Delay Analysis of Switches in Heavy Traffic

Shashibhushan Borade

Recently, heavy-traffic theory has been applied for understanding behavior of delay in switches. The next section explains the heavy traffic scaling. With some toy examples, it discusses how Brownian motion arises in delay analysis and the idea of state-space collapse. Section 2 defines a switch and sketches a standard method for its stability analysis using the *fluid scaling*. For fluid scaling, the steady state behavior of switches is also discussed. Section 3 first discusses the relations between the heavy traffic scaling and the fluid scaling. Second, the results by Stolyar [1] are sketched out, where only one of the resources of a switch is in heavy traffic (say an input port or an output port). Then it discusses the results from Shah's thesis [2], where all the resources are in heavy traffic. It shows an optimal (in heavy traffic scaling) algorithm for minimizing the average delay in a switch and also compares delays of other common algorithms.

1 Brownian Motion and State-Space Collapse in Heavy Traffic

1.1 Brownian Motion in Queueing systems

Consider a single server with a single queue in discrete time. The server can provide one unit of service in each unit of time. The discrete-time arrival process is a renewal process of rate λ. Hence the inter-arrival times $\{A_i\}$ have mean $1/\lambda$. The variance of each A_i is assumed to be a^2. Let the number of arrivals up to time k be denoted by $N(k)$. Each packet is assumed to have a service requirement of $1/\mu$, i.e. $1/\mu$ time of the server. This constant packet length assumption is not necessary for the results, but it shortens the equations. By heavy traffic, we will mean that the rate at which work arrives is same as the server capacity. In the current context, it means that $\lambda/\mu = 1$. Finally, assume that the system starts at time 0 with empty queues. This sub-section is devoted to analyzing the delay behavior of this system.

Imagine that the server keeps serving even when the queue is empty, thus the remaining work in the queue could be negative. In other words, the remaining work in this imaginary system performs a random walk (not necessarily with i.i.d. increments)–it is reduced by unity if no packet arrives and increases by $(1/\mu - 1)$ if a packet arrives (Fig. 1.1). Let the remaining work in this imaginary system at time k be denoted by $W(k) = N(k)/\mu - k$. Let $w^r(t)$ denote the remaining work in heavy traffic scaling: $w^r(t) = W(r^2t)/r$. Thus heavy-traffic scaling corresponds to shrinking the time by a factor of r^2 and shrinking the space by a factor of r, where r is a large number. The limit of $w^r(t)$ as r tends to infinity would be denoted by $w(t)$. In general, the limit of a random process $x^r(t)$ (or random variable x^r) is denoted $x(t)$ (or x). Similarly, if $X(k)$ denotes the actual discrete-time process, $x(t)$ would denote the heavy traffic scaling of the process.

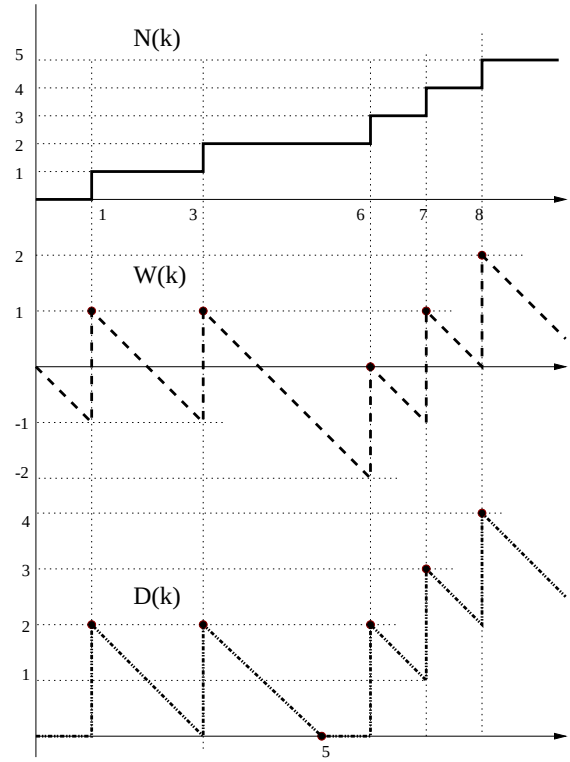

Figure 1: Example of an arrival process: $N(k)$, $W(k)$ and $D(k)$. Each packet is assumed to have size 2. X- axis denotes the discrete time. The discrete-time values of these discrete-time processes are linearly interpolated here.

We will show that $w(t)$ behaves as a Brownian motion. However, directly finding the distribution of $w^r(t)$ for a general renewal process is difficult. Hence, we will first find the distribution of the remaining work after a large number of arrivals. Then show that it is distributed similar to the remaining work after large time, i.e. similar to the distribution of $w^r(t)$ as r tends to infinity.

Before proceeding further, we briefly discuss Brownian motion. Let $\{\Delta_i\}$ be i.i.d. random variables with zero mean and variance σ^2. Define $B^r(t)$ as

$$B^r(t) = \frac{\sum_1^{r^2 t} \Delta_i}{r}$$

Let the limit of these processes as r tends to infinity be denoted by $B(t)$. Note that by central limit theorem, the marginal distribution of $B^r(t)$ converges to $\mathcal{N}(0, \sigma^2 t)$ in distribution. Also note that increments of $B(t)$ over disjoint time-intervals are independent because they are composed of disjoint subsets of $\{\Delta_i\}$. Finally, note that $B(t_2) - B(t_1)$ is distributed as $\mathcal{N}(0, \sigma^2|t_2 - t_1|)$. Such a process $B(t)$ is called a standard Brownian motion of standard deviation σ^2, which is denoted by $\mathcal{B}_{0,\sigma^2}(t)$. $B(t) + \theta t + c$ is called a Brownian motion with drift θ and shift c.

Coming back to the delay analysis, let $T(n) = \sum_1^n A_i$ denote time of the n'th packet arrival. The remaining work $W(T(n))$ after of the n'th packet is given by

$$W(T(n)) = n/\mu - T(n) = n/\mu - \sum_{i=1}^{n} A_i$$

Now define the following process in heavy traffic scaling

$$\begin{aligned} v^r(t) &= W(T(r^2t))/r && (1)\\ &= \frac{r^2t/\mu - T(r^2t)}{r} && (2)\\ &= \frac{\sum_1^{r^2t}(1/\mu - A_i)}{r} && (3) \end{aligned}$$

It is the amount of remaining work after r^2t arrivals, scaled down by a factor of r. Note that the average $\mathcal{E}[1/\mu - A_i] = 1/\mu - 1/\lambda = 0$ due to heavy traffic assumption and its variance equals the variance of A_i, which is a^2. Now the independence of $\{A_i\}$ implies that $v^r(t)$ converges to $\mathcal{B}_{0,a^2}(t)$. Using this, we now show that $w^r(t)$ also converges to a Brownian motion.

By the limit theorem for renewal processes, we know that as r goes to infinity,

$$\frac{N(r^2t)}{r^2t} \overset{a.s.}{\to} \lambda \text{ and } \frac{T(r^2t)}{r^2t} \overset{a.s.}{\to} 1/\lambda \quad (4)$$

Now note that $w^r(t)$ equals

$$\begin{aligned} w^r(t) &= \frac{N(r^2t)/\mu - r^2t}{r}\\ &= \frac{1}{r}\left(\frac{N(r^2t)}{r^2t}\frac{r^2t}{\mu} - T(r^2t)\frac{r^2t}{T(r^2t)}\right) \end{aligned}$$

Using (4) indicates that as r goes to infinity, $w^r(t)$ behaves as a scaled version of $v^r(t)$, which converges to a Brownian motion. Thus the remaining work in this imaginary system (where server is always active) behaves as a standard Brownian motion. Having obtained the distribution of the remaining work in this imaginary system, we can now go back to the work process in the actual system, where the remaining work cannot be negative. The actual remaining work after time k be denoted by $D(k)$. The relation between $D(k)$ and $W(k)$ is given by

$$D(k) = W(k) - \min_{0\le i\le k} W(i)$$

It is understood by the Figure 1.1 that this simply means the server cannot serve a job before it arrives. If $d^r(t) = D(r^2t)/r$ denotes the actual remaining work in the heavy traffic scaling,

$$d^r(t) = w^r(t) - \min_{\tau\in[0,t]} w^r(\tau)$$

This mapping from $w^r(.)$ to $d^r(.)$ is continuous. Hence the limit of the mapping has the same distribution as mapping of the limit of $w^r(t)$. However, the limit of $w^r(t)$ is a Brownian motion and the above mapping for a Brownian motion yields a *reflected* Brownian motion. Hence $d^r(t)$ also behaves as a reflected Brownian motion. Since all packets have length $1/\mu$, the queue-length process (the number of packets in the queue) behaves similar to the remaining work $d(t)$. That is, the queue-length behaves as a reflected Brownian motion. This is also true when the service requirements of packets are i.i.d. random variables. This ends our discussion about the origin of Brownian motion in queuing systems. Similar analysis can be done for more complex queueing systems with many queues and many servers or a switch, but the basic ideas are the same.

Remark Using almost the same steps, the reader can verify that if the work arrives at a rate bounded away from 1 i.e. if $\lambda/\mu = 1 - \delta < 1$, then the process $w(t)$ is a Brownian motion with drift $-\infty$. Hence the actual work $d(t)$ i.e. the corresponding reflected Brownian motion is always zero. Thus if a resource is not in heavy traffic, its queue-length in heavy traffic scaling is always zero.

1.2 State-space collapse

We will discuss the state-space collapse phenomenon using a simple example of a single server serving 2 queues, one at a time. Let the queue-length vector at time k be $Q(k)$, where its i'th element denotes length of the i'th queue[1]. Assume all packets to be of unit size. Assume that packets arrive at each queue according to a renewal process of rate λ_i and the two arrival processes are assumed to be independent. The stability constraint for this system is given by $\lambda_1 + \lambda_2 \leq 1$. The heavy traffic assumption in this case implies that the total arrival rate $\lambda_1 + \lambda_2 = 1$.

Let $q^r(t)$ denote the queue-length vector in heavy traffic scaling i.e. $q^r(t) = Q(r^2t)/r$. Let $q^r(t_0)$ be equal to $[a, b]$. Let $[c, d]$ be any other queue-length vector such that $a + b = c + d$. We will show (half-rigourously) that as r goes to infinity, the system can change the queue-length vector to any other $[c, d]$ instantaneously.

It is enough if we show that starting from queue-lengths $[a, b]$, queue-lengths $[0, a + b]$ can be attained in zero time. $q^r(t_0) = [a, b]$ corresponds to $Q(r^2t_0) = [ra, rb]$. The system now decides to keep serving the first queue till it drains out, assuming no new arrivals. This takes ra units of actual time i.e. a/r units of time in heavy traffic scaling. However, by limit theorem for renewal processes, for large enough r, there are roughly $\lambda_1 ra$ new actual arrivals i.e. the scaled queue-length q_1^r becomes $\lambda_1 a$ due to new arrivals. Again clearing out these new arrivals, (ignoring newer arrivals) takes $r\lambda_1 a$ units of actual time i.e. $\lambda_1 a/r$ time units in heavy traffic scaling. Due to newer arrivals however, the queue-length becomes $q_1^r = \lambda_1^2 a$.

Continuing this argument recursively, the first queue will eventually be empty because λ_1^n goes to zero. The total time in heavy traffic scaling, for this to happen equals $(a + a\lambda_1 + a\lambda_1^2 \cdots)/r = \frac{a/r}{1-\lambda_1}$. This corresponds to an actual time of $\frac{ra}{1-\lambda_1}$. Applying the limit theorem for renewal processes to the second queue implies that the number of actual new arrivals to the second queue in this much time roughly equals $\delta_2 = \frac{\lambda_2 ra}{1-\lambda_1}$. This corresponds to $\frac{\lambda_2 a}{1-\lambda_1}$ new arrivals in heavy traffic scaling. The second queue has not been served yet, so its queue-length equals the sum of the original queue-length and the number of new arrivals. Hence $q_2^r = b + a\frac{\lambda_2}{1-\lambda_1} = b + a$, because $\lambda_2 = 1 - \lambda_1$ due to the heavy traffic assumption. Thus $q^r = [0, b + a]$ after $\frac{a/r}{1-\lambda_1}$ time units in heavy traffic scaling. This time goes to zero as r goes to infinity and thus $[0, b + a]$ is attained zero time.

Thus in heavy traffic scaling, a system can instantaneously switch between two states with equal total queue-length. Hence from the system's point of view, any two states with the same total queue-length are equivalent in heavy traffic. In heavy traffic, this system with two queues is thus equivalent to a single-queue system, whose queue-length is given by the sum of the two queue-lengths of the original two-queue system. Hence the sum queue-length of this two-queue system behaves as a reflected Brownian motion [2].

[1] In this paper, x_i would mean the element i of vector x and y_{ij} would mean the element of matrix y.

[2] As discussed in Elif's talk, if the first queue costs more per packet, the system should shift all the $q_1(t) + q_2(t)$ packets to the second queue. Practically, this suggests that the system should serve the

This phenomenon of reduction in the dimensionality of the state (i.e. $q(t)$) is called as the state-space collapse. Similar result holds true for a system with m queues, where the stability constraint on the arrival rates is

$$\sum_{1}^{m} \xi_i \lambda_i \leq c \quad \text{for some } c > 0 \quad \text{and} \quad \xi \geq 0. \tag{5}$$

The two-queue system discussed above was a special case of this constraint, where $\xi_1 = \xi_2 = c = 1$. This general system is said to be in heavy traffic if $\sum \xi_i \lambda_i = c$. The state of such a system is defined by $\sum \xi_i q_i(t)$. In heavy traffic, this system can switch instantaneously between q and $\hat{q}$ if $\sum \xi_i q_i = \sum \xi_i \hat{q}_i$. Again, the state $\sum \xi_i q_i(t)$ behaves as a reflected Brownian motion.

Generalizing further, if a system with m queues has multiple stability constraints given by the matrix inequality

$$\xi\lambda \leq c \quad \text{where } \xi \in {\mathcal{R}_+}^{n \times m}, \ \lambda \in {\mathcal{R}_+}^{m}, \ c \in {\mathcal{R}_+}^{n}$$

where $\mathcal{R}_+$ denotes $[0, \infty)$. Let the arrival rates be such that first k constraints are met with equality. The state of the system is then given by the first k entries of the vector ξq. The state-space is collapsed to k dimensions from the original m dimensions.

2 Stability and Steady State of a Switch in Fluid Limit

2.1 Switch Properties, Fluid Scaling and Stability

We will consider a $n \times n$ switch with n input and output ports. Each input port i has a separate queue for every output j (Fig. 2.1). This will be called (i, j) queue. The queue-length of this queue at time k is denoted by $Q_{ij}(k)$ and its arrival rate is denoted by λ_{ij}. For simplicity, all queues are assumed to have non-zero arrival rates. Arrival process at each queue (i, j) is independent of arrival processes of all other queues. It is a renewal process of rate λ_{ij}. Each packet is of unit length. Each unit time, the switch can connect the input ports and output ports such that each input port is connected to exactly one output port and vice versa. This is called the crossbar constraint. Note that the parallel server network considered last week cannot be used to model a switch i.e. the crossbar constraint.

The switch connections at a given time can be represented by a permutation matrix π where $\pi_{ij} = 1$ indicates that input i was connected to output j. If that queue was not empty before service, $\pi_{ij} = 1$ also means that one packet from the queue (i, j) has been served. A permutation π is also called a *schedule* or a *matching* (from the input ports to the output ports). For stability, the arrival rate at any input or output port should not exceed 1, because only one packet can be served per unit time per port. Hence the stability constraints of a switch are given by

$$\sum_{k} \lambda_{ik} \leq 1 \quad \text{and} \quad \sum_{k} \lambda_{kj} \leq 1 \ \ \forall i, j \tag{6}$$

---

cheaper queue only if the costlier queue is empty. Thus in the heavy traffic limit, $q_1(t)$ will be zero at all times. The queue-length for the cheaper queue will be the same as the total queue-length, which is a reflected Brownian motion. Thus only the cheaper queue will be in heavy traffic and the other queue-length $Q_1(\cdot)$ will be essentially negligeble compared to the cheaper queue $Q_2(\cdot)$.

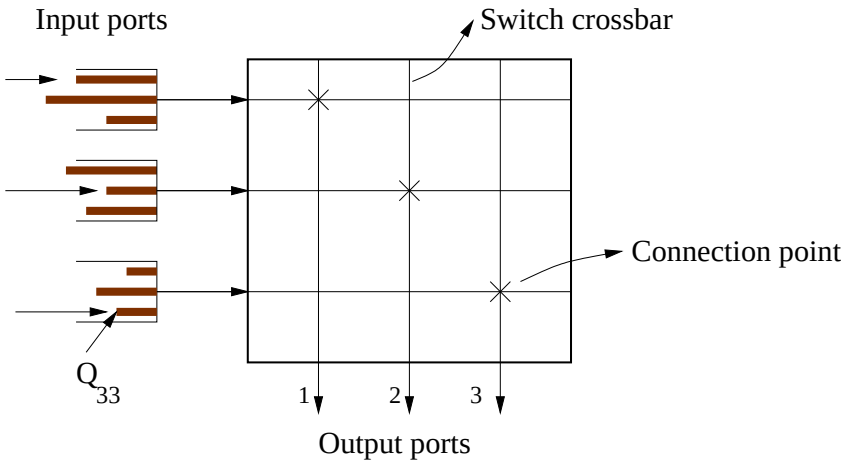

Figure 2: Schematic diagram of a switch showing the queues at each input port. Each input port i is connected to output port i, i.e., queues $(1,1)$, $(2,2)$ and $(3,3)$ are served.

Any arrival rate matrix satisfying the above constraints would be called a *stable arrival rate* matrix. Note that a doubly stochastic matrix would obey all of these constraints with equality.

The queue-length matrix at time k denoted by $Q(k)$ obeys

$$Q(k+1) = Q(k) - D(k) + A(k+1)$$

where $A(k+1)$ denotes the number of arrivals at time $k+1$ and $D(k)$ denotes the departures in the time interval $[k, k+1]$. If the matching at time k is denoted by $\pi(k)$ and $\pi_{ij}(k) = 1$, then $D_{ij}(k) = 1$ if $Q_{ij}(k) = 1$ and zero other wise–a packet will only depart if it existed. Mathematically, if $1_{\{E\}}$ denotes the indicator function for event E,

$$D_{ij}(k) = \pi_{ij}(k) 1_{\{Q_{ij}(k)>0\}} \tag{7}$$

The matrix of total number of arrivals and departures up to time k are denoted by $\bar{A}(k) = \sum_{i=1}^{k} A(i)$ and $\bar{D}(k) = \sum_{i=1}^{k-1} D(i)$, respectively. Let $\bar{P}_\pi(k)$ indicate the number of times the matching π was used till time k. Note that $\sum_\pi \bar{P}_\pi(k)$ equals k. Finally, $P(k)$ is the vector of all $n!$ matching counters $\{\bar{P}_\pi(k)\}$. The switch operation is completely described the process $X(.) = (Q(.), \bar{A}(.), \bar{D}(.), \bar{P}(.))$.

A maximum weight matching algorithm chooses a matching π^*, which maximizes the weight $\sum_{ij} \pi_{ij} f(Q_{ij}) \stackrel{\Delta}{=} \alpha_f(\pi, Q)$ over all permutations π for a given function $f(.)$. This maximum weight matching algorithm will be called MWM-f algorithm. We discuss MWM-f throughout the remaining paper.

Now we define the fluid scaling of the system.

$$X^r(t) = X(rt)/r$$

Note that both time and space are shrunk by a factor of r as opposed to the heavy traffic scaling. To avoid confusion with the notation for the heavy traffic limit, the limit of $X^r(t)$ is denoted by $x'(t) = (q'(t), a'(t), d'(t), p'(t))$. Time derivative of $x'(t)$ would be denoted by $\dot{x}'(t)$ and $x'(t)$ should not be confused as the time derivative of $x(t)$.

Using law of large numbers, we can show that almost surely,

$$\begin{aligned} a'_{ij}(t) &= \lambda_{ij} t \quad \text{i.e.} \quad a'(t) = \lambda t & (8)\\ q'(t) &= \lambda t - d'(t) & (9)\\ \sum_{\pi} {p'}_{\pi}(t) &= t \quad \text{hence} \quad \sum_{\pi} \dot{p'_{\pi}}(t) = 1 & (10) \end{aligned}$$

Now we rewrite the departure equation (7) as

$$D_{ij}(k) = \sum_{\pi} \pi_{ij} 1_{\{Q_{ij}(k)>0\}} (\bar{P}_{\pi}(k+1) - \bar{P}_{\pi}(k))$$

because $(\bar{P}_{\pi}(k+1) - \bar{P}_{\pi}(k))$ is nonzero (unity) only for the matching $\pi_{ij}(k)$ at time k. Using the above equation the following fluid relation is obtained

$$\dot{d_{ij}}'(t) = \sum_{\pi} \pi_{ij} 1_{\{q'_{ij}(t)>0\}} \dot{p'_{\pi}}(t) \tag{11}$$

Let $\sigma(t)$ denote the service rate matrix at time t,

$$\sigma(t) = \sum_{\pi} \pi \dot{p'_{\pi}}(t) \tag{12}$$

Differentiating Eq. (9) and using Eq. (11) yields

$$\begin{aligned} \dot{q'_{ij}}(t) &= \lambda_{ij} - \sigma_{ij}(t) \qquad \text{if} \quad q'_{ij} > 0 & (13)\\ &= (\lambda_{ij} - \sigma_{ij}(t))^{+} \qquad \text{if} \quad q'_{ij} = 0 & (14) \end{aligned}$$

These equations are quite intuitive if one imagines a water container with water level q'_{ij}, an input tap flowing at rate λ_{ij} and an output tap flowing at rate $\sigma_{ij}(t)$. In short, the above function for all $i,\ j$ will be written in matrix form as $\dot{q'}(t) = (\lambda - \sigma(t))^{+[q'=0]}$.

After this setup, we are ready to analyze the stability of a switch under MWM-f algorithm. We will use the following notion of stability.

Definition 1 *A switch algorithm is stable if for any stable arrival rate matrix λ, $q'(0) = 0$ implies that $q'(t) = 0$ for all $t > 0$.*

Condition 1 We assume the weight function $f : \mathcal{R}_+ \to \mathcal{R}_+$ is a strictly increasing continuous function and $f(0)$ equals zero. It should also satisfy a more involved condition that for any $(x_1, \cdots, x_n)$ and $(x_1, \cdots, x_n)$ in ${\mathcal{R}_+}^n$,

$$\sum_i f(x_i) \geq \sum_i f(y_i) \Leftrightarrow \sum_i f(\delta x_i) \geq \sum_i f(\delta y_i) \quad \forall \delta > 0$$

This condition ensures that if a particular matching is optimum (maximizes the weight) for actual queue-lengths $Q(.)$, it also maximizes the weight for those queue-lengths in fluid scaling i.e. $Q(.)/r$.

A MWM-f algorithm only chooses the matchings which maximize the overall weight and by our assumption about f, it only chooses the matchings which maximize the weight with scaled queue-lengths. Hence at every time t in the fluid scaling, the cumulative number of π matchings up to that time given by $p'_{\pi}(t)$ does not grow for suboptimal π. In other words, $\dot{p'_{\pi}}(t) = 0$ if π does not maximize $\sum_{ij} \pi_{ij} f(q'_{ij}(t)) = \alpha_f(\pi, q'(t))$. Let the

set of optimal matchings at time t (which maximize the weight $\alpha_f(\pi, q'(t))$) be denoted by $\pi^*(t)$. By Eq. (10), these optimal matchings grow such that the sum of their growth rates is 1.

$$\sum_{\pi \in \pi^*(t)} \dot{p'_\pi}(t) = 1 \tag{15}$$

Also from Eq. (12), the service rate is given by

$$\sigma(t) = \sum_{\pi \in \pi^*(t)} \pi \dot{p'_\pi}(t) \tag{16}$$

Now consider the following Lyapunav function of the queue-state q'

$$L(q') = \sum_{i,j} F(q'_{ij}) \qquad \text{where} \quad F(x) = \int_0^x f(y)\, dy.$$

We will show using the fluid equations that $L(q'(t))$ is a decreasing function of time for any stable arrival rate matrix λ. We will use the dot product notation $A \cdot B$ for two matrices A and B to denote $\sum_{i,j} A_{ij}B_{ij}$. Note that

$$\begin{aligned}
\frac{dL(q'(t))}{dt} &= \sum_{i,j} f(q'_{ij}(t)) \dot{q(t)}'_{ij} = f(q'(t)) \cdot \dot{q(t)}'_{ij} \quad \text{(chain rule of differentiation)} \\
&= f(q'(t)) \cdot (\lambda - \sigma(t))^{+[q'(t)=0]} \qquad \text{(by Eq.(13))} \\
&= f(q'(t)) \cdot (\lambda - \sigma(t)) \qquad \text{as } f(q'_{ij}) = 0 \text{ if } q'_{ij} = 0 \\
&\leq f(q'(t)) \cdot (\lambda^+ - \sigma(t))
\end{aligned}$$

where λ^+ is a doubly stochastic matrix such that $\lambda^+_{ij} \geq \lambda_{ij}$ for all i, j. Since every doubly stochastic matrix can be written as a convex combination all the permutation matrices, we have $\lambda^+ = \sum_\pi b_\pi \pi$ where $\sum b_\pi = 1$. Also note that $f(q'(t)) \cdot \sigma(t) = f(q'(t)) \cdot (\sum_{\pi \in \pi^*(t)} \pi \dot{p'_\pi}(t))$ from Eq. (16). If weight of any optimal matching at queue-state q' is denoted by $\alpha^*_f(q')$, we have $f(q'(t)) \cdot \pi = \alpha^*_f(q'(t))$ for all matchings π in the optimal set $\pi^*(t)$. Hence by Eq. (15),

$$f(q'(t)) \cdot \sigma(t) = \alpha^*_f(q'(t)) \sum_{\pi \in \pi^*(t)} \dot{p'_\pi}(t) = \alpha^*_f(q'(t)) \cdot 1$$

Now we have,

$$\begin{aligned}
\frac{dL(q'(t))}{dt} &\leq f(q'(t)) \cdot \left(\sum_\pi b_\pi \pi \right) - \alpha^*_f(q'(t)) \\
&\leq \left(\sum_\pi b_\pi \right) \alpha^*_f(q'(t)) - \alpha^*_f(q'(t)) \\
&= \alpha^*_f(q'(t)) - \alpha^*_f(q'(t)) = 0
\end{aligned}$$

The second last step follows because by definition, $f(q'(t)) \cdot \pi \leq \alpha^*_f(q'(t))$ for all π. Thus we have shown that the Lyapunav function is non-increasing at all times.

Now to prove the stability, we simply note that if $q'(0) = 0$ then $L(q'(0)) = 0$. Moreover, $L(q'(t)) = 0$ for all $t > 0$, because $L(q'(t))$ is nonnegative and non-decreasing– it cannot decrease below 0. Now $L(q'(t)) = 0$ implies $q'(t) = 0$, because $L(q')$ is non-zero for any other q'. Thus the definition of stability is satisfied and every MWM-f algorithm is proved to be stable.

2.2 Steady State of the Switch in Fluid Scaling

A state q_1 is called a steady state of the switch in fluid scaling if $q'(t_0) = q$ implies $q'(t) = q$ for all $t > t_0$. For example, 0 is a steady state as proved earlier. Henceforth, we will consider the case when one or more (input/output) ports are in heavy traffic. Let $q'_{i\cdot} = \sum_j q'_{ij}$ denote the total queue-length at input port i. $q'_{\cdot j}$ is similarly defined for every output port j. Similarly, $\lambda_{i\cdot}$ and $\lambda_{\cdot j}$ are defined as the sum arrival rate at input port i and output port j, respectively. An input port being in heavy traffic means that $\lambda_{\text{i}\cdot} = 1$ (or $\lambda_{\cdot j} = 1$ if it is an output port).

Now we show that if an input port i is in heavy traffic, $q'_{i\cdot}(t)$ is a non-decreasing function of time. From Eq. (13),

$$\begin{aligned} q'_{i\cdot}(t) &\geq \lambda_{i\cdot} - \sigma_{i\cdot} \quad \text{because } (\lambda_{ij} - \sigma_{ij})^{+[q'_{ij}=0]} \geq \lambda_{ij} - \sigma_{ij} \\ &\geq 0 \quad \text{because } \lambda_{i\cdot} = 1 \end{aligned}$$

The above expressions simply state that the input port i can at most be served at a rate of 1, which corresponds to always serving a nonempty queue at that port. Similar result holds true for an output port. Thus we have shown that the fluid scaling queue-length at a port in heavy traffic cannot decrease with time[3].

Trajectory Constraints We have shown two constraints over the trajectories of the queue-state $q'(t)$: 1) The Lyapunav function $L(q'(t)$ cannot increase over time 2) The queue-length at the heavy traffic ports cannot decrease over time.

Now also note that the Lyapunav function $L(q') = \sum_{ij} F(q'_{ij})$ is strictly convex in q'. It is because we have assumed that $f(.)$ is strictly increasing. Hence over any convex region, there is unique minimum for $L(.)$. Now if q is the initial state, any future state $q'_(t)$ is such that $q'_{i\cdot} \geq q_{i\cdot}$ and $q'_{\cdot j} \geq q_{\cdot j}$ for the ports in heavy traffic. Thus the future states lie in a convex region defined by the initial state and the Lyapunav function has a unique minima in this region. Since Lyapunav function keeps decreasing, the queue-state will eventually land up at that minima. Now if the initial state q itself was this minima, the queue-state would remain unchanged. Thus we have the following characterization of the steady state.

Theorem 2 *q is a steady state if an only if q itself is the solution to the optimization problem based on q: minimize $L(r)$ over the set of non-negative r such that $r_{i\cdot} \geq q_{i\cdot}$ and $r_{\cdot j} \geq q_{\cdot j}$ for the heavy traffic port(s). Also note that $\frac{dL(q'(t))}{dt} = 0$ at the steady state, because $q'(t)$ remains unchanged and hence its function also remains unchanged.*

We noted in the previous discussion that starting with any state, a steady state is reached eventually. Once a steady state is reached, it remains unchanged (by definition). To be precise, for arbitrarily small $\epsilon > 0$ and any initial state $q'(0)$, the queue-state $q'(t)$ goes within an ϵ-neighborhood of a steady state q, that is, $||q'(t) - q|| < \epsilon$ within some finite time $t \leq T(\epsilon)$.

[3]Compare this with the behavior of the queue-length in heavy traffic scaling, which behaves as a reflected Brownian motion and thus can decrease over time.

3 Heavy traffic theory

We have seen earlier that the heavy traffic scaling is given by $x^r(t) = X(r^2t)/r$, where $X(.)$ denotes the original discrete time process $(Q(.), \bar{A}(.), \bar{D}(.), \bar{P}(.))$ and $x^r(.)$ denotes the heavy-traffic scaled process. In this discussion, r should be thought as a large but fixed number. Recall that the fluid scaling $X^r(t) = X(rt)/r$ corresponds to shrinking time and space by a factor of r. The heavy traffic scaling further shrinks $X^r(t)$ in time by a factor of r. Intuitively, each instant in a heavy traffic process is a long time in fluid process. Since a fluid process converges to a steady state after some time, the heavy traffic process is in a steady state at every instant[4]. The next few paragraphs sketch this intuition little more precisely.

Studying the heavy traffic scaling $x^r(t)$ i.e. $(q^r(.), a^r(.), d^r(.), p^r(.))$ for a time interval of $t \in [0, T]$ corresponds to the original process in time $[0, r^2T]$. We form r sub-intervals of this heavy traffic time interval $[0, T]$. For ease of understanding, assume that these sub-intervals are disjoint and have equal length i.e. the interval m is $[\frac{mT}{r}, \frac{(m+1)T}{r}]$.

Consider the heavy-traffic process $x^r(t)$ in interval m. Stretch it in time by a factor of r. Define this new stretched process as $x^{rm}(t) = (q^{rm}(.), a^{rm}(.), d^{rm}(.), p^{rm}(.))$. More precisely,

$$\begin{aligned}
\forall\, t \in [0,T] \quad a^{rm}(t) &= a^r\left(\frac{t+mT}{r}\right) - a^r\left(\frac{mT}{r}\right) \\
d^{rm}(t) &= d^r\left(\frac{t+mT}{r}\right) - d^r\left(\frac{mT}{r}\right) \\
p^{rm}(t) &= p^r\left(\frac{t+mT}{r}\right) - p^r\left(\frac{mT}{r}\right) \\
q^{rm}(t) &= q^r\left(\frac{t+mT}{r}\right)
\end{aligned}$$

These equations can be re-written by substituting $x^r(t) = X(r^2t)/r$.

$$\begin{aligned}
\forall\, t \in [0,T] \quad a^{rm}(t) &= \frac{\bar{A}(rt+rmT) - \bar{A}(rmT)}{r} \\
d^{rm}(t) &= \frac{\bar{D}(rt+rmT) - \bar{D}(rmT)}{r} \\
p^{rm}(t) &= \frac{\bar{P}(rt+rmT) - \bar{P}(rmT)}{r} \\
q^{rm}(t) &= \frac{Q(rt+rmT)}{r}
\end{aligned}$$

Note that for every $1 \leq m < r$, process $x^{rm}(t)$ is a fluid scaling by r of the original discrete-time process. Recall that any starting state reaches ϵ-neighborhood of a steady state in fluid scaling within finite time $T(\epsilon)$. Thus if $T \gg T(\epsilon)$, each of the fluid processes essentially always (excluding the initial $T(\epsilon)$ time units) remains in steady state. Now concatenating all the r fluid processes $x^{rm}(t)$ ahead of each other yields the original heavy traffic process stretched in time by a factor of r i.e. $x^r(t/r)$.

This concatenated process is essentially always in steady state, as its pieces are essentially always in steady state. Each piece may be in a different steady state however.

[4]However, different times may be in different steady states.

Shrinking back $x^r(t/r)$ by a factor of r gives the original heavy traffic process of interest, $x^r(t)$. In every time interval $[\frac{mT}{r}, \frac{mT+T}{r}]$, this process is essentially always in some steady state. Taking r to infinity implies that the heavy traffic limit $q(t)$ is always in some steady state.

By steady state in heavy traffic limit, we meant any state q, which satisfies the optimization problem in Theorem 2. Because the queues behave as a Brownian motion in heavy traffic scaling, a steady state in heavy traffic scaling should not be interpreted as that in fluid scaling (which meant a steady state remains unchanged once reached).

Now recall that the optimization problem in Theorem 2 is defined by the sum queue-length(s) at the heavy traffic port(s). Since a steady state is a unique solution to this problem, it is completely defined by the sum queue-lengths at the heavy traffic ports. Since $q(t)$ for every t is a steady state in the heavy traffic limit, only specifying the sum queue-lengths ($q_{i\cdot}$ or $q_{\cdot j}$) at the heavy traffic ports completely describes the entire state $q(t)$. Thus the dimensionality of the state is reduced from n^2 to the number of ports in heavy traffic–state-space collapse again.

3.1 Only one port in heavy traffic

Without loss of generality, consider that the first input port is in heavy traffic, because similar analysis can be performed even for an output port in heavy traffic. Now if $q_{1\cdot}(t_0) = a$, the entire queue-state $q(t_0$ is given by the solution to the optimization: minimize $L(q)$ such that $q_{1\cdot} \geq a$ and all $q_{ij} \geq 0$. First, all rows of $q(t_0)$ other than the first should be zero for optimality. Second, the elements of the first row should be all equal due to Jensen's inequality and the fact that $L(q)$ is convex. Thus the entire queue-state is described as follows: the first row of $q(t_0)$ equals $[\frac{a}{n}, \cdots, \frac{a}{n}]$ and all other rows are zero.

More generally, when more than one port is in heavy traffic, if neither input port i nor output port j are in heavy traffic, $q_{ij}(t)$ should be 0 at all times[5]. Thus analyzing a switch with first k input and output ports in heavy traffic is same as analyzing a smaller switch of size k with all its ports in heavy traffic.

Going back to case of single heavy traffic port, recall the discussion at the end of section 1 (after Eq. (5)). Since the rate constraint $\sum_j \lambda_{1j} = 1$ is met with equality, $\sum_j q_{1j}(t)$ i.e. $q_{1\cdot}(t)$ is a reflected Brownian motion. Hence all the entries of the first row of the state $q(t)$ are also a reflected Brownian motion–moreover all of them are always equal to each other. [1] also generalizes these results for a broader definition of switch.

3.1.1 Comments

Assume that it costs $\sum_{i,j} F(Q_{ij}) = L(Q)$ each time the queue-state is Q (in discrete time). In the heavy traffic scaling, this will correspond to a cost rate of $\dot{c}(t) = \sum_{i,j} F(q_{ij}) = L(q(t))$. Since MWM-$f$ algorithm minimizes $L(q(t))$ for every t, it also minimizes the total cost incurred up to any given time. That is, it minimizes $c(t) = \int_0^T \dot{c}(t)dt$. If a weight function of the form $f(x) = x^\beta$ is used for some $\beta > 0$, we will call it MWM-β algorithm with little abuse of notation. MWM-β will minimize $\sum_{i,j} q_{ij}^{1+\beta}(t)$ for all t. Thus it minimizes the total cost incurred with the cost rate $\sum_{i,j} q_{ij}^{1+\beta}(t)$.

[5]This reminds us of the remark at the end of Section 1.1.

3.2 All ports in heavy traffic: minimizing delay

Minimizing the sum delay incurred by the packets is of interest in practice. By Little's law, this also corresponds to minimizing the sum of queue-length at all times. Thus we need an algorithm which minimizes $\sum_{i,j} Q_{ij}(k)$ at all times. However, we simplify the problem by looking at a coarser scaling, that is just find an algorithm which minimizes the sum queue-length in heavy traffic scaling ($\sum_{ij} q_{ij}(t)$) at all times k. Unfortunately, any MWM-f algorithm would not achieve this. Because, a weight function $f(x) = 1_{\{x>0\}}$ should be chosen for the Lyapunav function to equal the cost rate $\sum_{ij} q_{ij}(t)$. However, we have assumed strictly increasing f in our analysis (Condition 1).

This choice of $f(x) = 1_{\{x>0\}} \triangleq x^0$ is also called the maximum size matching, because it finds a matching of the maximum size in a bipartite graph, where an input port i is connected to an output port j if and only if $Q_{ij} > 0$. This algorithm is also called as MWM-0 algorithm. It does not give any extra service priority to the longer queues over the shorter queues. Delay optimality apart, it is known that it is not even a stable algorithm. That is, the Definition 1 of stability is not satisfied for some stable arrival rate matrix λ.

On the other hand, as discussed later, a slight modification of the MWM-0 algorithm, called as MWM-0+ algorithm is a stable algorithm, moreover, it minimizes the sum queue-length $\sum_{ij} q_{ij}(t)$ at all times in heavy traffic scaling. Thus it minimizes the average delay of the switch at least in the heavy traffic scaling. It is also shown that the traditional MWM-1 algorithm, which treats the queue-lengths Q_{ij} themselves as weights, is not a delay optimal algorithm–thus shattering a common folk-lore in switching community.

3.3 Delay optimality of MWM-0+

First we explain the meaning of this algorithm. This can be thought of as MWM-β algorithm for arbitrarily small $\beta > 0$. For small values of β,

$$Q_{ij}^{\beta} \approx 1 + \beta \log Q_{ij}$$

Thus every non-empty queue is given essentially unit weight and every empty queue weighs 0. Hence only maximum-size matchings need to be considered further. Let $\{e_1 \cdots, e_n\}$ denote the set of edges in a matching. Among all maximum size matchings, the one which also maximizes the weight

$$n + \beta \log \left(\prod_i Q_{e_i} \right)$$

is chosen. Thus the maximum size matching which maximizes the product of the queue-lengths is chosen.

For proving the optimality, the authors first characterize the steady state of a MWM-f algorithm. Note that since all the ports are in heavy traffic, the set of sum queue-lengths at each port i.e. all $q_{i\cdot}(t)$ and $q_{\cdot i}(t)$ determines the entire matrix $q(t)$ in heavy traffic limit. We denote this vector of sum queue-lengths at all ports by $\tilde{q}(t)$. This $\tilde{q}(t)$ is also called as the state of process since it completely determines all $q_{ij}(t)$. The analysis for characterizing the steady state is omitted, as it is not as straight-forward as that in Section 3.1 where only one port was in heavy traffic. We will summarize the (somewhat surprising) final characterization without proof: A steady state q satisfying Theorem 2 is such that all the matchings have the maximum weight $\sum_{ij} \pi_{ij} f(q_{ij})$.

Then they show that for MWM-0+ algorithm, for every non-negative vector $z \in \mathcal{R}_+^{2n}$, there exists a steady state q such that the sum queue-length at port equals an element of z. That is, $q_{i\cdot} = z_i$ and $q_{\cdot j} = z_{n+j}$ for all i and j. They show this by using the fact that all matchings are optimal in a steady state q. Thus the state vector $\tilde{q}(t)$ can take any value in $z \in \mathcal{R}_+^{2n}$.

Under MWM-0+ algorithm, for any strictly positive (element-wise) $\tilde{q}(t)$, all the entries of the corresponding steady state $q(t)$ can be shown to be strictly positive. Then any matching serves all the inputs ports and all the outputs ports without idling at any port, that is, none of the crossbar connections are wasted.

Moreover, as seen before in Section 1, each element of the state vector $\tilde{q}(t)$ performs a reflected Brownian motion. The set of times when a reflected Brownian motion hits zero has measure zero. Hence, almost at all times, the algorithm does not idle at any port. Since the system is never idle, the sum of queue-lengths cannot be reduced further, thus proving the optimality of MWM-0+ algorithm.

Finally, for showing that the traditional MWM-1 is not delay optimal, they simply show that the state vector $\tilde{q}(t)$ no more lies in the entire space $\mathcal{R}_+^{2n}$. Instead, it lies in a strict sub-space $\mathcal{S}_1$ of $\mathcal{R}_+^{2n}$. Hence, there exist arrival processes such that the state-vector $\tilde{q}(t)$ would go outside $\mathcal{S}_1$ under MWM-0+ algorithm, but MWM-1 would retain it within $\mathcal{S}_1$ by idling on some ports.

In a similar manner, let the state vector $\tilde{q}(t)$ under MWM-β_1 algorithm lie in space $\mathcal{S}_{\beta_1}$ and that for MWM-β_2 lie in $\mathcal{S}_{\beta_2}$. They show that $\mathcal{S}_{\beta_2}$ is contained in $\mathcal{S}_{\beta_1}$ if $\beta_2 > \beta_1$. Hence, MWM-β_1 has a better delay performance than MWM-β_1.

4 Summary

We discussed the origin of Brownian motion in heavy traffic scaling. Then we saw the idea of state-space collapse for a simple two-queue system. Later, we described the switch properties and a class of maximum weight matching algorithms called MWM-f was shown to be stable using fluid scaling. Then steady state in fluid scaling was characterized as solution to an optimization problem. Then we showed that a process in heavy traffic scaling, is in a (fluid scaling) steady state at all times. We then studied the case with only one port in heavy traffic. Finally, a delay optimal algorithm was discussed and the delay of some other common algorithms was compared.

References

[1] A.L. Stolyar, "MaxWeight Scheduling in a Generalized Switch: State Space Collapse and Workload Minimization in Heavy Traffic," *Annals of Applied Probability*, Vol.14, No.1, pp.1-53, 2004.

[2] D. Shah, Randomization and heavy traffic theory: new approaches to the design and analysis of switch algorithms, *Ph.D. Thesis*, Stanford University, 2004.