation 8 follows.

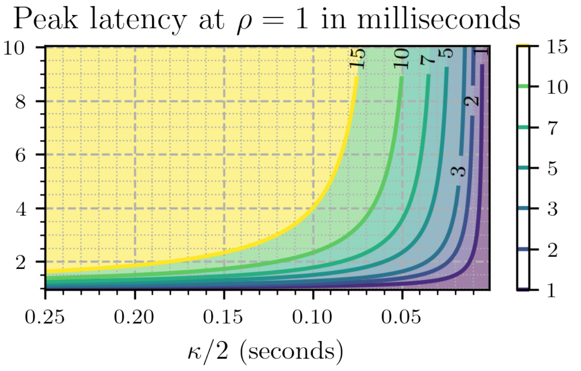


Fig. 5: Peak queuing delay as a function of the frequency of capacity changes for different values of r with 100% link utilization

$$\lceil q \rceil = \frac{\kappa}{2} * \frac{S(t) - A(t)}{S(t)} = \frac{\kappa(r - A(t))}{2r} \quad (8)$$

Now, we solve equation 8 for $A(t)$, and arrive at equation 9. We can now compute the maximum average value for $A(t)$ which, given r and κ, keeps the peak delay below a certain target value $\lceil q \rceil$ (here we assume $\lceil q \rceil$ is less than the maximum $q(t)$ of equation 7).

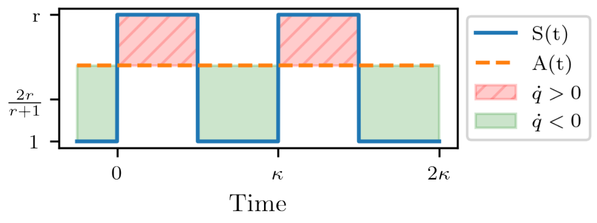
$$A_{\lceil q \rceil}(t) = r(1 - 2\lceil q \rceil f) \quad (9)$$

Reducing the arrival rate below the average service rate will reduce link utilization, but how much? We now derive an expression for the link utilization as a function of $\lceil q \rceil$. Let link utilization be defined as the achieved fraction of maximum average throughput. Since throughput is inversely proportional to the per-packet processing time, utilization can be expressed in terms of the average per-packet arrival times $A(t)$ and $A_{\lceil q \rceil}(t)$. Equation 10 gives the link utilization fraction as a function of r, f and $\lceil q \rceil$.

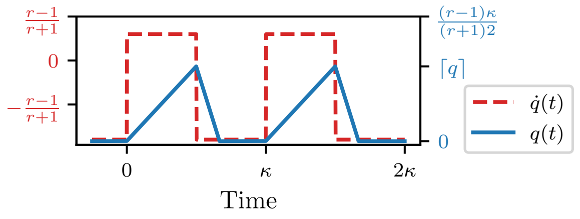
$$U_{\lceil q \rceil} = \frac{A(t)}{A_{\lceil q \rceil}(t)} = \frac{(2r)/(r+1)}{r(1 - 2\lceil q \rceil f)} = \frac{2}{(1 - 2\lceil q \rceil f)(r+1)} \quad (10)$$

Figure 7 shows the value of $U_{\lceil q \rceil}$ for different values of r and $\lceil q \rceil$. As we can see, the link utilization drops rapidly when the frequency of capacity changes drops below the threshold where $\lceil q \rceil$ becomes smaller than the $max(q)$ of equation 7. Conversely, achieving high link utilization when $\lceil q \rceil$ is small puts requirements on the frequency of capacity changes.

For a general ΔQ generative model, it may not be possible to derive an equation for the trade-off between utilization and $\lceil q \rceil$. It is, however, possible to run a Monte Carlo simulation at different utilization levels and plot the resulting peak queuing delays. Figure 8 shows both analytical results and the results of a Monte Carlo simulation for the case where $r = 2$ and $f = 100$ Hz. The analytical results are derived by solving equation 10 for $\lceil q \rceil$ and inserting values for r, f and $U_{\lceil q \rceil}$. The



(a) Time per arrival and per service



(b) Queue depth and queue growth rate

Fig. 6: A square-wave service pattern with constant arrival rate at the average service rate, and plots of the resulting $q(t)$ and $\dot{q}(t)$ when delay is bounded at $\lceil q \rceil$

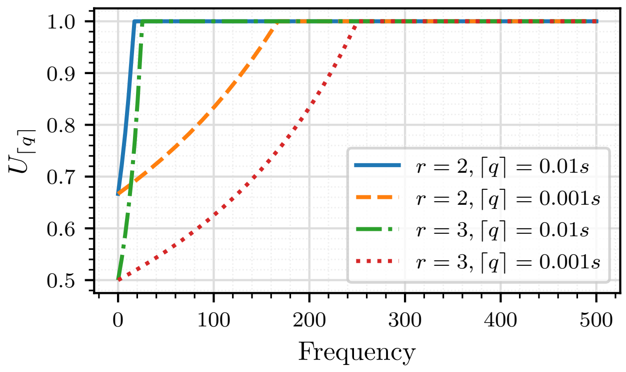


Fig. 7: Maximum utilization that keeps peak delay below $\lceil q \rceil$ as a function of square wave frequency

simulation is performed using a discrete event simulator. Simulations are run for two seconds of simulated time, accounting for 200 periods at the selected 100Hz frequency. The peak packet rate is set to 81000 packets a second in the simulator, corresponding to a typical 1 Gb/s link with maximum packet size of 1500 bytes. Since this particular model is deterministic, there is no need to run multiple random seeds, but this would obviously be necessary in the general case.

V. Discussion

We can sum up the results of our analysis as follows: We have derived expressions to quantify the relationship between delay and utilization when the service rate varies as a square wave, and no packets are lost. Additionally, we

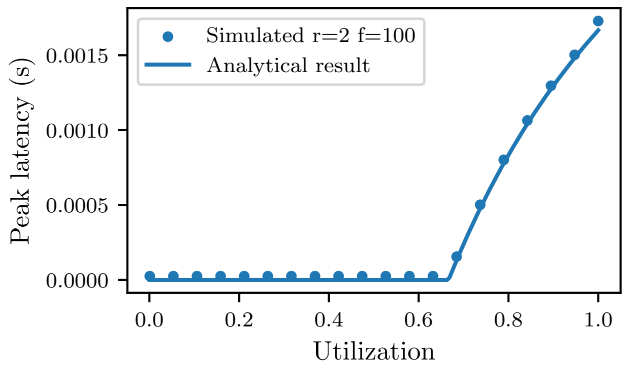


Fig. 8: Peak latency as a function of utilization assuming no packet loss

have shown that a Monte Carlo simulation-based approach applicable to all ΔQ generative models produces the same results as mathematical analysis of the square wave model.

We believe the square wave model is a reasonable first-order approximation to the behavior of network interfaces where transmit rate is uncertain. There are many examples of bearers subject to capacity variations on many different timescales in the current Internet. WiFi rates vary based on channel conditions, and effective PON and DOCSIS rates vary based on how many time slots a user is assigned. Both the channel conditions and the time-slot allocation can change in 4G, 5G, and in WiFi networks since WiFi 6. There is no reason to believe a square wave service pattern is particularly common in these network technologies. However, since the analysis is applicable to single periods of duration κ, the results may be reasonable approximations for temporary capacity drops of duration $\kappa/2$. This makes our results a potentially helpful point of reference for designing queuing policies and tuning end-to-end congestion control algorithms running over these technologies.

Congestion control algorithms such as TCP [1] or QUIC [2] can react and adapt to changes in network capacity. These algorithms observe end-to-end behavior such as packet loss and latency, make inferences about the capacity of the end-to-end connection, and adjust their transmission rate accordingly. One might argue that the presence of congestion control algorithms invalidates our assumption of a constant arrival rate at the average service rate. However, because of signaling delay, these algorithms cannot react immediately to changes in network capacity. We believe our conclusions about the coupling between queuing delay, packet loss, and utilization are good approximations when $\kappa/2$ is smaller than the round-trip time, even in the presence of end-to-end congestion controllers. Round-trip times in the Internet are often on the order of tens to hundreds of milliseconds, and so for the purposes of analyzing Internet applications, results for frequencies above 10 Hz seem reasonable.

VI. Conclusion

We have defined a novel class of models for reasoning about variations in network quality and described the basic algorithms needed to work with the models. Our results show that the models capture features that are important for buffer sizing. ΔQ generative models define a general format for describing network capacity variations. Such a format may serve as the interface between real-world networks and theoretical modeling and analysis because the model class can facilitate stochastic descriptions of recorded network traces. More work is needed to realize this possibility. Future work will explore machine learning of ΔQ generative models, and implement emulation of network behavior in a testbed based on ΔQ generative models. Learned models may pick up on sources of capacity variation that are not typically modeled in simulation studies, and thereby add to the robustness of results from simulation and emulation studies.

References

[1] V. Jacobson, "Congestion avoidance and control," in *Symposium Proceedings on Communications Architectures and Protocols, SIGCOMM 1988*, New York, New York, USA: Association for Computing Machinery, Inc, Aug. 1988, pp. 314–329, ISBN: 0897912799. DOI: 10.1145/52324.52356. [Online]. Available: http://portal.acm.org/citation.cfm?doid=52324.52356.

[2] A. Langley, A. Riddoch, A. Wilk, *et al.*, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, vol. 14, 2017. DOI: 10.1145/3098822. [Online]. Available: https://doi.org/10.1145/3098822.3098842.

[3] J. Kennedy, G. Armitage, and J. Thomas, "Household bandwidth and the 'need for speed' Evaluating the impact of active queue management for home internet traffic," *Australian Journal of Telecommunications and the Digital Economy*, vol. 5, no. 2, pp. 113–130, Jun. 2017, ISSN: 22031693. DOI: 10.18080/ajtde.v5n2.99.

[4] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet: Networks without effective AQM may again be vulnerable to congestion collapse," *Queue*, vol. 9, no. 11, pp. 40–54, Nov. 2011, ISSN: 15427749. DOI: 10.1145/2063166.2071893.

[5] B. Spang, S. Arslan, and N. McKeown, "Updating the Theory of Buffer Sizing," *Performance Evaluation*, vol. 151, Sep. 2021, ISSN: 01665316. DOI: 10.48550/arxiv.2109.11693. [Online]. Available: https://arxiv.org/abs/2109.11693v1.

[6] P. Goyal, M. Alizadeh, and T. E. Anderson, "Optimal Congestion Control for Time-varying Wireless Links," Feb. 2022. [Online]. Available: http://arxiv.org/abs/2202.04321.

[7] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," *Queue*, vol. 14, no. 5, pp. 20–53, Oct. 2016, ISSN: 1542-7730. DOI: 10.1145/3012426.3022184.

[8] B. Briscoe, K. D. Schepper, M. Bagnulo, and G. White, "Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Architecture," Internet Engineering Task Force, Tech. Rep. draft-ietf-tsvwg-l4s-arch-10, Jul. 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-l4s-arch-10.

[9] K. Nichols and V. Jacobson, "Controlling queue delay," *Queue*, vol. 10, no. 5, May 2012, ISSN: 15427730. DOI: 10.1145/2208917.2209336.

[10] R. Pan, P. Natarajan, C. Piglione, *et al.*, "PIE: A lightweight control scheme to address the bufferbloat problem," in *IEEE International Conference on High Performance Switching and Routing, HPSR*, 2013, pp. 148–155. DOI: 10.1109/HPSR.2013.6602305.

[11] T. Høiland-Jørgensen, D. Täht, J. Morton, T. Hoiland-Jorgensen, D. Taht, and J. Morton, *Piece of CAKE: A Comprehensive Queue Management Solution for Home Gateways*, Sep. 2018. DOI: 10.1109/LANMAN.2018.8475045. [Online]. Available: http://arxiv.org/abs/1804.07617.

[12] D. G. Kendall, "Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain," *The Annals of Mathematical Statistics*, vol. 24, no. 3, pp. 338–354, Sep. 1953, ISSN: 0003-4851. DOI: 10.1214/aoms/1177728975. [Online]. Available: http://projecteuclid.org/euclid.aoms/1177728975.

[13] Y. Liu and W. Whitt, "The Gt/GI/st+GI many-server fluid queue," *Queueing Systems*, vol. 71, no. 4, pp. 405–444, Mar. 2012, ISSN: 15729443. DOI: 10.1007/S11134-012-9291-0/FIGURES/7.

[14] M. Welzl, "Network congestion control : managing Internet traffic," p. 263, 2005.

[15] H. Haile, K. J. Grinnemo, S. Ferlin, P. Hurtig, and A. Brunstrom, "End-to-end congestion control approaches for high throughput and low delay in 4G/5G cellular networks," *Computer Networks*, vol. 186, p. 107 692, Feb. 2021, ISSN: 13891286. DOI: 10.1016/j.comnet.2020.107692.

[16] A. Srivastava, F. Fund, and S. S. Panwar, "An experimental evaluation of low latency congestion control for mmWave links," *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2020*, pp. 352–357, Jul. 2020. DOI: 10.1109/INFOCOMWKSHPS50562.2020.9162881.

[17] Broadband Forum, "TR-452.1 Quality Attenuation Measurement Architecture and Requirements," Tech. Rep., 2020. [Online]. Available: https://www.broadband-forum.org/download/TR-452.1.pdf.

[18] S. Haeri, P. Thompson, N. Davies, P. Van Roy, K. Hammond, and J. Chapman, "Mind Your Outcomes: The ΔQSD Paradigm for Quality-Centric Systems Development and Its Application to a Blockchain Case Study," *Computers*, vol. 11, no. 3, p. 45, Mar. 2022. DOI: 10.3390/COMPUTERS11030045.