Asymptotic Analysis of Real-Time Queues

John Lehoczky

Department of Statistics

Carnegie Mellon University

Pittsburgh, PA 15213

Co-authors: Steve Shreve, Kavita Ramanan, Lukasz Kruk, Bogdan Doytchinov, Calvin Yeung, and Jeffery Hansen.

Outline

- Thoughts about Steve Shreve
- Background on real-time queues
- Real-Time Queueing Theory
- Extensions

Thoughts about Steve Shreve

It has been my great good fortune to have Steve Shreve as a colleague for $\approx$ 35 years. He is:

- A distinguished mathematician and a wonderful research collaborator
- An outstanding teacher-educator
- A great colleague and resource who is willing to engage on others' problems
- A devoted advisor and mentor
- Attains the highest standards of integrity and ethics.

Background: 1

- Traditional queueing theory focuses on:
  - Stability of the system
  - The efficiency of the use of the service facility (e.g. the fraction of time it is busy) and
  - The experience of the customers (e.g. their waiting time or the length of the queue they join).
- There is a special category of queueing systems called real-time queues. Here, the tasks requiring service have deadlines by which the service must be completed. These systems have special performance metrics that focus on the fraction of tasks whose deadline was met.

Background: 2

- When tasks have deadlines and they must share a processing resource, then traditional queueing disciplines like FIFO or processor sharing should not be used since they pay no attention to deadlines. Deadline-sensitive scheduling policies like earliest deadline first must be used.
- There are a number of examples of real-time systems in which tasks have timing requirements. They especially occur in fairly autonomous systems with weight and power constraints, e.g. Mars rover, GPS satellites, autonomous aircraft, submarines, etc. in which one needs to minimize the number of computer systems on board and use them as efficiently as possible.

Background: 3

- The first widely recognized work on such systems was a 1973 paper by Lui and Layland in the *JACM*, written when C.L. Liu was a summer student at the Jet Propulsion Lab (JPL).
- In this work, a single computer was used to process a set of periodic tasks (to model worst case conditions). The question was to characterize the set of tasks that could be processed with no deadline being missed over $[0, \infty)$.
- Formalizing this: assume there are n periodic tasks, each with a worst case computation time, C_i, a period, T_i, a (relative) deadline D_i and an initial arrival time, I_i. Thus, jobs from such a task would arrive at $I_i, I_i + T_i, I_i + 2T_i$, etc. All C_i units of work arriving at $I_i + kT_i$ must be completed by $I_i + (k+1)I_i$.

Background: 4

- LL considered two broad classes of scheduling policies (queue disciplines): fixed priority and dynamic priority. Under fixed priority, each of the n tasks would be assigned a unique priority, and anytime a higher priority task arrived while a lower priority task was being processed, the higher priority task would preempt (at no cost in time).
- LL asked the question: for all values of n, $\{(I_i, C_i, T_i), 1 \leq i \leq n\}$, what is the optimal scheduling policy (among fixed or dynamic classes), and given the optimal policy, how can it be determined whether or not a scheduling policy can satisfy the deadline requirements for all tasks over $[0, \infty)$.

Background: 5

For fixed priority scheduling algorithms of periodic tasks, LL proved:

- The optimal scheduling algorithm is the rate monotonic scheduling algorithm. For this algorithm, if $T_i < T_j$, then task i is given higher priority than task j, independent of C_i, C_j
- The worst case phasing is the Critical Instant, that is $I_j = 0,\ 1 \leq j \leq n$.
- If $U = U_1 + \ldots + U_n < n(2^{1/n} - 1)$, then all deadlines will be met (we say the task set is schedulable).
- $\lim_{n\to\infty} n(2^{1/n} - 1) = \log(2) = .693$.

Background: 6

For dynamic scheduling algorithms, LL proved that earliest deadline first was optimal. Other scheduling algorithms also can be used successfully (e.g. least slack first).

- LL proved that any task set for which $U = U_1 + \ldots + U_n \leq 1$ will guarantee that all task deadlines can be met using the earliest deadline first algorithm.

Background: 7

- These results for the periodic task set case with hard deadlines has been greatly generalized and includes such factors as:
  - Deadlines different from periods
  - Context switching times
  - Precedence constraints among tasks
  - Task synchronization (i.e. some tasks need special resources like a communication bus and cannot be interrupted while using that resources by another task needing that resource, even if it has higher priority).

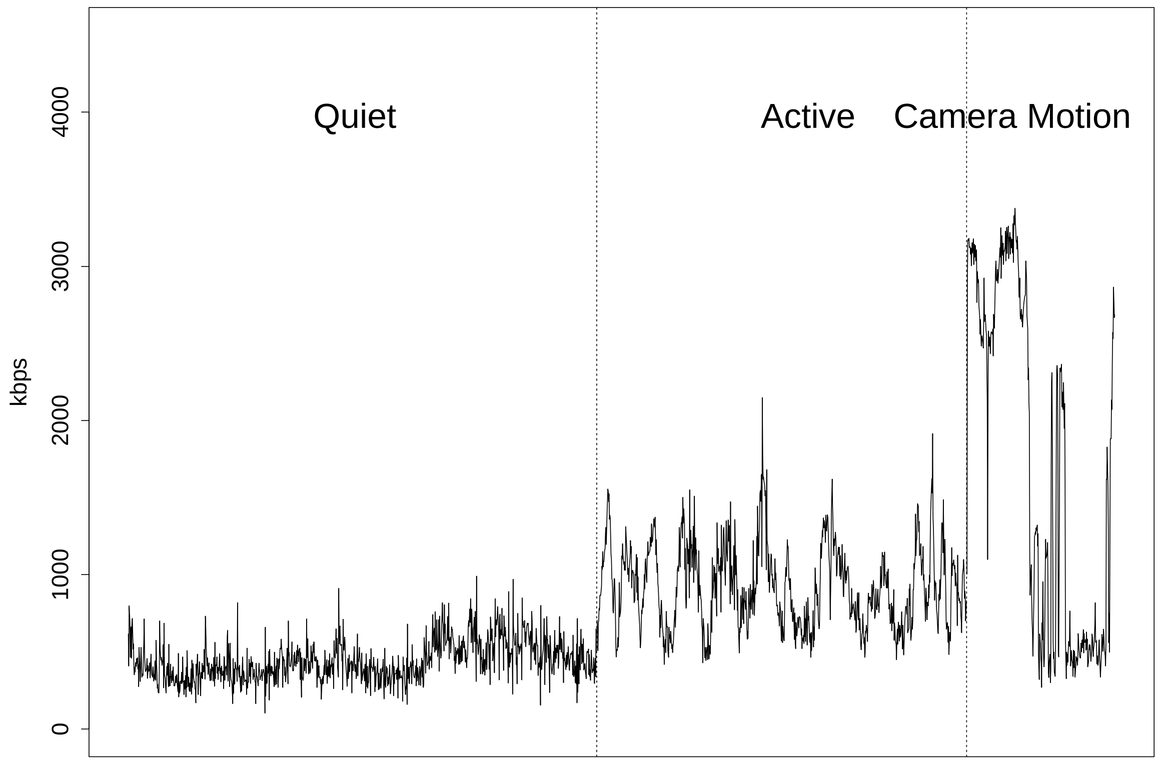
Background: 8

- For all deadlines to be met, LL assumed a deterministic environment, e.g. worst case execution times and interarrival times, and sufficiently long deadlines.
- There are applications that exhibit substantial variability either in terms of resource requirements or arrival times or both.
- In such applications, the worst case utilization (considering the minimum interarrival time and the maximum computation time) can be much larger than the average case utilization.

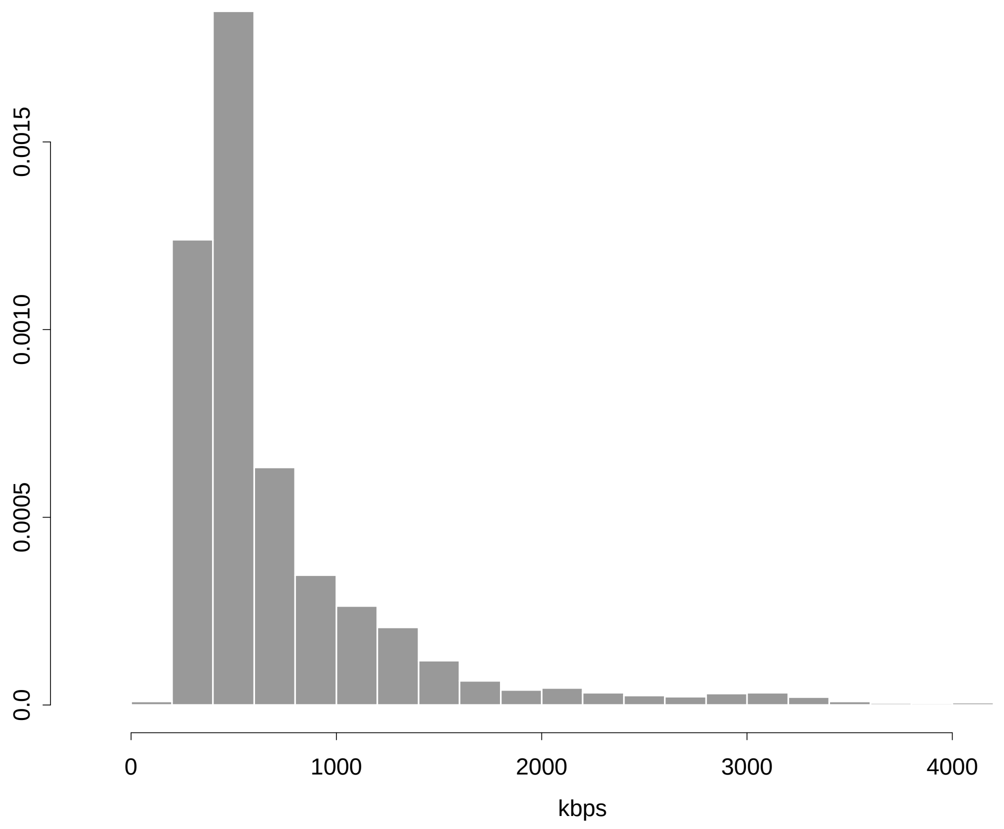
Background: 9

- Deadlines can be guaranteed, but the effective resource utilization can be very low, especially if there is a large difference between the worst-case utilization and the average-case utilization.
- Moreover, hard deadlines may be an unnecessarily restrictive constraint on the system. Many applications can miss some deadlines and still meet user requirements, provided that the missed deadlines do not occur over a short period of time, resulting in a “blackout” of service.

Video-Conference Bandwidth Usage



Video-Conference Bandwidth Usage



The Dilemma

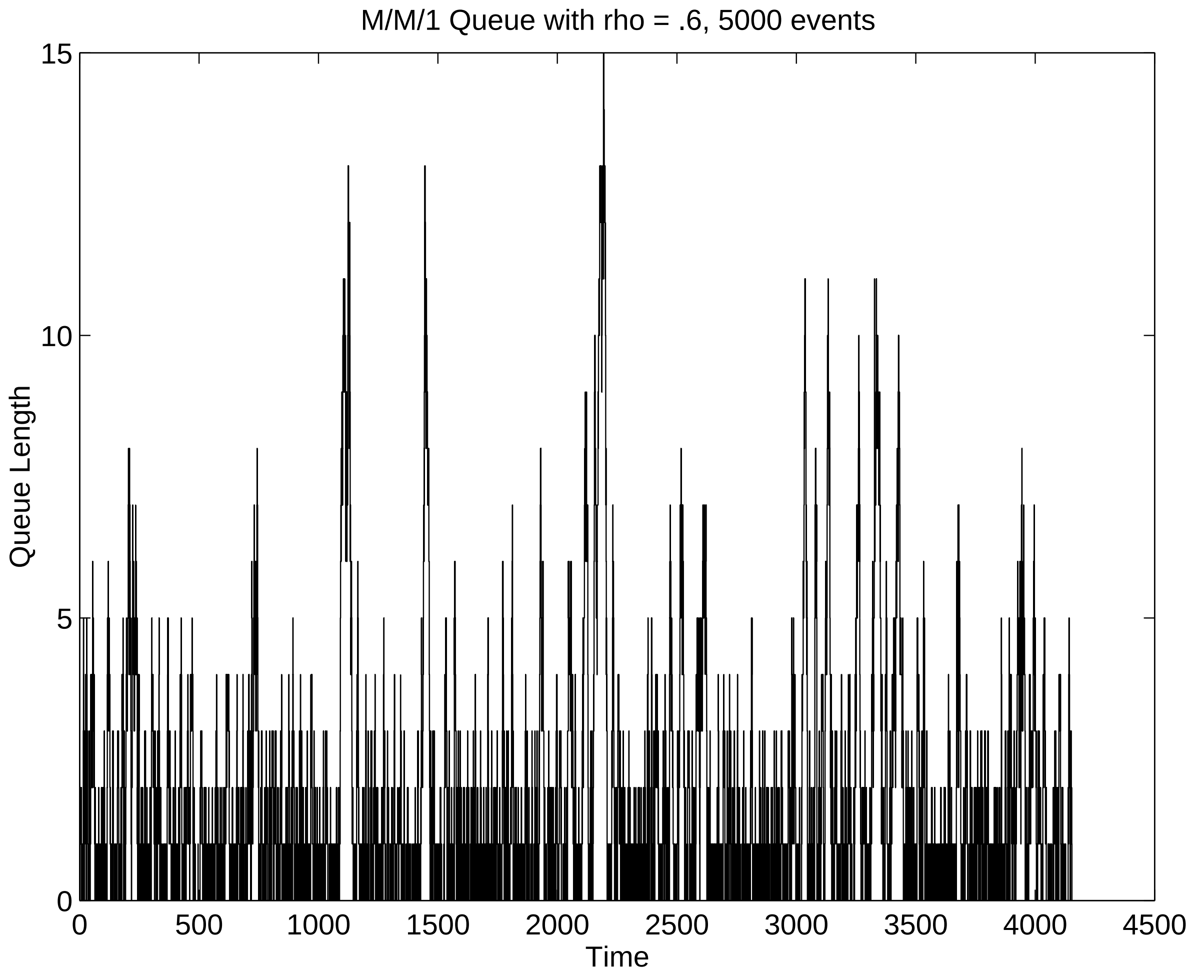
We want to create a theory that:

- permits the stochastic behavior associated with queueing models,
- takes individual task timing requirements into account.

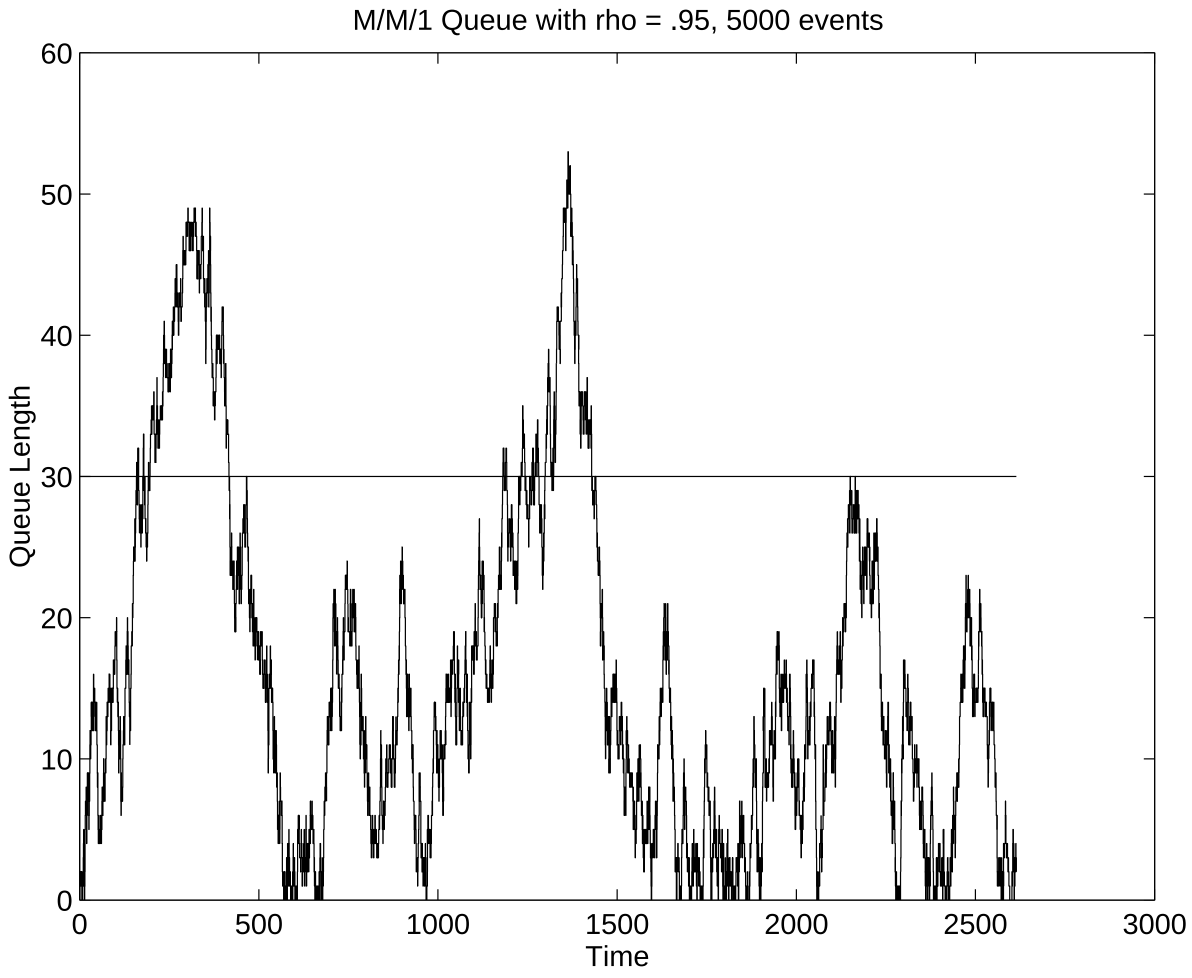
Such models will have so much detail that they will be unworkable analytically, and we will be back to simulation.

Fortunately, some ideas in heavy traffic queueing theory can help, since timing requirements will be met in light-moderate traffic.

M/M/1 Queue, $\rho = .6$, 5000 Events



M/M/1 Queue, $\rho = .95$, 5000 Events



Heavy Traffic Limit

The GI/G/1 queueing system, in heavy traffic ($\rho = 1 - \frac{\gamma}{\sqrt{n}}$), with time scaled by n and space scaled by $\sqrt{n}$ both the queue length and the workload processes converge as $n \to \infty$ to a

Drifted, Reflected Brownian motion.

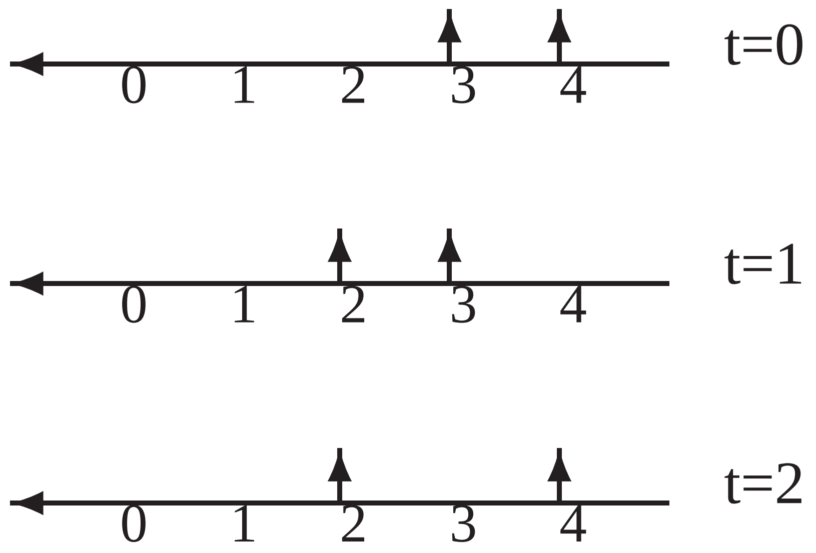
The parameters of this process depend only upon the mean and variance of the arrival and service processes. The drift is $-\gamma$ and the variance (Peterson) $\sum_{k=1}^{K} \lambda_k m_k^2 (C_{A_k}^2 + C_{S_k}^2)$.

This theory extends to the network case.

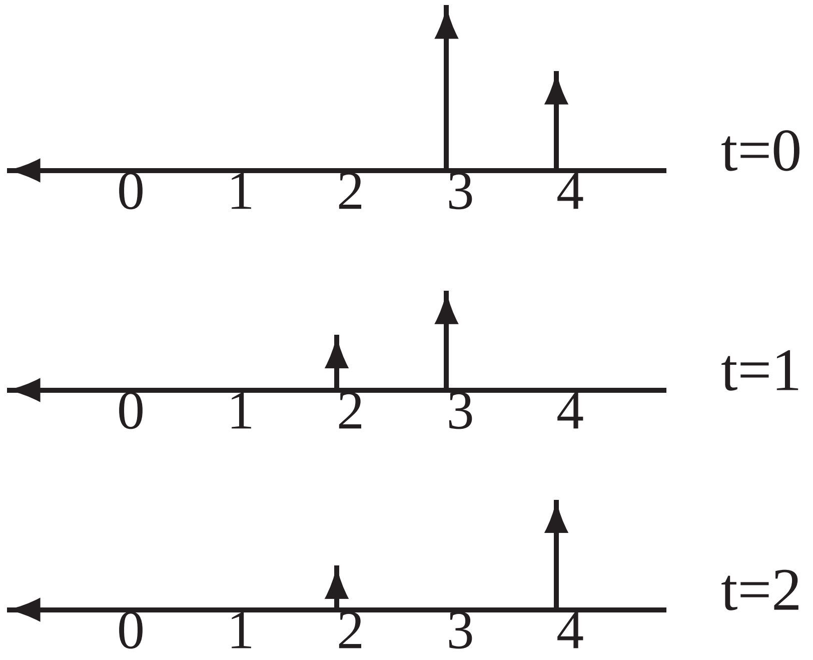
Real-Time Queueing Theory - RTQT

- Heavy traffic queueing theory offers a method to determine standard queueing performance measures using only moment assumptions concerning the basic arrival and service processes.
- But, what does this have to do with real-time systems in which we need to determine the ability of the system to meet task timing requirements?
- We need to introduce a dynamic variable for each task in the system, its lead-time = time until deadline.
- This becomes a high-dimensional system, but perhaps it simplifies in heavy-traffic.

Lead-Time Measures



Workload Measures



Real-Time Queueing: Assumptions

For the one flow, one node case:

- Renewal process arrivals, $(F_a, \frac{1}{\lambda}, \sigma_a)$.
- Independent service times, $(F_s, \frac{1}{\mu}, \sigma_s)$.
- Tasks have independent random deadlines with cdf G.
- Arrivals, services, deadlines are independent.

Let $\rho = \frac{\lambda}{\mu},\ \theta = 2(1-\rho)\mu/(\rho(C_a^2 + C_s^2))$

Real-Time Queueing: The Ideas 1

If there are $Q(t)$ tasks in the queue at time t, then the lead-time vector is $(L_1(t), \ldots, L_{Q(t)}(t))$ which changes with time.

A simple RT queueing system has state $\{(Q(t), (L_1(t), \ldots, L_{Q(t)}(t))), t \geq 0\}$. Such a high dimensional process is difficult to analyze, even in simple circumstances.

We study these models in heavy traffic (traffic intensity near 1). The key idea is to separate into two components: $\{Q(t), t \geq 0\}$ and $\{(L_1, \ldots, L_{Q(t)}), t \geq 0\}$ given $\{Q(t), t \geq 0\}$.

Real-Time Queueing: The Ideas 2

In heavy traffic ($\lambda^{(n)} = \lambda(1 - \frac{\gamma}{\sqrt{n}})$, $\mu^{(n)} = \lambda$, $\rho^{(n)} = 1 - \frac{\gamma}{\sqrt{n}}$),

- the scaled queue length, $\{\frac{Q(nt)}{\sqrt{n}}, t \geq 0\}$ and the scaled workload, $\{\frac{W(nt)}{\sqrt{n}}, t \geq 0\}$, converge to drifted, reflected Brownian motion processes.

- the lead-time profile converge to deterministic function of the queue length (or the workload)

  The deterministic function depends upon:

  – The initial task deadline probability distribution.

  – The scheduling policy.

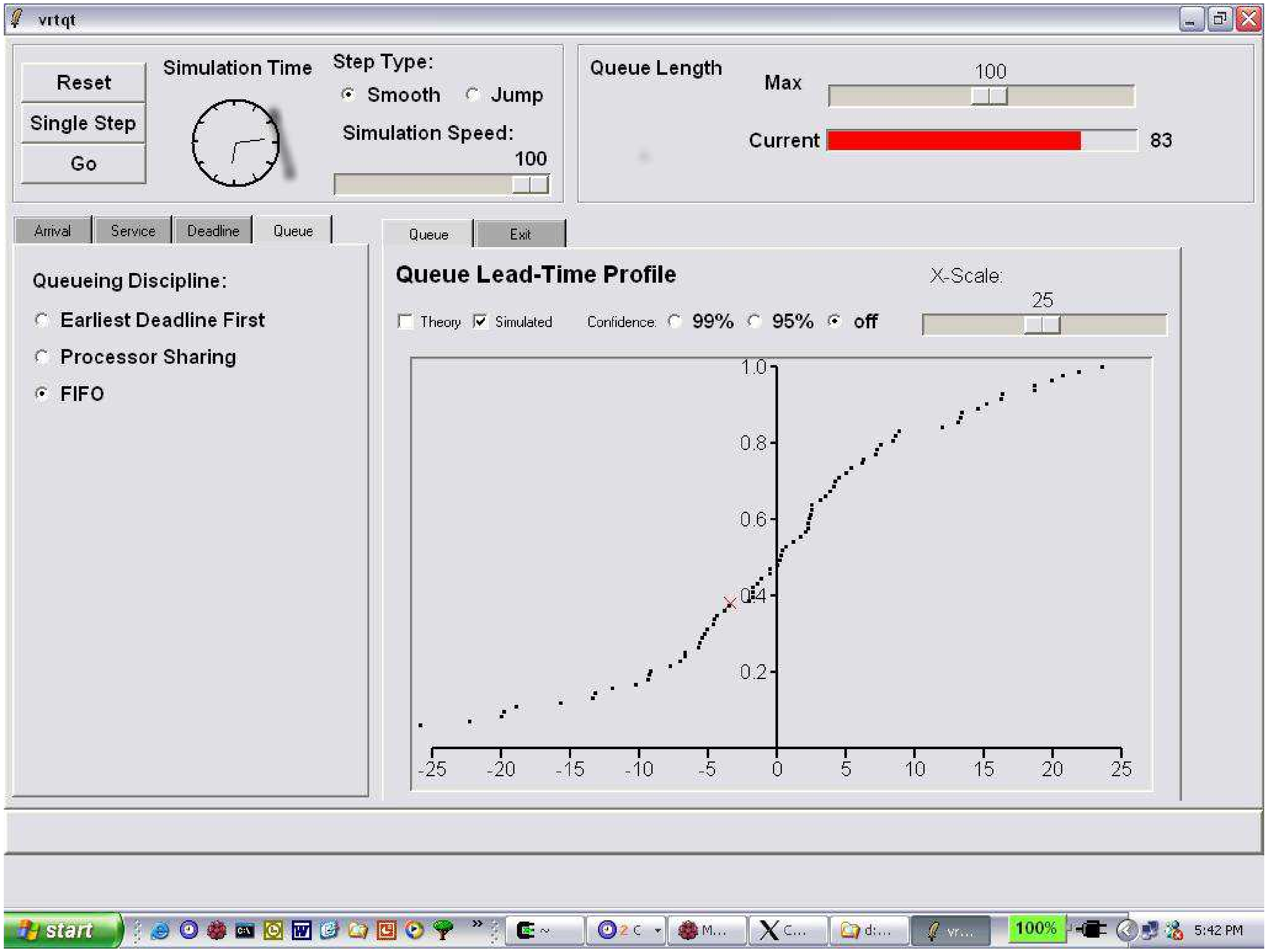
Real-Time Queueing: The Ideas 3

- Using the deterministic approximation, each queue length (or workload value) has an associate a lead-time profile.
- Lateness occurs when the left edge of the profile reaches 0.
- The left edge of the profile reaches 0 at a computable queue length (or workload value). Thus, the event of task lateness is associated with the queue length (or workload) being longer than a certain critical value.
- So we can express the probability of lateness (and other performance measures) in terms of the queue length (or workload) distribution, which can be computed from Brownian motion facts.

VRTQT Demonstration

Developed by Jeffery Hansen of the CMU Software Engineering Institute

Processor Sharing: Exp. Deadlines



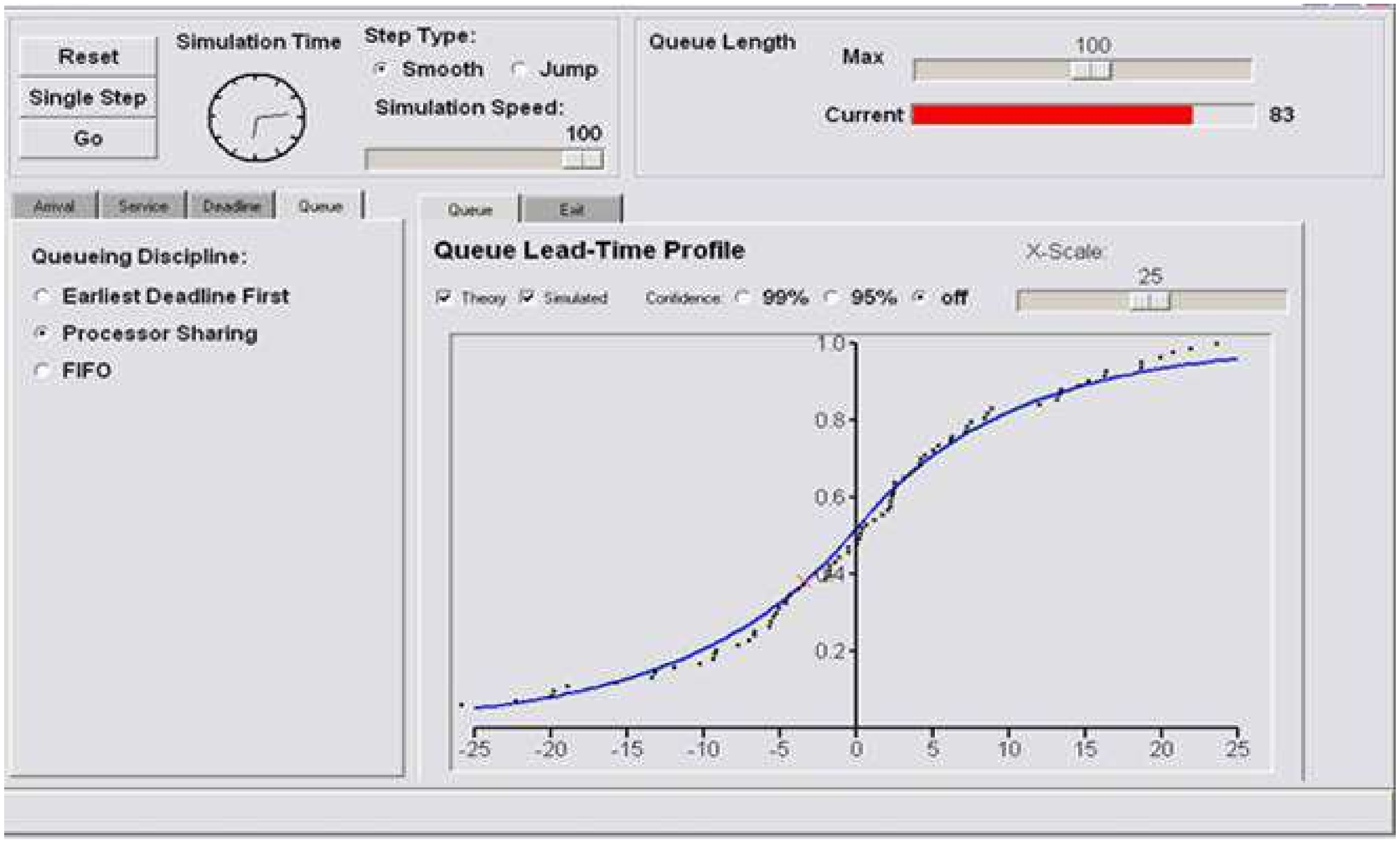
Step Type: [✓ Smooth] Jump

Queueing Discipline: Earliest Deadline First / Processor Sharing / [✓ FIFO]

Theory / [✓ Simulated]

Confidence: 99% / 95% / [✓ off]

Processor Sharing: Exp. Deadlines



Step Type: [✓ Smooth] Jump

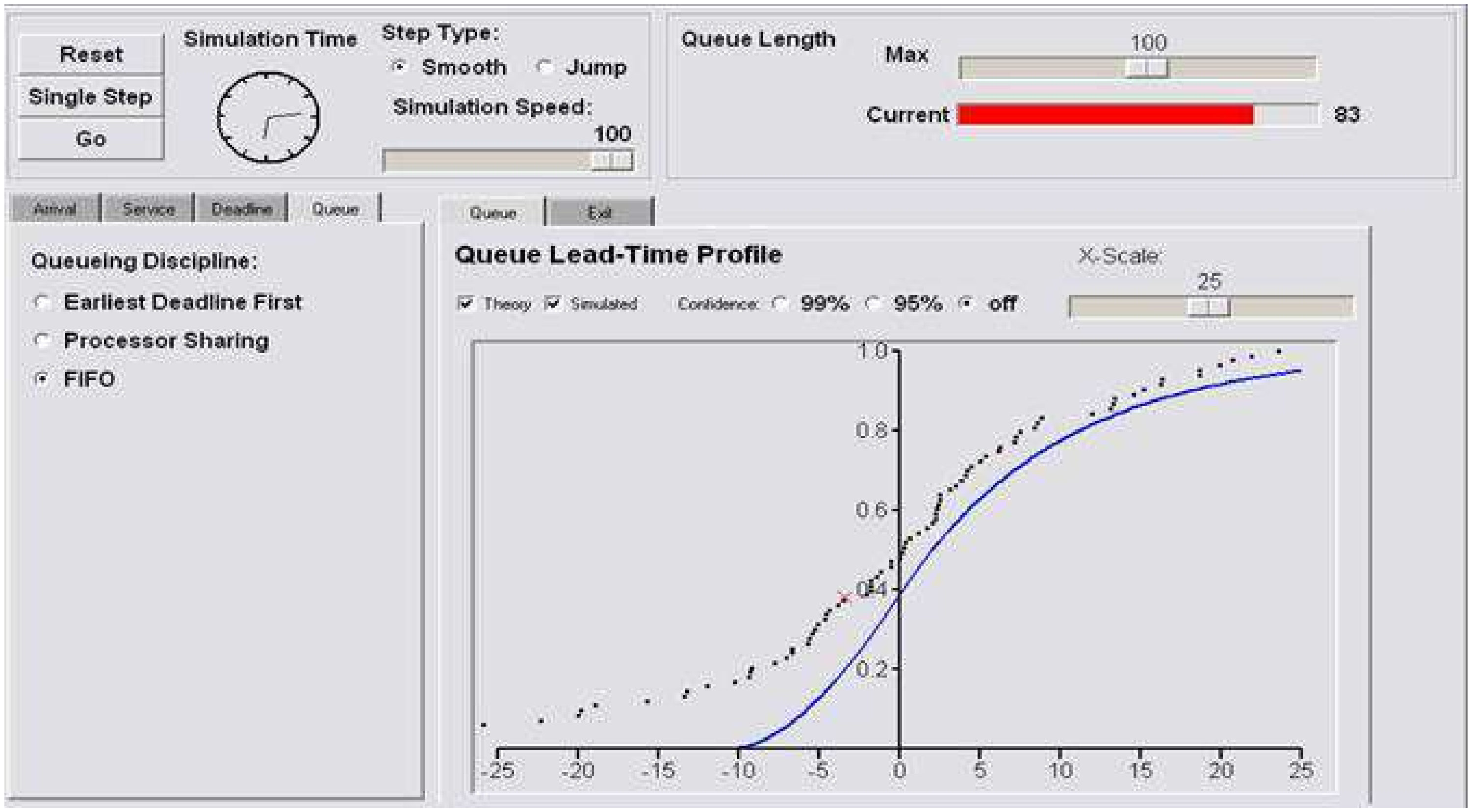
Queueing Discipline:

Earliest Deadline First

[✓ Processor Sharing]

FIFO

Processor Sharing: Exp. Deadlines



Step Type: [✓ Smooth] Jump

Simulation Speed: 100

Queue Length

Max 100

Current 83

Queueing Discipline:

Earliest Deadline First

Processor Sharing

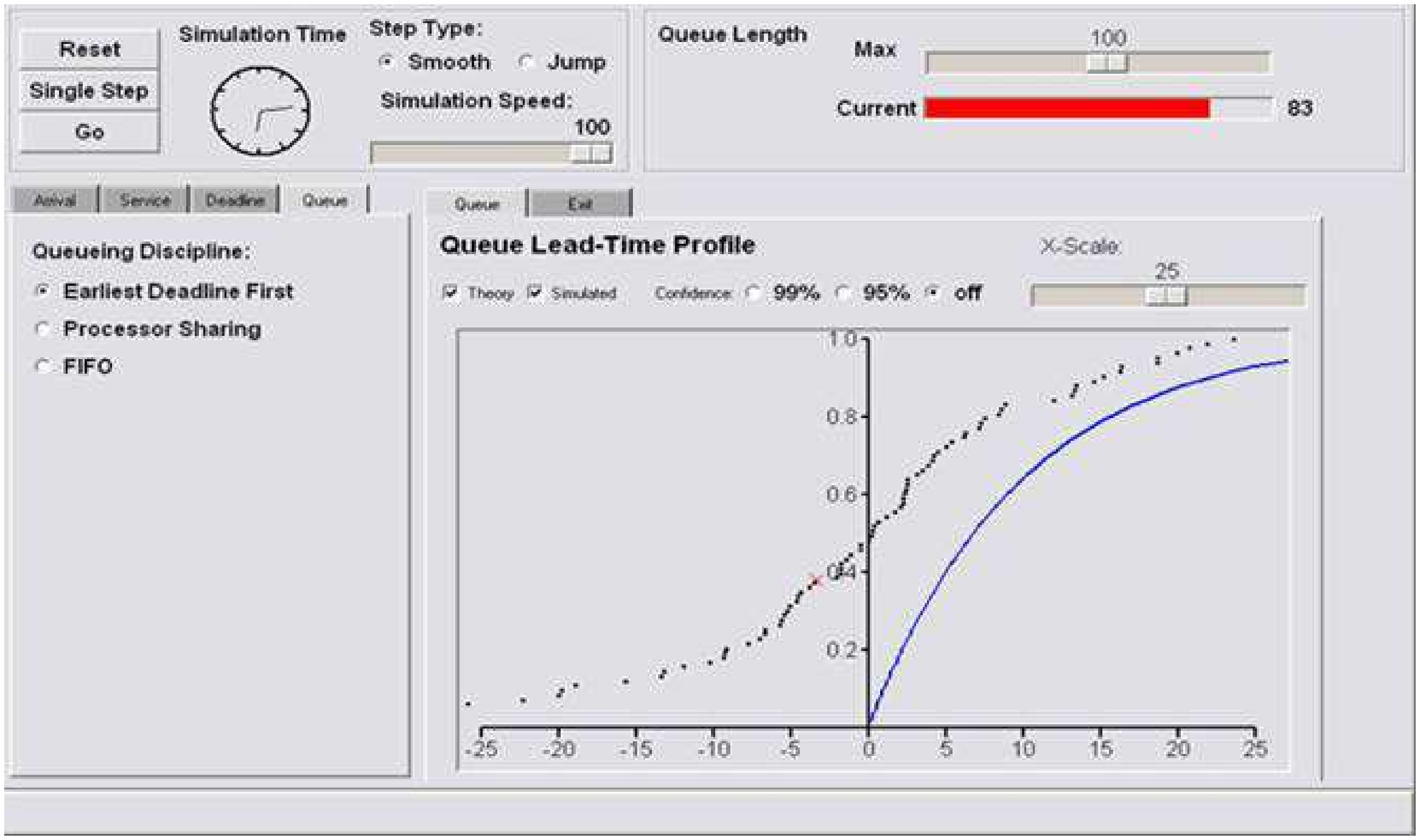
[✓ FIFO]

X-Scale: 25

[✓ Theory] [✓ Simulated]

Confidence: 99% 95% [✓ off]

Processor Sharing: Exp. Deadlines



Heavy Traffic Analysis: 1

- Queue length process, $Q^{(n)}(t)$,
- Workload process, $W^{(n)}(t)$,
- Lead-time measure, $\mathcal{Q}^{(n)}(t)(B)$,

  Number of customers in queue at t having lead-times in B at t.
- Workload measure, $\mathcal{W}^{(n)}(t)(B)$,

  Work at t from customers in queue having lead-times in B at t.
- Frontier,

  $F(t) \approx$ largest lead-time of any customer ever in service.

Heavy Traffic Analysis: 2

- $\hat{W}^{(n)}(t) = \frac{W^{(n)}(nt)}{\sqrt{n}} \Rightarrow W^*$,
  a reflected Brownian motion with drift $-\gamma$.
- $\hat{Q}^{(n)}(t) = \frac{Q^{(n)}(nt)}{\sqrt{n}} \Rightarrow \mu W^*$,
- Limiting scaled frontier process, $F^*(t) = H^{-1}(W^*(t))$ where

$$H(y) = \int_y^\infty (1 - G(u))du.$$

Heavy Traffic Analysis: 3

- $\hat{\mathcal{W}}^*(t)(B) = \int_{B\cap[F^*(t),\infty)}(1-G(u))du.$
- $\hat{\mathcal{Q}}^*(t)(B) = \mu\hat{\mathcal{W}}^*(t)(B).$

Then

- $\hat{\mathcal{W}}^{(n)} \Rightarrow \hat{\mathcal{W}}^*.$
- $\hat{\mathcal{Q}}^{(n)} \Rightarrow \hat{\mathcal{Q}}^*.$

Task Lead-Time Profile: EDF

For M/M/1 (and GI/G/1) queueing systems using the EDF scheduling algorithm, the instantaneous lead-time profile, given in the form of a p.d.f. is

$$f(x) = \frac{(1-G(x))}{W(t)}, F(t) \leq x < \infty$$

where G is the c.d.f. of the task deadline distribution, $W(t)$ is the current queue length and $F(t)$ is the frontier, where

$$W(t) = \int_{F(t)}^{\infty} (1 - G(x))dx.$$

Calculating Lateness: 1

- The workload process behaves like a drifted, reflected Brownian motion. Hence it has an exponential(θ) equilibrium distribution, where θ involves the first two moments of the interarrival and service distribution.
- Recall that we determine the frontier, $F(t)$, through the equation:

$$W(t) = \int_{F(t)}^{\infty} (1 - G(x))dx.$$

Calculating Lateness: 2

- Lateness is associated with $F(t) \leq 0$, or equivalently

$$W(t) \geq \int_0^\infty (1 - G(x))dx = E(D).$$

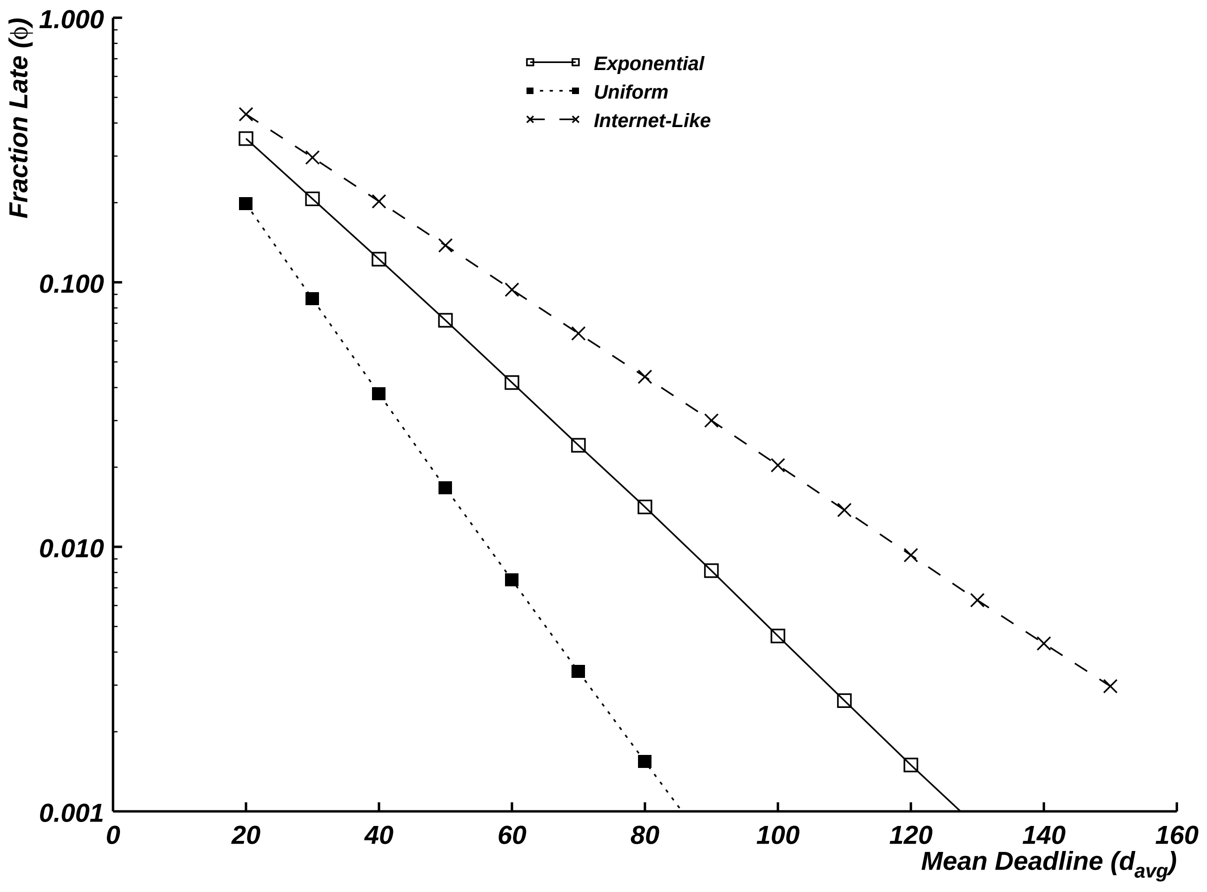
- Consequently,

$$P(\text{Lateness}) = P(W(t) \geq E(D)) = e^{-\theta E(D)},$$

or

$$\log(P(\text{Lateness})) = -\theta E(D).$$

EDF Lateness Probabilities



Extensions

- Determining the accuracy of the approximations
- Allowing customer reneging or if a customer is going to be late and counted as a deadline miss, then it is best to never serve it at all
- Feed-forward queueing networks
- Acyclic queueing networks