

The Stability of Two-Station Multi-Type Fluid Networks

J. G. Dai

School of Industrial and Systems Engineering
and School of Mathematics
Georgia Institute of Technology
Atlanta, GA 30332-0205

J. H. Vande Vate

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0205

June 1997

abstract

This paper studies the fluid models of two-station multiclass queueing networks with deterministic routing. A fluid model is globally stable if the fluid network eventually empties under each non-idling dispatch policy. We explicitly characterize the global stability region in terms of the arrival and service rates. We show that the global stability region is defined by the nominal workload conditions and the “virtual workload conditions” and we introduce two intuitively appealing phenomena: virtual stations and push starts, that explain the virtual workload conditions. When any of the workload conditions is violated, we construct a fluid solution that cycles to infinity, showing that the fluid network is unstable. When all of the workload conditions are satisfied, we solve a network flow problem to find the coefficients of a piecewise linear Lyapunov function. The Lyapunov function decreases to zero proving that the fluid level eventually reaches zero under any non-idling dispatch policy. Under certain assumptions on the interarrival and service time distributions, a queueing network is stable or positive Harris recurrent if the corresponding fluid network is stable. Thus, the workload conditions are sufficient to ensure the global stability of two-station multiclass queueing networks with deterministic routing.

To appear in *Operations Research*
December 1998

Introduction

Queueing networks offer an appealing method for modeling complex manufacturing processes and have been used to model telecommunication networks and manufacturing systems like wafer fabrication facilities. Unfortunately, they are themselves generally too complex for successful analysis. For example, the primary tool for evaluating the performance of a given dispatch policy is simulation. In fact, we generally resort to simulation even to determine whether a queueing network is stable under a given dispatch policy or whether the servers are unable to manage the workload.

Even very simple queueing networks exhibit surprising and often counterintuitive behavior. Consider the simple two-station re-entrant queueing network depicted in Figure 1, where m_k is the average processing time for stage k and jobs arrive at rate λ . The two rectangles indicate the two stations, Station A and Station B, and the line traces the route the jobs follow between them. In this example, each job passes through five stages of processing: the first, at Station A, lasts an average of m_1 time units; the second, at Station B, lasts an average of m_2 time units; the third, again at Station A, lasts an average of m_3 time units, and so on. At any point in time, Station A, for example, may have jobs waiting in all three of the stages it serves and must decide which job to process next. A strategy for making these decisions at each station constitutes a dispatch policy. Dai and Vande Vate (1996) showed that under certain non-idling (or work-conserving) dispatch policies the servers are unable to serve jobs as quickly as they arrive even when the nominal workload at each station is significantly less than 100%, i.e., $\lambda(m_1 + m_3 + m_5) < 1$ and $\lambda(m_2 + m_4) < 1$. In fact, Dai and Vande Vate (1996) showed that if the servers give highest priority to jobs at stages 2 and 5, they will be unable to serve the jobs as quickly as they arrive unless the nominal utilization at each station is less than 100% *and*

$$\lambda(m_2 + m_5) < 1.$$

For example, if the average service times are

$$(0.1, 0.6, 0.1, 0.1, 0.6),$$

servers employing this dispatch policy will be able to keep up with the workload only if the arrival rate to the system satisfies $\lambda < 1/(m_2 + m_5) = 5/6$. The condition $\lambda(m_2 + m_5) < 1$ reflects the fact that under this dispatch policy Station A and Station B can serve their high priority stages

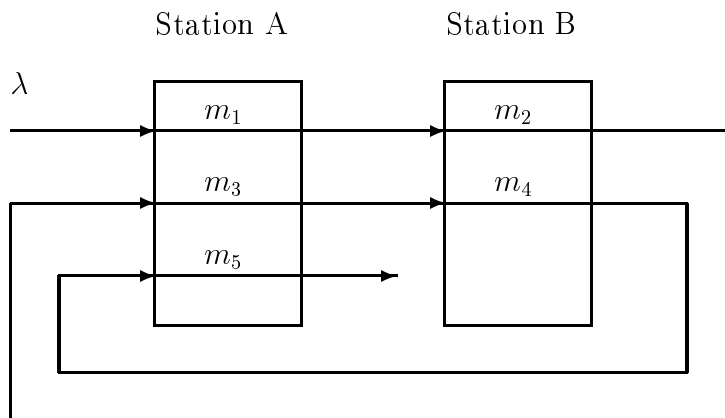


Figure 1: A five class network

simultaneously only during a transient initial period. Thus, although they are served by different stations, these two stages can form a bottleneck that determines the capacity of the entire system. We discuss this phenomenon, which we call a “virtual station”, in more detail in Section 2.

This paper focuses on the deterministic fluid network corresponding to a given queueing network. For the fluid network corresponding to the queueing network pictured in Figure 1, fluid arrives continuously from the outside at rate λ . The server at Station A pumps fluid in stages 1, 3 and 5 and the server at Station B pumps fluid in stages 2 and 4. When a server devotes its full effort to stage k fluid, it pumps at a maximum rate of $\mu_k = 1/m_k$, assuming there is stage k fluid, $k = 1, \dots, 5$. A dispatch policy in the fluid network context describes how to allocate each server’s pumping capacity at each time among the different stages it serves. A fluid network is stable under a dispatch policy if it will eventually empty no matter what the initial fluid levels are.

Our work is largely motivated by a result of Dai (1995) showing that, under certain distributional assumptions on interarrival and service times, a queueing network is stable or positive Harris recurrent if the corresponding fluid network is stable. Related work can be found in Rybko and Stolyar (1992), Chen (1995), Dai and Meyn (1995), Stolyar (1994), Meyn (1995), Dai (1996) and Bramson (1998a).

We develop necessary and sufficient conditions for a two-station fluid network to be *globally stable* or stable under any non-idling dispatch policy. Determining the global stability region is especially important when it is difficult or impossible to implement a well-studied dispatch policy. In such systems, it is possible for servers to unwittingly employ a policy under which the system is unstable even though the traffic intensity or nominal workload at each station is less than one. Although it is sometimes difficult to avoid such bad policies, we can avoid their consequences by maintaining service times that are in the global stability region. In this way, we can ensure that even under bad policies, the system will remain stable.

We show that a two-station fluid network is globally stable if and only if the processing times satisfy the nominal workload conditions *and* the “virtual workload conditions”. In particular, we introduce two intuitively appealing phenomena, *virtual stations* and *push starts*, that give rise to the two classes of virtual workload conditions: “virtual station conditions” and “push start conditions”.

Virtual stations affect the global stability of two-station fluid networks because, under some non-idling dispatch policies, certain groups of stages cannot be served simultaneously even though they are served at different stations. Thus, just as at stations, the traffic intensities at these groups must be less than one.

Push starts magnify the influence of virtual stations in fluid networks by giving highest priority to the first few stages. Fluid passes through these stages to the rest of the network as quickly as it arrives, but having focused so much attention on the first few stages, the servers have less capacity to dedicate to the rest of the network. Push starts do not influence the capacity of a single station because the work required to serve the remaining stages at a station is reduced by exactly the effort required to expedite the first few stages. Push starts do influence the capacity of virtual stations, however, because they involve stages at both stations: the effort spent expediting stages at one station does not reduce the work remaining at the other.

Under certain distributional assumptions, the virtual workload conditions together with the nominal workload conditions are sufficient to ensure the global stability of two-station queueing networks. An independent argument in a companion paper Dai and Vande Vate (1996) shows that the virtual station conditions are also necessary for the global stability of two-station queueing networks. The push start conditions, however, are *not* in general necessary. See, for example, Dai and Vande Vate (1996).

There has been a recent surge in the study of stability conditions for multiclass queueing

networks. These studies were primarily motivated by Kumar and Seidman (1990), Lu and Kumar (1991), Rybko and Stolyar (1992), Bramson (1994a, 1994b) and Seidman (1994), which demonstrated that a number of non-idling dispatch policies are unstable even if the traffic intensity at each station is less than one. In these unstable examples, the total number of jobs in the system goes to infinity with time. Other recent work on the stability of queueing networks and fluid networks includes: Harrison and Nguyen (1995), Bramson (1997, 1998b), Bertsimas, Gamarnik and Tsitsiklis (1996), Dumas (1996, 1997), Dai and Weiss (1996), Foss and Rybko (1995), Winograd and Kumar (1996), Kumar and Meyn (1995, 1996), Chen and Zhang (1997, 1998), Morrison and Kumar (1998), Hasenbein (1997).

Recently, Bertsimas, Gamarnik and Tsitsiklis (1996) showed that a two-station fluid network is globally stable if and only if a certain linear program has bounded objective value. In this paper we extend the results of Bertsimas et al. by stating explicitly in terms of the arrival rates and service times, necessary and sufficient conditions for a two-station fluid network to be globally stable.

The explicit description of necessary and sufficient conditions for the global stability of two-station fluid networks provides a number of corollaries not immediately available from the linear programming characterization of Bertsimas, Gamarnik and Tsitsiklis (1996). Most important among these is a complete understanding of global stability in two-station fluid networks via virtual stations and push starts. In addition, our conditions demonstrate that the global stable region of a two-station fluid network is *monotone*, i.e., reducing service times maintains global stability. This is not the case for stability with respect to a given dispatch policy. It is possible for a dispatch policy to be stable for a given fluid network, but unstable when the service times are reduced. For fluid networks with more than two stations, even the global stable region need not be monotone (see, for example, Dai, Hasenbein and Vande Vate 1998).

Our approach relies on the fact that a fluid network is stable if there is a piecewise linear Lyapunov function for it. We formulate the problem of determining the coefficients of the Lyapunov function as a linear programming problem, which has unbounded objective values only if the coefficients and hence the Lyapunov function exist. Our linear program arises directly from the piecewise linear Lyapunov function introduced in Dai and Weiss (1996), which generalizes that of Botvich and Zamyatin (1992) and is simpler than that independently formulated by Down and Meyn (1994).

We transform our linear program into a parametric network flow problem in an acyclic network. The fluid network is globally stable if there is a value of the parameter for which the minimum flow in this network is sufficiently small. Thus, invoking the Min-Flow Max-Cut Theorem, we see that the fluid network is globally stable if there is a value of the parameter for which the capacity of each cut in the acyclic network is sufficiently small. Finally, we show that these “cut conditions” are equivalent to the more easily understood virtual workload conditions.

In Dai and Vande Vate (1996), we show that the virtual station conditions are necessary for the stability of two-station *queueing* networks by showing that under certain non-idling dispatch policies, both stations cannot simultaneously serve the classes of a virtual station. This implies that the virtual station conditions are also necessary for the stability of two-station fluid networks. We also provide an example showing that the push start conditions need not be necessary for global stability in these networks. In this paper, we offer a direct construction showing that *both* the virtual station conditions and the push start conditions are necessary for the stability of two-station *fluid* networks. In particular, when the service times strictly violate the virtual workload conditions, we construct a non-idling dispatch policy that causes the work-in-process to grow without bound. We are able to construct the policy without knowing the specific service times since it depends only on the violated conditions. In every case, we identify a finite sequence of states through which the

fluid network cycles with greater work-in-process in each successive cycle.

The proof in Dai and Vande Vate (1996) that the virtual station conditions are necessary to ensure global stability requires less detailed analysis and applies to a broader class of networks encompassing both queueing networks and fluid networks. The proof presented here is rather detailed and applies only to two-station fluid networks. It does, however, demonstrate the necessity of the push start conditions for two-station fluid networks and show exactly how the work-in-process in an unstable system swings from class to class as it grows to infinity. Further, it is not apparent how to extend the simpler proof given in Dai and Vande Vate (1996) to certain classes of conditions necessary to ensure the global stability of fluid networks with more than two stations. These larger networks appear to require the more direct proof technique used in this paper (see, for example, Dai, Hasenbein and Vande Vate 1998).

We show that whenever the fluid network is unstable, there is a static buffer priority dispatch policy under which the work-in-process goes to infinity. Thus, the class of static buffer priority dispatch policies is “worst” among all non-idling policies in the sense that a two-station fluid network is globally stable if and only if it is stable under all static buffer priority dispatch policies.

We introduce the virtual workload conditions in Section 2 and show that they are sufficient for stability in Sections 3 through 6. Finally, we prove they are necessary for stability in Section 7.

1 Preliminaries

Before introducing the virtual workload conditions, we give a brief review of the Minimum Flow Problem and introduce our notation for fluid networks. For an excellent and accessible treatment of network flows, see Ahuja *et al.* (1993).

1.1 The Minimum Flow Problem

Consider a directed network (N, E) with node set N and edge set E . We distinguish two vertices s , the source, and t , the sink. Given (possibly infinite) lower bounds $\ell = (\ell_{ij})$ and upper bounds $u = (u_{ij})$, we wish to find a minimum flow from the source s to the sink t subject to flow conservation constraints and edge capacity constraints. Thus, the minimum flow problem is:

$$\begin{aligned}
 & \text{minimize } v \\
 & \text{subject to} \\
 (1.1) \quad & \sum_{j \in N} x_{sj} - \sum_{j \in N} x_{js} = v \\
 (1.2) \quad & \sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = 0 \text{ for each node } i \in N \setminus \{s, t\} \\
 (1.3) \quad & \sum_{j \in N} x_{tj} - \sum_{j \in N} x_{jt} = -v \\
 (1.4) \quad & \ell_{ij} \leq x_{ij} \leq u_{ij} \quad \text{for each edge } (i, j) \in E.
 \end{aligned}$$

Suppose (x, v) satisfies (1.1)–(1.4). We refer to the vector x as a *feasible flow* and the value v as the *value of the flow* x . A *minimum flow* is a feasible flow with smallest value among all feasible flows.

An s, t -cut in the network (N, E) is a partition of N into two sets S and T with $s \in S$ and

$t \in T$. The *capacity* of the cut (S, T) , denoted $c(S, T)$, is:

$$c(S, T) = \sum_{(i,j) \in E: i \in S, j \in T} \ell_{ij} - \sum_{(i,j) \in E: i \in T, j \in S} u_{ij}.$$

Note that our definition of capacity interchanges the roles of upper and lower bounds in the usual definition as applied to the maximum flow problem. This definition is appropriate for the minimum flow problem and is sometimes referred to as the *floor* of a cut. A *maximum s, t -cut* is one with largest capacity among all s, t -cuts. Theorem 1.1 is a classic result of network flows and can be found in Ahuja *et. al.* (1993, Exercise 6.18, pp. 202).

Theorem 1.1. *The value of a minimum flow equals the capacity of a maximum s, t -cut.*

1.2 Multi-Type Fluid Networks

We consider fluid networks with two single-server stations, denoted A and B , and a set I of different fluid types. Type i fluid arrives at a constant rate $\lambda_i > 0$ and follows a prescribed route visiting one station and then the other a number of times before exiting the system. Different types of fluid may follow different routes. We number the stages fluid i passes through consecutively from 1 to c_i and let A_i and B_i denote the stages in which fluid i is served at Station A and at Station B , respectively.

We refer to type i fluids waiting for the k^{th} stage as *class (i, k) fluids*, which reside in *buffer (i, k)* . Each unit of class (i, k) fluid requires service lasting $m_k^i > 0$ units of time. The service time m_k^i is the time it takes the station to process one unit of class (i, k) fluid. Equivalently, $\mu_k^i = 1/m_k^i$ is the rate at which the server depletes class (i, k) fluid from the buffer when it devotes all its efforts to serving that class.

A *fluid solution* is a vector $(Q(\cdot), T(\cdot)) = (Q_k^i(\cdot), T_k^i(\cdot))_{i \in I, k=1, \dots, c_i}$ of functions of time satisfying:

$$(1.5) \quad Q_k^i(t) = Q_k^i(0) + \mu_{k-1}^i T_{k-1}^i(t) - \mu_k^i T_k^i(t) \geq 0, \text{ for } t \geq 0, \quad i \in I, k = 1, \dots, c_i,$$

$$(1.6) \quad T_k^i(0) = 0 \text{ and } T_k^i(t) \text{ is nondecreasing for all } i \in I, \text{ and } k = 1, \dots, c_i,$$

$$(1.7) \quad t - \sum_{i \in I} \sum_{k \in A_i} T_k^i(t) \text{ is nondecreasing,}$$

$$(1.8) \quad t - \sum_{i \in I} \sum_{k \in B_i} T_k^i(t) \text{ is nondecreasing,}$$

where $T_0^i(t) = t$ and $\mu_0^i = \lambda_i$ for each type $i \in I$ to model the exogenous arrival of fluids.

We interpret $Q_k^i(t)$ as the volume of class (i, k) fluid in the buffer at time t , and $T_k^i(t)$ as the cumulative time spent serving class (i, k) fluids up to time t . The relationship between the buffer levels and the cumulative allocations of effort is given by (1.5) for each class of fluid. These equations simply relate the buffer levels to the initial buffer levels and the total volume of fluid entering and leaving each buffer. Conditions (1.6) ensure that no work is completed before time 0. Conditions (1.7) and (1.8) ensure that each server divides its time between serving the various classes and accumulating idle time.

Each fluid solution $(Q(\cdot), T(\cdot))$ has derivatives at almost all times $t > 0$ (with respect Lebesgue measure on $[0, \infty)$); see Dai and Weiss (1996). A point $t \in [0, \infty)$ is a *regular* point of the fluid solution (Q, T) if T is differentiable at t . We henceforth use $f'(t)$ to denote the derivative of f at t .

We consider fluid networks under non-idling dispatch policies or policies that do not allow a server to be idle when there is work for it to do. We can express this restriction via the “complementarity” conditions on fluid solutions $(Q(\cdot), T(\cdot))$:

$$(1.9) \quad \sum_{i \in I} \sum_{k \in A_i} \dot{T}_k^i(t) = 1 \text{ whenever } \sum_{i \in I} \sum_{k \in A_i} Q_k^i(t) > 0 \text{ and ,}$$

$$(1.10) \quad \sum_{i \in I} \sum_{k \in B_i} \dot{T}_k^i(t) = 1 \text{ whenever } \sum_{i \in I} \sum_{k \in B_i} Q_k^i(t) > 0,$$

for each regular point t of $(Q(\cdot), T(\cdot))$. The cumulative idle time at Station A up to time t is simply

$$t - \sum_{i \in I} \sum_{k \in A_i} T_k^i(t)$$

and Condition (1.7) ensures that it is nondecreasing. Condition (1.9) further ensures that when Station A is accumulating idle time, the buffers it serves are empty. The cumulative idle time at Station B is defined similarly. Henceforth, we consider only fluid solutions satisfying (1.5)–(1.10). When there is only a single type of fluid, we omit references to the type and speak of class k fluid as having buffer levels $Q_k(t)$, etc.

A buffer priority is a one-to-one mapping π from the set of buffers onto $\{1, \dots, c\}$, where c is the total number of classes in the network. When $\pi(i, k) > \pi(j, \ell)$ for two classes both served at the same station, class (i, k) has higher priority than class (j, ℓ) . A static buffer priority discipline with buffer priorities π stipulates that, in addition to (1.5)–(1.10), every fluid solution $(Q(\cdot), T(\cdot))$ must also satisfy:

$$(1.11) \quad \sum_{i \in I, k \in A_i, \pi(i, k) \geq \pi(j, \ell)} \dot{T}_k^i(t) = 1 \text{ whenever } \sum_{i \in I, k \in A_i, \pi(i, k) \geq \pi(j, \ell)} Q_k^i(t) > 0$$

for each $j \in I$ and $\ell \in A_j$, and regular point t of $(Q(\cdot), T(\cdot))$ and

$$(1.12) \quad \sum_{i \in I, k \in B_i, \pi(i, k) \geq \pi(j, \ell)} \dot{T}_k^i(t) = 1 \text{ whenever } \sum_{i \in I, k \in B_i, \pi(i, k) \geq \pi(j, \ell)} Q_k^i(t) > 0$$

for each $j \in I$ and $\ell \in B_j$ and regular point t of $(Q(\cdot), T(\cdot))$. Equations (1.11)–(1.12) dictate that whenever a buffer accumulates fluid, no lower priority buffer at the same station can receive service.

The fluid network is said to be *stable under non-idling dispatch policies*, or simply *globally stable*, if there is some finite time $\tau > 0$ beyond which any fluid solution $(Q(\cdot), T(\cdot))$ that begins with one unit of work-in-process or WIP, i.e., with

$$\sum_{i \in I} \sum_{k=1}^{c_i} Q_k^i(0) = 1,$$

will have no WIP, i.e.,

$$\sum_{i \in I} \sum_{k=1}^{c_i} Q_k^i(t) = 0,$$

for all $t \geq \tau$. Dai (1995) showed if a fluid network is stable, the corresponding queueing network is positive Harris recurrent under some distributional assumptions.

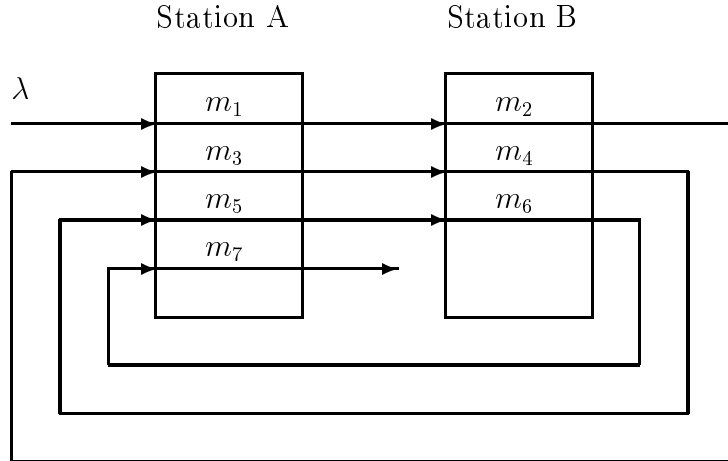


Figure 2: A seven class network

A fluid solution $(Q(\cdot), T(\cdot))$ is said to be unstable if there exists $\{t_n\}$ with $t_n \rightarrow \infty$ such that $Q(t_n) \neq 0$ for each n . A fluid solution $(Q(\cdot), T(\cdot))$ is said to diverge to infinity if the WIP goes to infinity as time $t \rightarrow \infty$. A divergent fluid solution is clearly unstable.

The traffic intensities or nominal workloads at the stations are:

$$\rho_A = \sum_{i \in I} \sum_{k \in A_i} \lambda_i m_k^i \quad \text{and} \quad \rho_B = \sum_{i \in I} \sum_{k \in B_i} \lambda_i m_k^i.$$

It is well-known (see, for example, Dai 1996) that the fluid network can only be stable if the traffic intensities are less than one, i.e.,

$$(1.13) \quad \rho_A < 1 \quad \text{and} \quad \rho_B < 1.$$

2 Virtual Workload Conditions

The stability conditions for a two-station fluid network take two forms: the nominal workload conditions (1.13) that arise because classes at the same station must share the server's time; and the *virtual workload conditions*, generalizing condition $\lambda(m_2 + m_5) < 1$ for the fluid network in Figure 1, that arise through the interactions between virtual stations and push starts.

Two intuitively appealing phenomena give rise to the virtual workload conditions. The intuition behind the first of these phenomena is best described in the context of queueing networks. The second phenomenon is most easily understood in the context of fluid networks.

Figure 1 illustrates a simple single-type queueing network. If we give highest priority to class 5 at Station A and to class 2 at Station B, these two classes can only be served simultaneously during a transient initial period, see Dai and Vande Vate (1996, Lemma 3.1). Thus, these two classes form a “virtual station” and, although they are served at different stations, the workload at these two classes cannot exceed 1. This virtual station gives rise to the virtual workload condition:

$$\lambda(m_2 + m_5) < 1,$$

which we refer to as a “virtual station condition”. These conditions also apply to fluid networks.

The fluid network of Figure 2 illustrates the second phenomenon giving rise to virtual workload conditions. Assume that the nominal workload conditions (1.13) hold. If we give highest priority to class 1 at Station A and to class 2 at Station B in this network, the fluid levels in these two buffers will reach zero and remain zero thereafter. For the sake of our discussion, we assume that these two buffers are always empty. Then, the server at Station A will *constantly* devote a fraction λm_1 of its time to class 1 to keep the buffer empty, and hence have only a fraction $1 - \lambda m_1$ of its time left for other classes at Station A . Similarly, the server at Station B will constantly devote a fraction λm_2 of its time to class 2 and have only a fraction $1 - \lambda m_2$ of its time left for the other classes at Station B . Note that in a queueing network we cannot anticipate a constant, uninterrupted devotion of time to these classes, but we can in a fluid network. The fact that the servers are slowed by their efforts on classes 1 and 2 magnifies the time required to serve each unit of fluid in the remaining classes. In particular, the server at Station A will require $m_7/(1 - \lambda m_1)$ units of time to complete one unit of class 7 fluid and the server at Station B will require $m_4/(1 - \lambda m_2)$ units of time to complete one unit of class 4 fluid. Because buffers 1 and 2 remain empty, fluid passes through them as quickly as it arrives, and hence arrives at buffer 3 at rate λ . Thus, push starting the first two classes magnifies the virtual station condition:

$$\lambda(m_4 + m_7) < 1$$

in the induced network to give the virtual workload condition:

$$\frac{\lambda m_4}{1 - \lambda m_2} + \frac{\lambda m_7}{1 - \lambda m_1} < 1,$$

ensuring that the virtual station can divide its time between serving the two classes. We refer to this condition as a “push start condition”.

Together, these two phenomena explain all the virtual workload conditions of two-station fluid networks. Although these ideas are intuitively appealing, formalizing them is more involved. We formalize the conditions under which classes at different stations cannot receive service simultaneously in the following way.

The first notion in our characterization of virtual stations is the idea of an *excursion* or set of consecutive classes at the same station. In the network of Figure 3 each type makes four excursions at Station A and four excursions at Station B . For example, the second excursion for type 2 consists of classes (2, 2) and (2, 3).

We let E denote the set of excursions and, for each type $i \in I$, we let E^i denote the set of excursions for type i customers, which we number consecutively from 1 to n_i . We partition E^i into E_A^i , the set of excursions at Station A , and E_B^i , those at Station B . Since an excursion at one station must be followed by an excursion at the other (unless it is the last excursion), one of these is the set of odd numbered excursions and the other is the set of even numbered excursions depending on where type i customers first enter the network. We use $[i, e]$ to denote the e^{th} excursion for type i fluid. Recall that (i, k) denotes the type i fluid that is waiting for its k^{th} service.

We let $E[i, e]$ denote the classes of excursion $[i, e]$ and we partition these classes into the *last class* and all the rest, which we call *first classes* of the excursion. We let $\ell[i, e]$ denote the last class and $f[i, e]$ the set of first classes in $E[i, e]$. If an excursion consists of only one class, that class is the last class and the excursion has no first classes. For example, in the twenty-eight class network of Figure 3, $\ell[1, 1] = (1, 1)$, $f[1, 1] = \emptyset$, $\ell[1, 2] = (1, 3)$, $f[1, 2] = \{(1, 2)\}$, \dots , $\ell[1, 8] = (1, 14)$ and $f[1, 8] = \emptyset$. When $e > n_i$, both $E[i, e]$ and $f[i, e]$ are empty. To simplify notation, we sometimes use $\ell[i, e]$ to denote the stage number of the last class and $f[i, e]$ to denote the set of stage numbers of the first classes in excursion $E[i, e]$.

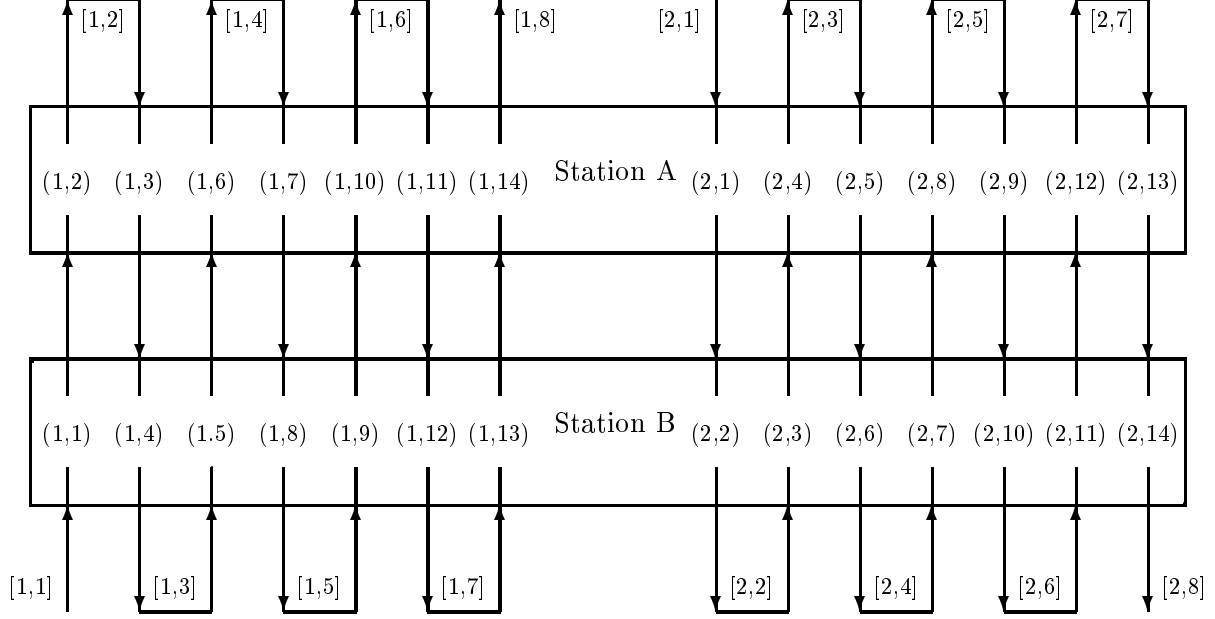


Figure 3: A twenty-eight class fluid network with two types of fluid

Definition 2.1. The neighbors of a set X of excursions is the set

$$\Gamma(X) = \{[i, e] \in E : [i, e - 1] \text{ or } [i, e + 1] \text{ is in } X\}.$$

Definition 2.2. A set S of excursions is said to be separating if it contains no consecutive excursions. Thus, a set S of excursions is separating if $S \cap \Gamma(S) = \emptyset$.

Definition 2.3. A separating set S is said to be A -strictly separating if it contains no first excursion at Station A , i.e., if $S \cap \{[i, 1] \in E_A^i : i \in I\} = \emptyset$. Similarly, a separating set S is said to be B -strictly separating if it contains no first excursion at Station B , i.e., if $S \cap \{[i, 1] \in E_B^i : i \in I\} = \emptyset$. A separating set S is said to be strictly separating if it contains no first excursion, i.e., if $S \subseteq \{[i, e] : i \in I, e = 2, \dots, n_i\}$.

The set of excursions at Station A , for example, is B -strictly separating. Likewise, the set of excursions at Station B is A -strictly separating. We refer to these two separating sets as *trivial* separating sets.

Each strictly separating set S of excursions induces a virtual station $V(S)$ or maximal collection of classes with the property that if we give highest priority to these classes the two servers can simultaneously serve classes of $V(S)$ only during a transient initial period.

Definition 2.4. Each separating set S of excursions induces a collection $V(S)$ consisting of the classes in excursions of S together with the first classes of excursions whose immediate predecessor is not in S . Thus,

$$V(S) = (\cup_{[i,e] \in S} E[i, e]) \cup (\cup_{[i,e] \in E^i \setminus S} f[i, e + 1]).$$

When S is strictly separating we refer to $V(S)$ as a virtual station.

A virtual station V , then, is a set of classes satisfying:

1. No class of a first excursion is in V , i.e., $E[i, 1] \cap V = \emptyset$ for each type i .
2. If the last class of an excursion is in V , then every class of that excursion is in V and if a first class of an excursion is in V , then every first class of that excursion is in V . Thus, a virtual station must have either none of the classes, all of the classes, or all but the last class of each excursion.
3. The last class of an excursion (except a last excursion) is in V if and only if no class of the next excursion is in V , i.e., for each excursion $e < n_i$, $\ell[i, e] \in V$ if and only if $E[i, e + 1] \cap V = \emptyset$.

In the network of Figure 1, the separating set $S = \{2, 5\}$ of excursions gives rise to the virtual station $V(S)$ consisting of classes 2 and 5 (there are no first classes in excursion 3). This is the only virtual station that is not itself a subset of the classes at a station.

The second phenomenon determining the global stability of a two-station fluid network is push starting. Giving highest priority to the first few classes of each type can magnify the effects of virtual stations in the subnetwork consisting of the remaining classes.

Definition 2.5. Let $\mathbf{e} = (e_i)_{i \in I}$ be a vector with $1 \leq e_i \leq n_i$ for each type i . We let $F^<(\mathbf{e})$ denote the push start set consisting of the collection of all classes up to but not including the last class of excursion $[i, e_i]$ for each type $i \in I$ and we let $R(\mathbf{e})$ denote all the remaining classes. Thus,

$$F^<(\mathbf{e}) = \{(i, k) : i \in I, 1 \leq k < \ell[i, e_i]\}$$

and

$$R(\mathbf{e}) = \{(i, k) : i \in I, \ell[i, e_i] \leq k \leq c_i\}.$$

We let $F^{\leq}(\mathbf{e})$ denote the collection of all classes up to and including the last class of excursion $[i, e_i]$ for each type $i \in I$. Thus,

$$F^{\leq}(\mathbf{e}) = \{(i, k) : i \in I, 1 \leq k \leq \ell[i, e_i]\}.$$

Note that if $V(S)$ is a virtual station and $F^<(\mathbf{e})$ is a push start set, then $V(S) \setminus F^{\leq}(\mathbf{e})$ is the classes of a virtual station in the subnetwork consisting of the classes of $R(\mathbf{e})$.

Given a set X of classes, we define X_A to be the classes of X served at Station A and X_B to be those served at station B . For example, we use $V_A(S)$ to denote the classes of the virtual station $V(S)$ at Station A and we use $F_A^<(\mathbf{e})$ to denote the classes of $F^<(\mathbf{e})$ at Station A . Further, to simplify our notation, we adopt the convention that for each set X of classes,

$$\lambda m(X) = \sum_{(i,k) \in X} \lambda_i m_k^i.$$

Theorem 2.1. A two-station fluid network is globally stable if and only if

$$(2.1) \quad \rho_A < 1, \quad \rho_B < 1,$$

and for each vector $\mathbf{e} = (e_i)_{i \in I}$ of excursions and each separating set S , we have

$$(2.2) \quad \frac{\lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e}))}{1 - \lambda m(F_A^<(\mathbf{e}))} + \frac{\lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}))}{1 - \lambda m(F_B^<(\mathbf{e}))} < 1.$$

Furthermore, if some vector $\mathbf{e} = (e_i)_{i \in I}$ of excursions and separating set S satisfy

$$(2.3) \quad \frac{\lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e}))}{1 - \lambda m(F_A^<(\mathbf{e}))} + \frac{\lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}))}{1 - \lambda m(F_B^<(\mathbf{e}))} > 1,$$

then there exists a non-idling fluid solution such that the WIP diverges to infinity with time.

We refer to the conditions (2.2) as the virtual workload conditions. When $F^{\prec}(\mathbf{e}) = \emptyset$, we refer to the virtual workload condition (2.2) as a virtual station condition. Otherwise, the condition involves push starting $F^{\prec}(\mathbf{e})$ and we refer to it as a push start condition. For example, the virtual workload conditions of the fluid network in Figure 2 are:

$$\begin{aligned} \lambda(m_2 + m_5 + m_7) &< 1, \\ \lambda(m_2 + m_4 + m_7) &< 1, \\ \frac{\lambda m_3}{1 - \lambda m_1} + \lambda m_6 &< 1, \\ \frac{\lambda m_4}{1 - \lambda m_2} + \frac{\lambda m_7}{1 - \lambda m_1} &< 1. \end{aligned}$$

The remainder of this paper is devoted to proving that the nominal workload conditions and the virtual workload conditions are necessary and sufficient to ensure the global stability of two-station fluid networks. We argue in Sections 3 and 4 that we can construct a certain piecewise linear Lyapunov function showing that the WIP will eventually go to zero and remain zero if the arrival rates and service rates satisfy certain constraints. We then argue in Sections 5 and 6 that these rather complicated constraints are equivalent to the virtual workload conditions.

In Section 7, we show that the virtual workload conditions (2.2) are necessary to ensure the global stability of the fluid network. We offer a direct proof that explicitly demonstrates a non-idling dispatch policy under which, if some vector $\mathbf{e} = (e_i)_{i \in I}$ of excursions and separating set S satisfy (2.3), WIP levels grow without bound. In fact, we demonstrate the trajectory of the fluid network through a finite sequence of states with greater and greater WIP in each successive cycle.

3 A Piecewise-Linear Lyapunov Function

We show that the virtual workload conditions of Theorem 2.1 are sufficient to ensure global stability of a two-station fluid network by showing that when they are satisfied there is a potential function or Lyapunov function G proving that the WIP drains to zero regardless of the initial conditions. Consider a fluid solution $(Q(\cdot), T(\cdot))$. We let $Z_k^i(t)$ denote the volume of fluid i that has already entered the network by time t , but has not yet received class (i, k) service, i.e.,

$$Z_k^i(t) = Z_k^i(0) + \lambda_i t - \mu_k^i T_k^i(t) = \sum_{\ell \leq k} Q_\ell^i(t).$$

We define G to be the maximum of two linear functions of $(Z_k^i(t))$ — one for each station — and so it is a piecewise-linear function of the buffer levels $(Q_k^i(t))$. In particular, given weights $x = (x_k^i)$ for the classes, we define the linear functions at Station A and Station B to be:

$$\begin{aligned} G_A(x, t) &= \sum_{i \in I} \sum_{k \in A_i} x_k^i Z_k^i(t) / \lambda_i \quad \text{and} \\ G_B(x, t) &= \sum_{i \in I} \sum_{k \in B_i} x_k^i Z_k^i(t) / \lambda_i. \end{aligned}$$

Let

$$G(x, t) = \max\{G_A(x, t), G_B(x, t)\}.$$

If there is $\epsilon > 0$ such that

- $G(x, t) > 0$ and
- $\dot{G}(x, t) \equiv \frac{\partial G(x, t)}{\partial t} \leq -\epsilon$,

whenever the WIP is not zero at time t and $Q(\cdot)$ and $G(x, \cdot)$ are differentiable at t , then after time $\tau = G(x, 0)/\epsilon$ all buffers will have drained to zero; proving that the fluid network is globally stable. Dai and Weiss (1996) showed that G will satisfy these conditions if there is $\epsilon > 0$ and weights $x > 0$ such that:

$$(3.1) \quad G_A(x, t) \leq G_B(x, t) \quad \text{whenever} \quad \sum_{i \in I} \sum_{k \in A_i} Q_k^i(t) = 0,$$

$$(3.2) \quad G_B(x, t) \leq G_A(x, t) \quad \text{whenever} \quad \sum_{i \in I} \sum_{k \in B_i} Q_k^i(t) = 0,$$

$$(3.3) \quad \frac{\partial G_A(x, t)}{\partial t} \leq -\epsilon \quad \text{whenever} \quad \sum_{i \in I} \sum_{k \in A_i} Q_k^i(t) > 0, \quad \text{and}$$

$$(3.4) \quad \frac{\partial G_B(x, t)}{\partial t} \leq -\epsilon \quad \text{whenever} \quad \sum_{i \in I} \sum_{k \in B_i} Q_k^i(t) > 0,$$

where conditions (3.1) and (3.2) apply for all t and conditions (3.3) and (3.4) apply only when t is a regular point of $(Q(\cdot), T(\cdot))$. Thus, we have the following proposition.

Proposition 3.1. *If there exists $\epsilon > 0$ and $x > 0$ such that (3.1)–(3.4) hold, the fluid network is stable under non-idling dispatch policies.*

4 A Linear Programming Formulation

We transform the problem of finding weights x such that $G_A(x, t)$ and $G_B(x, t)$ satisfy (3.1)–(3.4) into a linear programming problem. The linear program has a solution with strictly positive objective value if and only if the desired weights x exist and any solution with strictly positive objective value provides weights satisfying the desired conditions.

We first transform (3.1) into linear constraints on x . When

$$(4.1) \quad \sum_{i \in I} \sum_{k \in A_i} Q_k^i(t) = 0,$$

G_A reduces to:

$$(4.2) \quad \sum_{i \in I} \sum_{k \in A_i} \left(x_k^i \sum_{\ell \in B_i, \ell < k} Q_\ell^i(t) / \lambda_i \right) = \sum_{i \in I} \sum_{\ell \in B_i} \left(Q_\ell^i(t) \sum_{k \in A_i, k > \ell} x_k^i / \lambda_i \right)$$

and G_B reduces to:

$$(4.3) \quad \sum_{i \in I} \sum_{k \in B_i} \left(x_k^i \sum_{\ell \in B_i, \ell \leq k} Q_\ell^i(t) / \lambda_i \right) = \sum_{i \in I} \sum_{\ell \in B_i} \left(Q_\ell^i(t) \sum_{k \in B_i, k \geq \ell} x_k^i / \lambda_i \right).$$

It follows that (3.1) is satisfied if:

$$(4.4) \quad \sum_{k \in A_i, k > \ell} x_k^i \leq \sum_{k \in B_i, k \geq \ell} x_k^i$$

for each $i \in I$ and $\ell \in B_i$. Since the weights x are non-negative, we can restrict attention to those constraints of (4.4) where $\ell \in B_i$, but $\ell + 1 \notin B_i$. In other words, (4.4) is equivalent to

$$(4.5) \quad \sum_{k \in A_i, k > \ell[i, e]} x_k^i \leq \sum_{k \in B_i, k \geq \ell[i, e]} x_k^i$$

for each excursion $[i, e]$ at Station B .

Similar analysis leads to the conclusion that G_A and G_B satisfy (3.2) if:

$$(4.6) \quad \sum_{k \in B_i, k > \ell[i, e]} x_k^i \leq \sum_{k \in A_i, k \geq \ell[i, e]} x_k^i$$

for each excursion $[i, e]$ at Station A .

We next transform (3.3) into linear conditions on x . When

$$\sum_{i \in I} \sum_{k \in A_i} Q_k^i(t) > 0,$$

the non-idling condition (1.9) ensures that Station A is not accumulating idle time and so,

$$(4.7) \quad \sum_{i \in I} \sum_{k \in A_i} \dot{T}_k^i(t) = 1.$$

Now,

$$\begin{aligned} \dot{G}_A(t) &= \sum_{i \in I} \sum_{k \in A_i} x_k^i \dot{Z}_k^i(t) / \lambda_i \\ &= \sum_{i \in I} \sum_{k \in A_i} x_k^i \left(1 - \mu_k^i \dot{T}_k^i(t) / \lambda_i \right) \\ &= \sum_{i \in I} \sum_{k \in A_i} x_k^i - \sum_{i \in I} \sum_{k \in A_i} x_k^i \mu_k^i \dot{T}_k^i(t) / \lambda_i. \end{aligned}$$

Thus, (3.3) is satisfied if:

$$(4.8) \quad \sum_{j \in I} \sum_{k \in A_j} x_k^j + \epsilon \leq x_\ell^i \mu_\ell^i / \lambda_i$$

for each $i \in I$ and $\ell \in A_i$.

Similar analysis shows that (3.4) is satisfied if:

$$(4.9) \quad \sum_{j \in I} \sum_{k \in B_j} x_k^j + \epsilon \leq x_\ell^i \mu_\ell^i / \lambda_i$$

for each $i \in I$ and $\ell \in B_i$.

Finding the largest possible value of ϵ for which there are weights x satisfying (4.5)–(4.6) and

(4.8)–(4.9) reduces to solving the following linear program for ϵ and x :

$$(4.10) \quad \begin{aligned} & \text{maximize } \epsilon \\ & \text{subject to:} \end{aligned}$$

$$(4.11) \quad \sum_{k \in A_i, k > \ell[i, e]} x_k^i - \sum_{k \in B_i, k \geq \ell[i, e]} x_k^i \leq 0 \text{ for each } i \in I \text{ and } [i, e] \in E_B^i$$

$$(4.12) \quad \sum_{k \in B_i, k > \ell[i, e]} x_k^i - \sum_{k \in A_i, k \geq \ell[i, e]} x_k^i \leq 0 \text{ for each } i \in I \text{ and } [i, e] \in E_A^i$$

$$(4.13) \quad \left(\sum_{j \in I} \sum_{k \in A_j} x_k^j \right) - x_\ell^i \mu_\ell^i / \lambda_i + \epsilon \leq 0 \text{ for } i \in I \text{ and } \ell \in A_i$$

$$(4.14) \quad \left(\sum_{j \in I} \sum_{k \in B_j} x_k^j \right) - x_\ell^i \mu_\ell^i / \lambda_i + \epsilon \leq 0 \text{ for } i \in I \text{ and } \ell \in B_i$$

$$(4.15) \quad x, \epsilon \geq 0$$

The constraints (4.11)–(4.15) define a cone with the single extreme point given by $x = 0$ and $\epsilon = 0$. Thus, we have the following proposition:

Proposition 4.1. *If the linear program (4.10)–(4.15) has unbounded objective values, then each solution (x, ϵ) with $\epsilon > 0$ provides weights $x > 0$ such that $G(x, t)$ is a piecewise-linear Lyapunov function proving that the fluid network is stable.*

In Section 5, we transform the linear program (4.10)–(4.15) into a parametric network flow problem and, by exploiting a dual formulation, derive sufficient conditions for stability of a two-station fluid network. In Section 6, we show that these conditions are equivalent to the conditions of Theorem 2.1.

5 A Network Flows Formulation

The linear program (4.10)–(4.15) offers a computationally attractive method for determining whether or not a two-station fluid network with specified service times is globally stable. A network with given arrival rates and service times is globally stable if the linear program (4.10)–(4.15) has unbounded objective values. Otherwise, as we show in Section 7, it is not. The linear program does not, however, provide a theoretically attractive characterization of the global stability region for a two-station fluid network.

In order to obtain an explicit characterization of the arrival rates and service times under which a two-station fluid network is globally stable, we translate the linear program (4.10)–(4.15) into an equivalent parametric network flow problem. The parametric network flow problem is equivalent to the linear program in the sense that the linear program has unbounded objective values if and only if there is a value of the parameter for which the network flow problem has strictly positive objective value.

To transform (4.10)–(4.15) into an equivalent network flow problem, we first observe that since (4.11)–(4.15) defines a cone, there is a solution (x, ϵ) to (4.11)–(4.15) with $\epsilon > 0$ if and only if there is a solution with $\epsilon > 0$ and

$$(5.1) \quad \sum_{i \in I} \sum_{k \in A_i} x_k^i + \epsilon = 1.$$

Although we can arbitrarily scale the sum of the weights on classes at one station, (we have chosen Station A) to 1, we cannot simultaneously scale the sum of the weights on classes at the other station to a fixed value. Thus, we let β denote the sum of the weights on the classes served at Station B :

$$(5.2) \quad \sum_{i \in I} \sum_{k \in B_i} x_k^i + \epsilon = \beta.$$

We do not know a priori a value of β at which ϵ is maximized, but by treating it as a parameter rather than a variable, we can express the constraints (4.13)–(4.14) as lower bounds:

$$(5.3) \quad \lambda_i m_k^i \leq x_k^i \quad \text{for } i \in I \text{ and } k \in A_i$$

$$(5.4) \quad \beta \lambda_i m_k^i \leq x_k^i \quad \text{for } i \in I \text{ and } k \in B_i.$$

Next, we add slack variables $s = (s_e^i)$ and write the constraints (4.11)–(4.12) as:

$$(5.5) \quad \sum_{k \in A_i, k > \ell[i, e]} x_k^i - \sum_{k \in B_i, k \geq \ell[i, e]} x_k^i + s_e^i = 0 \quad \text{for each } i \in I \text{ and } [i, e] \in E_B^i$$

$$(5.6) \quad \sum_{k \in B_i, k > \ell[i, e]} x_k^i - \sum_{k \in A_i, k \geq \ell[i, e]} x_k^i + s_e^i = 0 \quad \text{for each } i \in I \text{ and } [i, e] \in E_A^i.$$

Adding (5.5) for an excursion $[i, e]$ at Station B and (5.6) for excursion $[i, e + 1]$ at Station A and multiplying by -1 , we obtain:

$$(5.7) \quad - \sum_{k \in f[i, e+1]} x_k^i + x_{\ell[i, e]}^i - s_e^i - s_{e+1}^i = 0.$$

Similarly, adding (5.6) for excursion $[i, e]$ at Station A and (5.5) for excursion $[i, e + 1]$ at Station B , we obtain

$$(5.8) \quad \sum_{k \in f[i, e+1]} x_k^i - x_{\ell[i, e]}^i + s_e^i + s_{e+1}^i = 0.$$

Adopting the convention that $s_{n_i+1}^i = 0$, we can write (5.5) for a last excursion $[i, n_i] \in E_B^i$ as:

$$- \sum_{k \in f[i, n_i+1]} x_k^i + x_{\ell[i, n_i]}^i - s_{n_i}^i - s_{n_i+1}^i = 0$$

and we can write (5.6) for a last excursion $[i, n_i] \in E_A^i$ as:

$$\sum_{k \in f[i, n_i+1]} x_k^i - x_{\ell[i, n_i]}^i + s_{n_i}^i + s_{n_i+1}^i = 0.$$

Combining these transformations gives the following linear program:

$$(5.9) \quad \begin{array}{l} \text{maximize } \epsilon \\ \text{subject to:} \end{array}$$

$$(5.10) \quad \sum_{k \in f[i, e+1]} x_k^i - x_{\ell[i, e]}^i + s_e^i + s_{e+1}^i = 0 \text{ for } i \in I \text{ and } [i, e] \in E_A^i$$

$$(5.11) \quad - \sum_{k \in f[i, e+1]} x_k^i + x_{\ell[i, e]}^i - s_e^i - s_{e+1}^i = 0 \text{ for } i \in I \text{ and } [i, e] \in E_B^i$$

$$(5.12) \quad \sum_{i \in I} \sum_{k \in A_i} x_k^i + \epsilon = 1$$

$$(5.13) \quad - \sum_{i \in I} \sum_{k \in B_i} x_k^i - \epsilon = -\beta$$

$$(5.14) \quad \lambda_i m_k^i \leq x_k^i \text{ for } i \in I \text{ and } k \in A_i$$

$$(5.15) \quad \beta \lambda_i m_k^i \leq x_k^i \text{ for } i \in I \text{ and } k \in B_i$$

$$(5.16) \quad x, s, \epsilon \geq 0.$$

There is a value of the parameter β such that the linear program (5.9)–(5.16) has optimum objective value $\epsilon > 0$ if and only if the linear program (4.10)–(4.15) has unbounded objective values.

The linear program (5.9)–(5.16) is a network flow problem with right-hand-sides and lower bounds that depend on the parameter β . The nodes of the network are:

- A node for each excursion $[i, e]$ corresponding to the constraints (5.10) and (5.11).
- A node for each Station A and B corresponding to the constraints (5.12) and (5.13).
- A node called *the root* corresponding to the redundant constraint

$$\sum_{i \in I} \left(\sum_{k \in B_i \cap f[i, 1]} x_k^i - \sum_{k \in A_i \cap f[i, 1]} x_k^i \right) + \sum_{i \in I: \ell[i, 1] \in B_i} s_1^i - \sum_{i \in I: \ell[i, 1] \in A_i} s_1^i = \beta - 1$$

obtained by adding (5.10)–(5.13) and multiplying by -1 .

The edges of the network are:

- E.1. An edge from the node for Station A to the node for excursion $[i, e]$ at Station A . This edge corresponds to the variable $x_{\ell[i, e]}^i$ and has lower bound $\lambda_i m_{\ell[i, e]}^i$.
- E.2. An edge from the node for excursion $[i, e]$ at Station B to the node for Station B . This edge corresponds to the variable $x_{\ell[i, e]}^i$ and has lower bound $\beta \lambda_i m_{\ell[i, e]}^i$.
- E.3. An edge from the node for Station A to the node for excursion $[i, e]$ at Station B for each class (i, k) in $f[i, e + 1]$. These edges correspond to the variables x_k^i for the classes in $f[i, e + 1]$. The edge for class (i, k) has lower bound $\lambda_i m_k^i$.
- E.4. An edge from the node for excursion $[i, e]$ at Station A to the node for Station B for each class (i, k) in $f[i, e + 1]$. These edges correspond to the variables x_k^i for the classes in $f[i, e + 1]$. The edge for class (i, k) has lower bound $\beta \lambda_i m_k^i$.

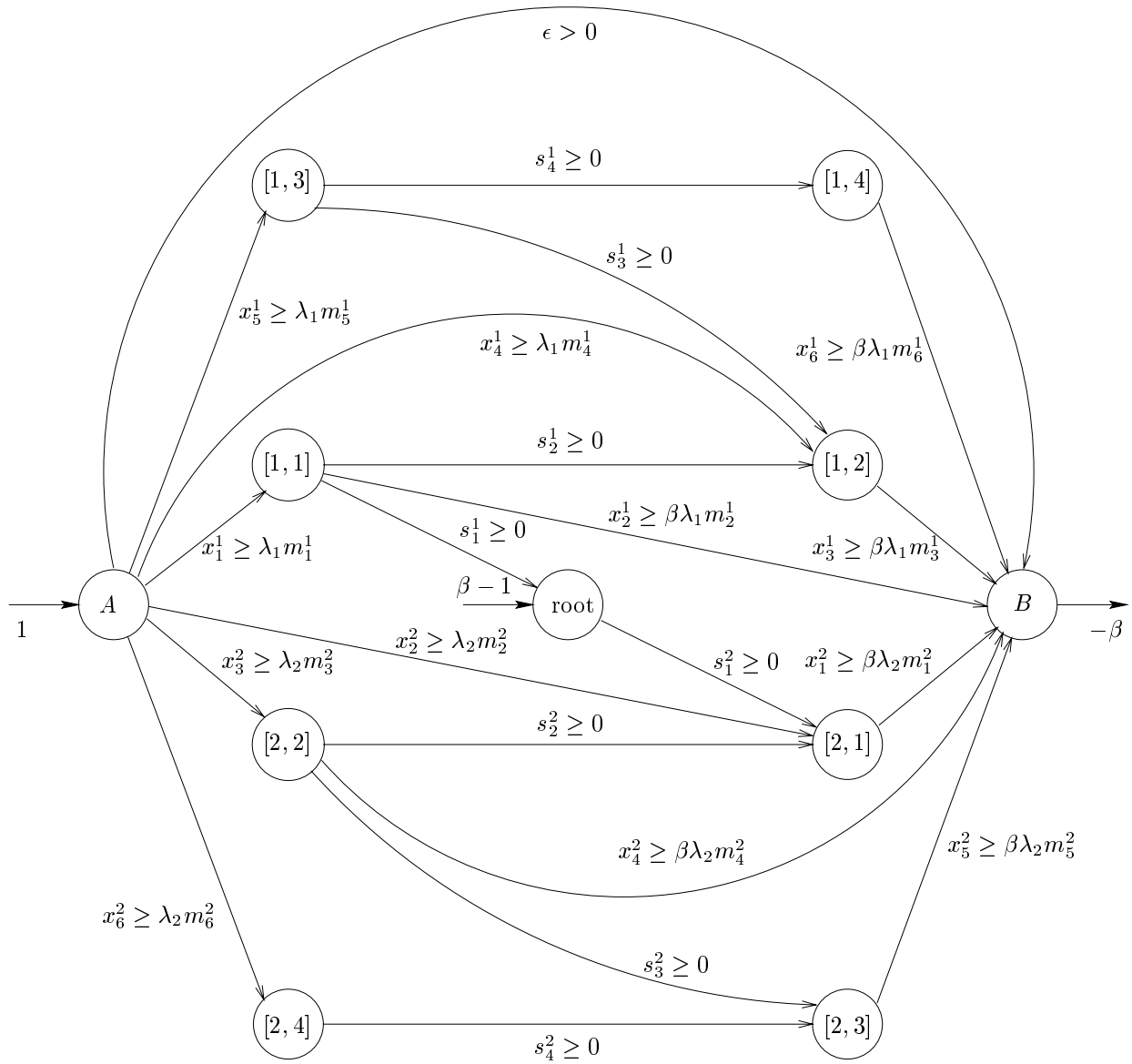


Figure 4: This parametric network flow problem is equivalent to the linear program (5.9)–(5.16) as applied to the fluid network in Figure 5. The source A has supply 1. The sink B has demand β . If $\beta > 1$, the root has supply $\beta - 1$. If $\beta \leq 1$, the root has demand $1 - \beta$.

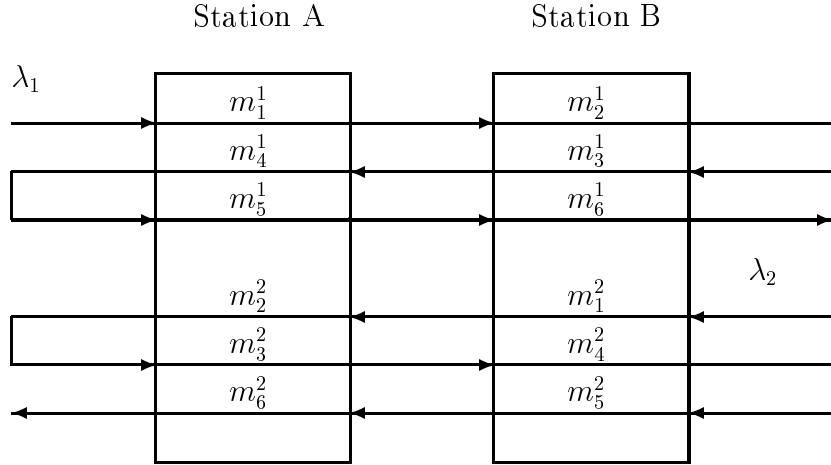


Figure 5: A twelve class fluid network with two types of fluid

- E.5. An edge from the node for Station A to the root for each class (i, k) in $f[i, 1]$ served at Station A . These edges correspond to the variables x_k^i for the classes in $f[i, 1]$ served at Station A . The edge for class (i, k) has lower bound $\lambda_i m_k^i$.
- E.6. An edge from the root to the node for Station B for each class (i, k) in $f[i, 1]$ served at Station B . These edges correspond to the variables x_k^i for the classes in $f[i, 1]$ served at Station B . The edge for class (i, k) has lower bound $\beta \lambda_i m_k^i$.
- E.7. An edge from the node for excursion $[i, 1]$ at Station A to the root. This edge corresponds to the variable s_1^i and has lower bound 0.
- E.8. An edge from the root to the node for excursion $[i, 1]$ at Station B . This edge corresponds to the variable s_1^i and has lower bound 0.
- E.9. An edge from the node for excursion $[i, e]$ at Station A to the node for excursion $[i, e - 1]$ at Station B . This edge corresponds to the variable s_e^i and has lower bound 0.
- E.10. An edge from the node for excursion $[i, e - 1]$ at Station A to the node for excursion $[i, e]$ at Station B . This edge corresponds to the variable s_e^i and has lower bound 0.
- E.11. An edge from the node for Station A to the node for Station B . This edge corresponds to the variable ϵ .

The node for A has a supply of 1 and the node for B has a demand for β . The remaining supply (if $\beta > 1$) or demand (if $\beta < 1$) is at the root. The linear program (4.10)–(4.15) has unbounded objective values if and only if there is a value of $\beta > 0$ such that there is a feasible flow in this network with $\epsilon > 0$. Figure 4, illustrates this construction for the fluid network in Figure 5.

Rather than consider separately the two cases $\beta > 1$ and $\beta < 1$, we model the supply or demand at the root by changing the supply at the node for A and the demand at the node for B to $\max\{1, \beta\}$ and adding two additional edges:

- An edge from the node for A to the root with lower bound $\max\{0, \beta - 1\}$ representing any supply at the root.

- An edge from the root to the node for B with lower bound $\max\{0, 1 - \beta\}$ representing any demand at the root.

The linear program (4.10)–(4.15) has unbounded objective values if and only if there is $\beta > 0$ such that there is a feasible flow with value $\max\{1, \beta\}$ and $\epsilon > 0$. Figure 6, illustrates this construction for the fluid network in Figure 5.

Finally, for given $\beta > 0$, there is a flow of value $\max\{1, \beta\}$ from A to B in this network (see, for example, Figure 6) with a strictly positive flow on the edge corresponding to ϵ if and only if the minimum flow from the node for A to the node for B in this network without the edge for ϵ (see, for example, Figure 7) is strictly less than $\max\{1, \beta\}$ — the remaining flow can be assigned to ϵ . Thus, we henceforth omit the edge for ϵ from the network and consider the resulting Minimum Flow Problem.

To summarize, given a feasible solution (x, ϵ) to the linear program (5.9)–(5.16) for some $\beta > 0$, we can construct a feasible solution to the minimum flow problem with value $\max\{1, \beta\} - \epsilon$ by sending $\max\{0, \beta - 1\}$ on the new edge from A to the root and $\max\{0, 1 - \beta\}$ on the new edge from the root to B . Conversely, given a feasible flow x for the minimum flow problem for some $\beta > 0$ with value $\max\{1, \beta\} - \epsilon$, (x, ϵ) is a feasible solution to the linear program (5.9)–(5.16).

From Theorem 1.1, the value of a minimum flow equals the capacity of a maximum A, B -cut and so, there are weights x satisfying (5.10)–(5.16) if and only if, for some value of $\beta > 0$, each A, B -cut in this network has capacity strictly less than $\max\{1, \beta\}$. Thus, we have proved the following lemma.

Lemma 5.1. *A two-station fluid network is globally stable if there is a value of $\beta > 0$ for which the capacity of a maximum A, B -cut is strictly less than $\max\{1, \beta\}$.*

Given an A, B -cut (L, R) , we let L_A denote the excursions in L that are served at Station A and L_B denote those served at Station B . Similarly, we let R_A denote the excursions in R served at Station A and R_B denote those at Station B .

We refer to an A, B -cut with the root in L as an L -cut. An A, B -cut with the root in R is an R -cut. Note that since the upper bound on each edge is infinite, an A, B -cut (L, R) in this network has capacity $-\infty$ if some edge extends from a node in R to a node in L . That is to say, an A, B -cut (L, R) in this network has finite capacity if and only if no edge extends from a node in R to a node in L , i.e., if and only if (L, R) satisfies:

Rule 1. If $[i, e] \in L_B$, then $[i, e + 1]$ is in L_A , otherwise the edge corresponding to the variable s_{e+1}^i (see E.9) extends from a node in R to a node in L ,

Rule 2. If $[i, e] \in R_A$, then excursion $[i, e + 1]$ is in R_B , otherwise the edge corresponding to the variable s_{e+1}^i (see E.10) extends from a node in R to a node in L ,

Rule 3. If (L, R) is an R -cut, then $[i, 1] \notin L_B$ for each type i , otherwise the edge corresponding to the variable s_1^i (see E.8) extends from a node in R to a node in L , and

Rule 4. If (L, R) is an L -cut, then $[i, 1] \notin R_A$ for each type i , otherwise the edge corresponding to the variable s_1^i (see E.7) extends from a node in R to a node in L .

Thus, we have the following lemma, which allows us to speak in terms of separating sets rather than cuts.

Lemma 5.2. *An L -cut (L, R) has finite capacity if and only if $L_B \cup R_A$ is an A -strictly separating set. Similarly, an R -cut (L, R) has finite capacity if and only if $L_B \cup R_A$ is a B -strictly separating set.*

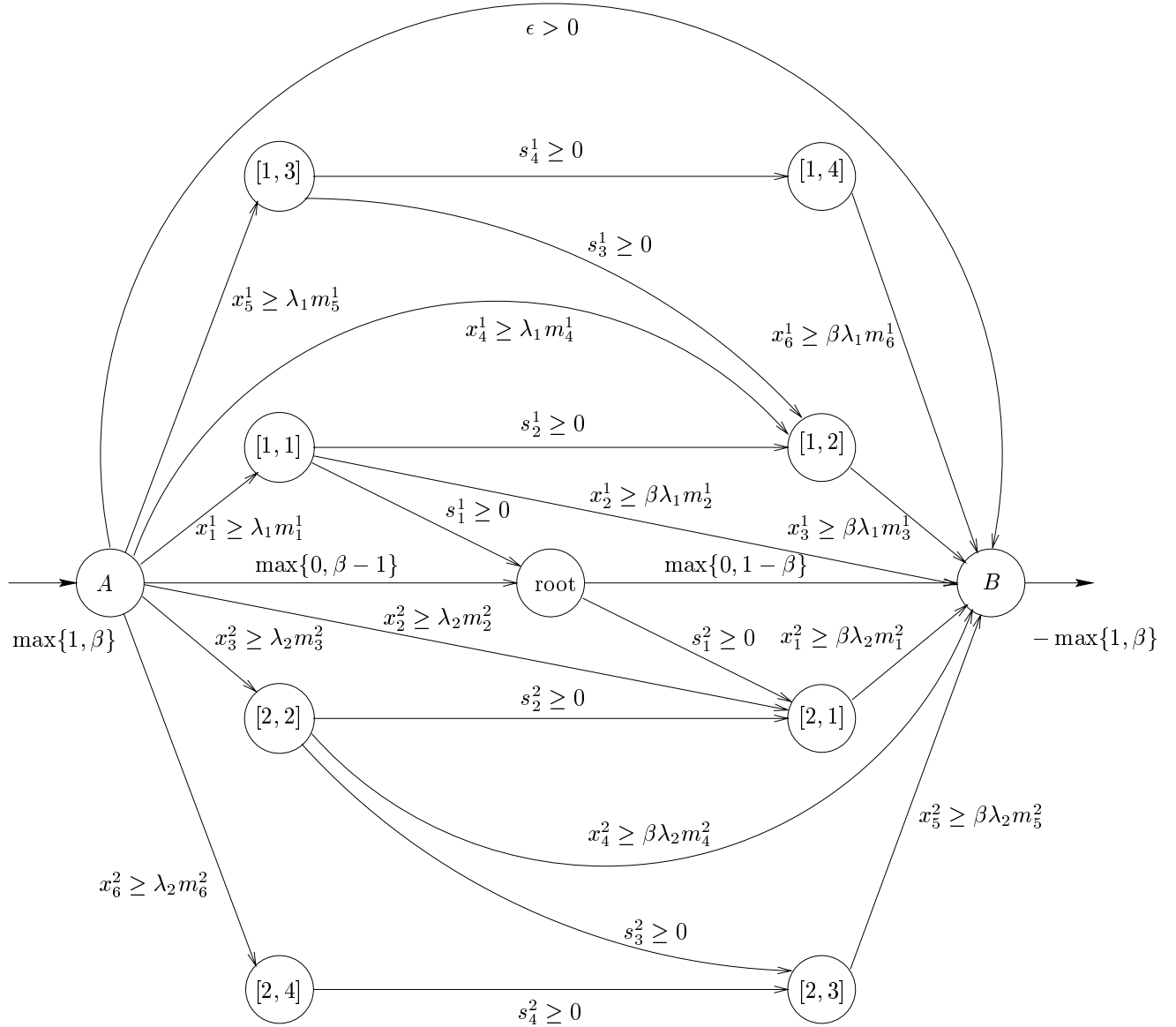


Figure 6: For given $\beta > 0$, there is a feasible flow in the network of Figure 4 if and only if there is a feasible flow in this network with value $\max\{1, \beta\}$ and $\epsilon > 0$.

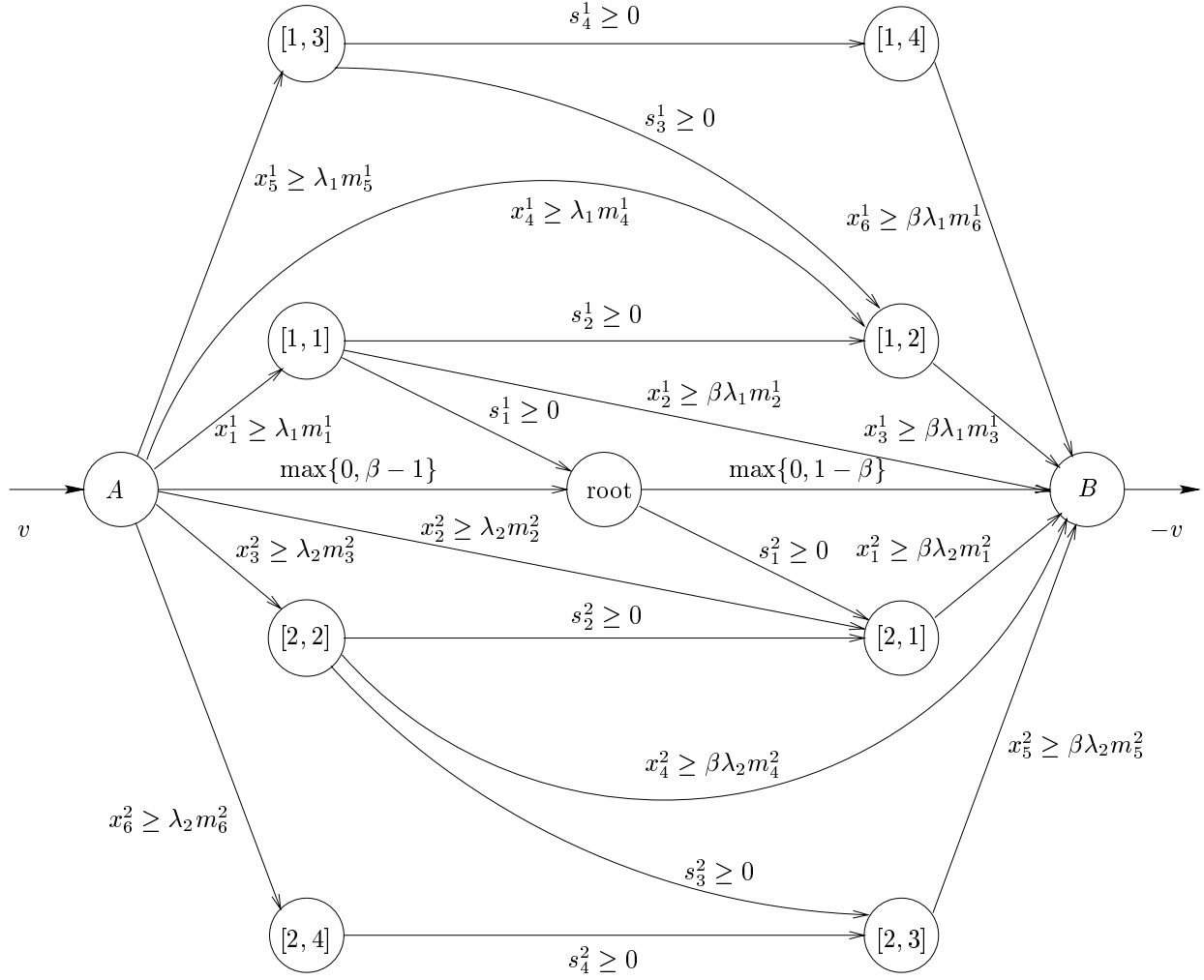


Figure 7: For given $\beta > 0$, there is a feasible flow in the network of Figure 6 with value $\max\{1, \beta\}$ and $\epsilon > 0$ if and only if the minimum flow in this network has value v strictly less than $\max\{1, \beta\}$.

We can express the capacity of a finite capacity L -cut (L, R) in terms of the corresponding A -strictly separating set $L_B \cup R_A$ and the collection $V(L_B \cup R_A)$ of classes it generates as follows:

$$c(L, R) = \lambda m(V_A(L_B \cup R_A)) + \beta \lambda m(V_B(L_B \cup R_A)) + \max\{0, 1 - \beta\}.$$

To see this observe that every edge contributing a positive amount to the capacity of an L -cut either starts at A or ends at B . The first term $\lambda m(V_A(L_B \cup R_A))$ captures the contributions of edges that start at A and the second term $\beta \lambda m(V_B(L_B \cup R_A))$ captures the contributions of edges that end at B except for the edge from the root to B , which contributes $\max\{0, 1 - \beta\}$. The edges starting at A that contribute to the capacity of the cut are:

- The last class of each excursion $[i, e] \in R_A$. These are the edges corresponding to edges E.1 that cross the cut.
- The first classes of each excursion $[i, e] \in A$ such that $[i, e - 1] \in R_B$. These are the edges corresponding to edges E.3 that cross the cut.

These are exactly the classes of $V_A(L_B \cup R_A)$. The classes of the virtual station served at A include all the classes of R_A and the first classes of those excursions $[i, e] \in L_A$ such that $[i, e - 1] \notin L_B$. If $[i, e] \in R_A$, then, by Rule 1, $[i, e - 1] \in R_B$ and so all the classes of the excursion cross the cut. If $[i, e] \in L_A$ and $[i, e - 1] \in R_B$ then all the first classes of the excursion cross the cut, but not the last class. Analogous arguments verify that $\beta \lambda m(V_B(L_B \cup R_A))$ is exactly the contribution of edges that end at B crossing the cut except for the edge from the root to B whose contribution is captured in the third term.

Thus, assuming the service times satisfy the nominal workload conditions, each finite capacity L -cut (L, R) imposes the condition:

$$(5.17) \quad \frac{\lambda m(V_A(L_B \cup R_A))}{1 - \lambda m(V_B(L_B \cup R_A))} < \beta$$

on β . One exception to (5.17) arises when R_A is empty and L_B consists of all the excursions at Station B , i.e., when $L_B \cup R_A$ is the trivial separating set consisting of all excursions at Station B . In this case, $V_A(L_B \cup R_A) = \emptyset$ and the condition $c(L, R) < 1$ reduces to the nominal workload condition at Station B .

Similarly, we can express the capacity of a finite capacity R -cut (L, R) in terms of the corresponding B -strictly separating set $L_B \cup R_A$ and the collection $V(L_B \cup R_A)$ of classes it generates as follows:

$$c(L, R) = \lambda m(V_A(L_B \cup R_A)) + \beta \lambda m(V_B(L_B \cup R_A)) + \max\{0, \beta - 1\}.$$

Thus, each finite capacity R -cut (L, R) imposes the condition:

$$(5.18) \quad \frac{1 - \lambda m(V_A(L_B \cup R_A))}{\lambda m(V_B(L_B \cup R_A))} > \beta$$

on β . One exception to (5.18) arises when L_B is empty and R_A consists of all the excursions at Station A , i.e., when $L_B \cup R_A$ is the trivial separating set consisting of all excursions at Station A . In this case, $V_B(L_B \cup R_A) = \emptyset$ and the condition $c(L, R) < 1$ reduces to the nominal workload condition at Station A .

Combining (5.17) and (5.18) with Lemma 5.2 proves the following theorem, which provides explicit constraints on the service times sufficient to ensure the global stability of a two-station fluid network. In Section 6, we show that these ‘‘cut conditions’’ are equivalent to the virtual workload conditions, which we prove are also necessary to ensure global stability.

Theorem 5.1. *A two-station fluid network with service times m and arrival rates $\lambda = (\lambda_i)_{i \in I}$ satisfying the nominal workload conditions is globally stable if (1.13) holds and for each non-trivial A -strictly separating set S' and non-trivial B -strictly separating set S ,*

$$(5.19) \quad \frac{\lambda m(V_A(S'))}{1 - \lambda m(V_B(S'))} < \frac{1 - \lambda m(V_A(S))}{\lambda m(V_B(S))}.$$

6 Sufficiency

In Section 5, we showed that a two-station fluid network satisfying the nominal workload conditions is globally stable *if* the arrival rates and service times satisfy the cut conditions. We show that the virtual workload conditions are also sufficient to ensure global stability by showing that the arrival rates and service times satisfy the cut conditions if they satisfy the virtual workload conditions.

Theorem 6.1. *A two-station fluid network satisfying the nominal workload conditions (1.13) is globally stable if for each vector $\mathbf{e} = (e_i)_{i \in I}$ of excursions and each separating set S , we have*

$$(6.1) \quad \frac{\lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e}))}{1 - \lambda m(F_A^{\leq}(\mathbf{e}))} + \frac{\lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}))}{1 - \lambda m(F_B^{\leq}(\mathbf{e}))} < 1.$$

Proof. Each cut condition (5.19) is defined by a pair of non-trivial separating sets: an A -strictly separating set S' and a B -strictly separating set S . We show that the cut condition induced by the pair (S', S) is implied by a pair of virtual workload conditions.

For each type $i \in I$, let e_i be the largest index such that:

- a. Every earlier excursion served at Station B is in S' (and hence no earlier excursion served at Station A is in S'),
- b. Every earlier excursion served at Station A is in S (and hence no earlier excursion served at Station B is in S).

Note that $[i, e_i] \notin S$. To see this observe that if $[i, e_i] \in S$, then it must be in E_A^i and since every earlier excursion served at Station A is in S and every earlier excursion served at Station B is in S' , either S is trivial or $e_i + 1$ satisfies (a) and (b). A similar argument shows that $[i, e_i] \notin S'$.

The vector $\mathbf{e} = (e_i)_{i \in I}$ of excursions and the separating set S' induce the virtual workload condition:

$$(6.2) \quad \frac{\lambda m(V_A(S') \setminus F_A^{\leq}(\mathbf{e}))}{1 - \lambda m(F_A^{\leq}(\mathbf{e}))} + \frac{\lambda m(V_B(S') \setminus F_B^{\leq}(\mathbf{e}))}{1 - \lambda m(F_B^{\leq}(\mathbf{e}))} < 1.$$

Similarly, the vector \mathbf{e} and the separating set S induce the virtual workload condition:

$$(6.3) \quad \frac{\lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e}))}{1 - \lambda m(F_A^{\leq}(\mathbf{e}))} + \frac{\lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}))}{1 - \lambda m(F_B^{\leq}(\mathbf{e}))} < 1.$$

We show that (6.2) and (6.3) imply the cut condition for the pair (S', S) .

From (6.2) we have that

$$\frac{\lambda m(V_A(S') \setminus F_A^{\leq}(\mathbf{e}))}{1 - \lambda m(F_B^{\leq}(\mathbf{e})) - \lambda m(V_B(S') \setminus F_B^{\leq}(\mathbf{e}))} < \frac{1 - \lambda m(F_A^{\leq}(\mathbf{e}))}{1 - \lambda m(F_B^{\leq}(\mathbf{e}))}$$

and from (6.3) we have that

$$\frac{1 - \lambda m(F_A^<(\mathbf{e})) - \lambda m(V_A(S) \setminus F_A^<(\mathbf{e}))}{\lambda m(V_B(S) \setminus F_B^<(\mathbf{e}))} > \frac{1 - \lambda m(F_A^<(\mathbf{e}))}{1 - \lambda m(F_B^<(\mathbf{e}))}.$$

Thus,

$$(6.4) \quad \frac{1 - \lambda m(F_A^<(\mathbf{e})) - \lambda m(V_A(S) \setminus F_A^<(\mathbf{e}))}{\lambda m(V_B(S) \setminus F_B^<(\mathbf{e}))} > \frac{\lambda m(V_A(S') \setminus F_A^<(\mathbf{e}))}{1 - \lambda m(F_B^<(\mathbf{e})) - \lambda m(V_B(S') \setminus F_B^<(\mathbf{e}))}.$$

Now, since $[i, e_i] \notin S$ for each $i \in I$,

$$\lambda m(V_A(S) \setminus F_A^<(\mathbf{e})) + \lambda m(F_A^<(\mathbf{e})) \geq \lambda m(V_A(S)).$$

Further, since e_i satisfies (b),

$$\lambda m(V_B(S) \setminus F_B^<(\mathbf{e})) = \lambda m(V_B(S)).$$

Likewise, since $[i, e_i] \notin S'$ for each $i \in I$,

$$\lambda m(V_B(S') \setminus F_B^<(\mathbf{e})) + \lambda m(F_B^<(\mathbf{e})) \geq \lambda m(V_B(S')).$$

And, since e_i satisfies (a),

$$\lambda m(V_A(S') \setminus F_A^<(\mathbf{e})) = \lambda m(V_A(S')).$$

Thus, (6.4) implies that

$$\frac{1 - \lambda m(V_A(S))}{\lambda m(V_B(S))} > \frac{\lambda m(V_A(S'))}{1 - \lambda m(V_B(S'))},$$

which is exactly the cut condition for the pair S' and S . □

7 Necessity

In this section, we show that the virtual workload conditions (2.2) are necessary to ensure the global stability of a two-station fluid network.

We first generalize to virtual stations the argument used to show the necessity of the nominal workload conditions to ensure global stability.

Lemma 7.1. *Let C be a set of classes such that*

$$\lambda m(C) \geq 1.$$

Each non-idling fluid solution $(Q(\cdot), T(\cdot))$ satisfying

$$(7.1) \quad \sum_{(i,k) \in C} \dot{T}(t) \leq 1$$

for each regular point t is unstable.

Proof. Consider a non-idling fluid solution $(Q(\cdot), T(\cdot))$ satisfying (1.5)–(1.10) and (7.1). Define

$$(7.2) \quad W(Q(t)) = \sum_{(i,k) \in C} m_k^i Z_k^i(t) = \sum_{(i,k) \in C} m_k^i \sum_{\ell=1}^k Q_\ell^i(t)$$

to be the total workload for the classes of C . Since

$$Z_k^i(t) = Z_k^i(0) + \lambda_i t - T_k^i(t)/m_k^i,$$

$$\begin{aligned} W(Q(t)) &= W(Q(0)) + \sum_{(i,k) \in C} \lambda_i m_k^i t - \sum_{(i,k) \in C} T_k^i(t) \\ &\geq W(Q(0)) + (\lambda m(C) - 1) t. \end{aligned}$$

Hence the workload for the classes of C grows linearly with time and the fluid solution diverges. \square

The classes of any set C satisfying (7.1) can be viewed as being served by a single “virtual” server, which allocates its efforts among them. We show that under the appropriate static buffer priority policy, a virtual station as defined in Definition 2.4 satisfies (7.1), hence the moniker “virtual station”.

We next show how expediting the first few classes magnifies the influence of virtual stations in the remaining network. This concept was originally introduced under the rubric of push starting in Dai and Vande Vate (1996).

Let $\mathbf{e} = (e_i)_{i \in I}$ be a vector of excursions, one for each type. The vector \mathbf{e} partitions the classes of the network into the classes of $F^<(\mathbf{e})$ and the remainder of the classes, which we denote by $R(\mathbf{e})$. Let

$$(7.3) \quad \tilde{m}_k^i = m_k^i / (1 - \lambda m(F_A^<(\mathbf{e}))) \text{ for } (i, k) \in R_A(\mathbf{e}),$$

$$(7.4) \quad \tilde{m}_k^i = m_k^i / (1 - \lambda m(F_B^<(\mathbf{e}))) \text{ for } (i, k) \in R_B(\mathbf{e}).$$

Consider the induced fluid model on the classes of $R(\mathbf{e})$:

$$(7.5) \quad Q_k^i(t) = Q_k^i(0) + \tilde{\mu}_{k-1}^i T_{k-1}^i(t) - \tilde{\mu}_k^i T_k^i(t) \geq 0, \quad t \geq 0, \quad (i, k) \in R(\mathbf{e}),$$

$$(7.6) \quad T_k^i(0) = 0 \text{ and } T_k^i(\cdot) \text{ is nondecreasing,} \quad (i, k) \in R(\mathbf{e}),$$

$$(7.7) \quad t - \sum_{(i,k) \in R_A(\mathbf{e})} T_k^i(t) \text{ is nondecreasing,}$$

$$(7.8) \quad t - \sum_{(i,k) \in R_B(\mathbf{e})} T_k^i(t) \text{ is nondecreasing,}$$

$$(7.9) \quad \sum_{(i,k) \in R_A(\mathbf{e})} \dot{T}_k^i(t) = 1 \text{ whenever } \sum_{(i,k) \in R_A(\mathbf{e})} Q_k^i(t) > 0 \text{ and } t \text{ is a regular point,}$$

$$(7.10) \quad \sum_{(i,k) \in R_B(\mathbf{e})} \dot{T}_k^i(t) = 1 \text{ whenever } \sum_{(i,k) \in R_B(\mathbf{e})} Q_k^i(t) > 0 \text{ and } t \text{ is a regular point,}$$

where, $\tilde{\mu}_k^i = 1/\tilde{m}_k^i$ for $(i, k) \in R(\mathbf{e})$. Note that for each type $i \in I$, $\ell[i, e_i]$ is the index of the first class of type i in $R(\mathbf{e})$. Thus, for each type $i \in I$, we let $\tilde{\mu}_{\ell[i, e_i-1]}^i = \lambda_i$ and $T_{\ell[i, e_i-1]}^i(t) = t$ to model the arrivals to the induced fluid network.

Lemma 7.2. *If the fluid model (7.5)–(7.10) is unstable, then the fluid model (1.5)–(1.10) is unstable.*

We leave the proof of this lemma to the appendix.

Proof of Theorem 2.1. In light of Lemma 7.2, it is enough to show that if the virtual station $V(S)$ corresponding to some strictly separating set S satisfies

$$(7.11) \quad \lambda m(V(S)) \geq 1,$$

then there is an unstable fluid solution. Because $V(S)$ is a virtual station in the corresponding queueing network, there is a static buffer priority discipline under which no two classes in $V(S)$ can be served simultaneously; see Dai and Vande Vate (1996). Therefore, any fluid limit $(Q(\cdot), T(\cdot))$ as defined in Dai (1996) is a fluid solution that satisfies (7.1). By Lemma 7.1, the fluid network is not globally stable. When (2.3) is satisfied or (7.11) is strictly satisfied, the same argument shows that the WIP goes to infinity. \square

8 Unstable Cycles

Although the preceding proof of Theorem 2.1 is succinct, it involves a rather circuitous argument via queueing networks and fluid limits. We provide a more direct, but somewhat longer argument establishing the necessity of the virtual workload conditions to ensure stability of a two-station fluid network. When the arrival rates and service times satisfy the nominal workload conditions, but do not satisfy the virtual workload conditions, we provide an explicit construction of a non-idling fluid solution $(Q(\cdot), T(\cdot))$ that is unstable. This argument is not only more direct, but also illustrates how the work-in-process in an unstable system swings from station to station as it grows to infinity. Iterations like those presented here are indispensable when studying the stability of fluid networks with more than two stations. See, for example, Dai, Hasenbein and Vande Vate (1998).

Suppose the arrival rates and service times satisfy the nominal workload conditions, but do not satisfy the virtual workload conditions. We choose a strictly separating set S and a push start set $F^<(\mathbf{e})$ so that among all such pairs,

$$\frac{\lambda m(V_A(S) \setminus F_A^<(\mathbf{e}))}{1 - \lambda m(F_A^<(\mathbf{e}))} + \frac{\lambda m(V_B(S) \setminus F_B^<(\mathbf{e}))}{1 - \lambda m(F_B^<(\mathbf{e}))}$$

is maximum. Thus, we assume that

$$(8.1) \quad \frac{\lambda m(V_A(S) \setminus F_A^<(\mathbf{e}))}{1 - \lambda m(F_A^<(\mathbf{e}))} + \frac{\lambda m(V_B(S) \setminus F_B^<(\mathbf{e}))}{1 - \lambda m(F_B^<(\mathbf{e}))} \geq 1$$

and for each strictly separating set S' and push start set $F^<(\mathbf{e}')$,

$$(8.2) \quad \frac{\lambda m(V_A(S) \setminus F_A^<(\mathbf{e}))}{1 - \lambda m(F_A^<(\mathbf{e}))} + \frac{\lambda m(V_B(S) \setminus F_B^<(\mathbf{e}))}{1 - \lambda m(F_B^<(\mathbf{e}))} \geq \frac{\lambda m(V_A(S') \setminus F_A^<(\mathbf{e}'))}{1 - \lambda m(F_A^<(\mathbf{e}'))} + \frac{\lambda m(V_B(S') \setminus F_B^<(\mathbf{e}'))}{1 - \lambda m(F_B^<(\mathbf{e}'))}.$$

Note that since $e_i \geq 1$ for each type i , the set $F^<(\mathbf{e})$ will include all the classes of every first excursion. Thus we may, without loss of generality, restrict attention to strictly separating sets S as we have above.

In order to construct an unstable fluid solution, we must consider in greater detail the structure of the strictly separating set S . The notation required to describe this dissection of strictly separating sets in full generality is somewhat involved, but the ideas are relatively straightforward. We first partition S into *monotype* separating sets or separating sets consisting of excursions for a single type of fluid. In particular, we let $S^i = S \cap E^i$, be the excursions of S for fluid type i .

We further partition the union of each monotype separating set and its neighbors into *intervals* or sets of consecutive excursions. We use $[i, s, t)$ to denote the interval $\{[i, e] \in E^i : s \leq e < t\}$ and (i, s, t) to denote the interval $\{[i, e] \in E^i : s < e < t\}$.

A maximal interval with the property that all its excursions at one station are in S^i and, consequently, none of its excursions at the other station are in S^i is called a *section* of S^i . We refer to a section X of S^i with $X_A \subseteq S^i$ and $X \cap S_B^i = \emptyset$ as an *A-section* of S^i . Similarly, we refer to a section Y of S^i with $Y_B \subseteq S^i$ and $Y \cap S_A^i = \emptyset$ as a *B-section* of S^i . A typical *A-section* is of the form $[i, s, t)$ with $s \in E_B^i$. The ending excursion t is either in E_A^i or $t = n_i + 1$. Similarly a typical *B-section* is of the form (i, s, t) with $s \in E_A^i$, and $t \in E_B^i$ or $t = n_i + 1$. In either case, the end excursions s and t are not in the separating set S and $s - 1 \notin S$.

For example, consider the strictly separating set

$$(8.3) \quad S = \{[1, 2], [1, 7], [2, 3], [2, 5], [2, 8]\}$$

in Figure 3. Following the definition of virtual station in Definition 2.4,

$$\begin{aligned} V(S) = & \{(1, 2), (1, 3), (1, 6), (1, 8), (1, 10), (1, 12), (1, 13), \\ & (2, 2), (2, 4), (2, 5), (2, 8), (2, 9), (2, 12), (2, 14)\}. \end{aligned}$$

The intervals $[1, 1, 4) = \{[1, 1], [1, 2], [1, 3]\}$ and $[2, 2, 7) = \{[2, 2], [2, 3], [2, 4], [2, 5], [2, 6]\}$ of excursions are the *A-sections* of this separating set. The *B-sections* are $[1, 6, 9) = \{[1, 6], [1, 7], [1, 8]\}$ and $[2, 7, 9) = \{[2, 7], [2, 8]\}$.

The sections of S^i partition $S^i \cup \Gamma(S^i)$ into intervals. We partition the remaining excursions of E^i into *trivial sections*. In particular, each excursion of $E^i \setminus (S^i \cup \Gamma(S^i))$ forms a trivial section of S^i . Each excursion $[i, e] \in E_A^i \setminus (S^i \cup \Gamma(S^i))$ forms a trivial *B-section* of S^i (because $\{[i, e]\} \cap E_B^i = \emptyset \subseteq S^i$ and $\{[i, e]\} \cap S_A^i = \emptyset$). Likewise, each excursion $[i, e] \in E_B^i \setminus (S^i \cup \Gamma(S^i))$ forms a trivial *A-section* of S^i . For example, $[1, 5, 6) = \{[1, 5]\}$ is a trivial *A-section* and $[1, 4, 5) = \{[1, 4]\}$ and $[2, 1, 2) = \{[2, 1]\}$ are the trivial *B-sections* of the separating set (8.3) in Figure 3.

When we include the trivial sections, the sections of S^i partition the excursions of E^i into intervals and, if we order these intervals in the natural way, they alternate between *A-sections* and *B-sections*.

Our construction relies on coordinating the activities of the servers across classes related to, but offset from, the classes of each section. In particular, we associate with each section $[i, s, t)$ the collection of classes

$$C([i, s, t)) = \{\ell[i, s]\} \cup_{e \in (i, s, t)} E[i, e] \cup f[i, t]$$

called a *block*. Note that the block $C([i, s, t))$ differs from the classes of the section $[i, s, t)$ in that we omit the first classes of the first excursion in $[i, s, t)$ and we add the first classes of the first excursion of the next section. In the example of Figure 3, $C([1, 1, 4)) = \{(i, k) : 1 \leq k \leq 6\}$ and $C([2, 2, 7)) = \{(2, k) : 3 \leq k \leq 12\}$.

The following lemma should help motivate our definition of sections and blocks. Its proof is postponed to the appendix.

Lemma 8.1. *Let S be a strictly separating set and $F^{<}(\mathbf{e})$ a push-start set satisfying (8.1) and (8.2). Then for each A -section $X = [i, s, t]$ of S^i , where $s \in E_B^i$ and $t \in E_A^i$,*

$$(8.4) \quad \frac{\lambda m(C_A(X) \setminus F_A^{<}(\mathbf{e}))}{1 - \lambda m(F_A^{<}(\mathbf{e}))} \geq \frac{\lambda m(C_B(X) \setminus F_B^{<}(\mathbf{e}))}{1 - \lambda m(F_B^{<}(\mathbf{e}))}.$$

Likewise, for each B -section $Y = [i, s, t]$ of S^i , where $s \in E_A^i$ and $t \in E_B^i$,

$$(8.5) \quad \frac{\lambda m(C_B(Y) \setminus F_B^{<}(\mathbf{e}))}{1 - \lambda m(F_B^{<}(\mathbf{e}))} \geq \frac{\lambda m(C_A(Y) \setminus F_A^{<}(\mathbf{e}))}{1 - \lambda m(F_A^{<}(\mathbf{e}))}.$$

The transformations (7.3) – (7.4) allow us to assume, without loss of generality, that $F^{<}(\mathbf{e})$ is empty. Implicit in this is the assumption that for each type $i \in I$, $e_i = 1$ and the first excursion consists only of the the class $(i, 1)$.

Let I_A be the set of types with first excursion $[i, 1] \in E_A^i$ and I_B the types with first excursion $[i, 1] \in E_B^i$. For $i \in I_A$, the sections of S^i alternate between B -sections and A -sections beginning with a B -section. We denote these sections as:

$$Y_1^i, X_2^i, Y_3^i, X_4^i, \dots, Y_{2b_i-1}^i, X_{2b_i}^i,$$

where b_i is the total number of B -sections and $X_{2b_i}^i$ is possibly empty. Similarly, for $i \in I_B$, the sections of S^i alternate between A -sections and B -sections beginning with an A -section. We denote these sections as:

$$X_1^i, Y_2^i, X_3^i, Y_4^i, \dots, X_{2b_i-1}^i, Y_{2b_i}^i,$$

with $Y_{2b_i}^i$ possibly empty. The sections Y_1^i and X_1^i are called *input sections*.

In the example of Figure 3, type 1 is in I_B and type 2 is in I_A . The sections for type 1 are:

$$X_1^1 = [1, 1, 4], Y_2^1 = [1, 4, 5], X_3^1 = [1, 5, 6], Y_4^1 = [1, 6, 9].$$

The sections for type 2 are:

$$Y_1^2 = [2, 1, 2), X_2^2 = [2, 2, 7), Y_3^2 = [2, 7, 9).$$

Therefore,

$$\begin{aligned} C(X_1^1) &= \{(1, k) : k = 1, \dots, 6\}, \\ C(Y_2^1) &= \{(1, 7), (1, 8)\}, \\ C(X_3^1) &= \{(1, 9), (1, 10)\}, \\ C(Y_4^1) &= \{(1, 11), (1, 12), (1, 13), (1, 14)\}, \\ C(Y_1^2) &= \{(2, 1), (2, 2)\}, \\ C(X_2^2) &= \{(2, k) : k = 3, \dots, 12\}, \\ C(Y_3^2) &= \{(2, 13), (2, 14)\}. \end{aligned}$$

The following lemma is a direct consequence of our definitions.

Lemma 8.2. *The virtual station $V(S)$ has the following decomposition.*

$$V(S) = \cup_{i \in I_A} \cup_{r=1}^{b_i} (C_B(Y_{2r-1}^i) \cup C_A(X_{2r}^i)) \cup_{i \in I_B} \cup_{r=1}^{b_i} (C_A(X_{2r-1}^i) \cup C_B(Y_{2r}^i)).$$

Let us restate part of Theorem 2.1.

Theorem 8.1. *Assume that there is a strictly separating set S and push start set $F^{<}(\mathbf{e})$ such that (8.1) and (8.2) hold. One can construct an unstable non-idling fluid solution $(Q(\cdot), T(\cdot))$. Furthermore, the unstable fluid solution satisfies the static buffer priority conditions (1.11)–(1.12).*

Proof. In light of Lemma 7.2, we may assume that the push start set $F^{<}(\mathbf{e})$ is empty. This would be the case, for example, if $e_i = 1$ and $f[i, 1] = \emptyset$ for each type $i \in I$, which is true of the induced network on $R(\mathbf{e})$ obtained from the construction used in the proof of Lemma 7.2. Thus, we assume that the arrival rates and service times satisfy the nominal workload conditions, but violate the virtual workload condition for a strictly separating set S , i.e.,

$$\lambda m(V(S)) \geq 1.$$

Under this transformation, (8.4) reduces to

$$(8.6) \quad \lambda m(C_A(X)) \geq \lambda m(C_B(X))$$

for each A -section X , and (8.5) reduces to

$$(8.7) \quad \lambda m(C_B(Y)) \geq \lambda m(C_A(Y))$$

for each B -section Y .

For the example in Figure 3, we have

$$\lambda_1(m_2^1 + m_3^1 + m_6^1 + m_8^1 + m_{10}^1 + m_{12}^1 + m_{13}^1) + \lambda_2(m_2^2 + m_4^2 + m_5^2 + m_8^2 + m_9^2 + m_{12}^2 + m_{14}^2) \geq 1,$$

and

$$\begin{aligned} m_2^1 + m_3^1 + m_6^1 &\geq m_1^1 + m_4^1 + m_5^1, \\ m_8^1 &\geq m_7^1, \\ m_9^1 &\geq m_{10}^1, \\ m_{12}^1 + m_{13}^1 &\geq m_{11}^1 + m_{14}^1, \\ m_2^2 &\geq m_1^2, \\ m_4^2 + m_5^2 + m_8^2 + m_9^2 + m_{12}^2 &\geq m_3^2 + m_6^2 + m_7^2 + m_{10}^2 + m_{11}^2, \\ m_{14}^2 &\geq m_{13}^2. \end{aligned}$$

We now construct a non-idling fluid solution $(Q(\cdot), T(\cdot))$ such that

$$\sum_{(i,k) \in V(S)} \dot{T}_k^i(t) \leq 1$$

for each regular point t . Let

$$\theta_k^i = \dot{T}_k^i(t).$$

We intentionally drop the variable t from θ_k^i since, in our construction, $T_k^i(\cdot)$ is piecewise linear and hence θ_k^i is piecewise constant. If the fluid solution $(Q(\cdot), T(\cdot))$ is linear in an interval $[a, b]$, it is enough to specify $Q(a)$, $T(a)$ and (θ_k^i) to completely characterize the fluid solution throughout the interval. In fact, for $t \in [a, b]$,

$$(8.8) \quad T_k^i(t) = T_k^i(a) + \theta_k^i(t - a),$$

$$(8.9) \quad Q_k^i(t) = Q_k^i(a) + \mu_{k-1}^i \theta_{k-1}^i(t - a) - \mu_k^i \theta_k^i(t - a).$$

Note that θ_k^i is the fraction of its effort the server allocates to class (i, k) and $\mu_k^i \theta_k^i$ is the rate at which fluid leaves the buffer and hence the rate at which it enters the next buffer. Throughout our construction we show that not only must

$$\sum_{i \in I, k \in A_i} \dot{T}_k^i(t) = \sum_{i \in I, k \in A_i} \theta_k^i \leq 1$$

and

$$\sum_{i \in I, k \in B_i} \dot{T}_k^i(t) = \sum_{i \in I, k \in B_i} \theta_k^i \leq 1$$

at each time t , but also the two stations cannot serve classes in $V(S)$ simultaneously. Thus, we show that whenever $\theta(V_A(S)) > 0$, $\theta(V_B(S)) = 0$ and whenever $\theta(V_B(S)) > 0$, $\theta(V_A(S)) = 0$ from which it immediately follows that

$$\sum_{(i,k) \in V(S)} \dot{T}_k^i(t) \leq 1$$

for each regular point t . Again, for a set X of classes, we let $\theta(X) = \sum_{(i,k) \in X} \theta_k^i$.

We begin with u_r^i units in the first buffer of $C(X_r^i)$ for each A -section X_r^i so that all the WIP is initially at Station B . So, in the example of Figure 3, we begin with u_1^1 units in buffer $(1, 1)$ (the first class of the block $C(X_1^1)$), u_3^1 units in buffer $(1, 9)$ (the first class of the block $C(X_3^1)$) and u_2^2 units in buffer $(2, 3)$ (the first class of the block $C(X_2^2)$).

Step 1. One at a time, for each non-input A -section $X_r^i = [i, s, t)$ drain the contents of $\ell[i, s]$, the first buffer of $C(X_r^i)$, into the first buffer of $C_A(X_r^i)$ and the first buffer $\ell[i, t]$ of $C(Y_{r+1}^i)$ (or out of the system if X_r^i is the last section for type i) both at Station A . This process continues until the first buffer of $C(X_r^i)$ is empty. Figure 8 illustrates this step for the A -sections X_3^1 and X_2^2 in the example of Figure 3.

At the end of this step all the buffers at Station B in the blocks of non-input sections are empty. To see that this is consistent with a non-idling dispatch policy, observe that the system defining the flows is:

$$\begin{aligned} \theta(C_A(X_r^i)) &= 1 && \text{Since work is accumulating at Station } A, \\ \theta(C_B(X_r^i)) &= 1 && \text{Since there is work accumulated at Station } B, \\ \mu_k^i \theta_k^i - \mu_{k-1}^i \theta_{k-1}^i &= 0 && \text{For each class } (i, k) \text{ in } C(X_r^i) \text{ except the first two. The first} \\ &&& \text{class's buffer is draining, so the rate of flow out is faster than} \\ &&& \text{the rate of flow in. The second class's buffer is accumulating} \\ &&& \text{fluid, so the rate of flow in is greater than the rate of flow out.} \end{aligned}$$

This system has the unique solution:

$$\theta_k^i = \frac{m_k^i}{m(C_A(X_r^i))}$$

for each class in $C(X_r^i)$ except the first one. This class, $\ell[s, i]$, has

$$\theta_{\ell[s, i]}^i = \frac{m_{\ell[s, i]}^i}{m(C_A(X_r^i))} + \frac{m(C_A(X_r^i)) - m(C_B(X_r^i))}{m(C_A(X_r^i))}$$

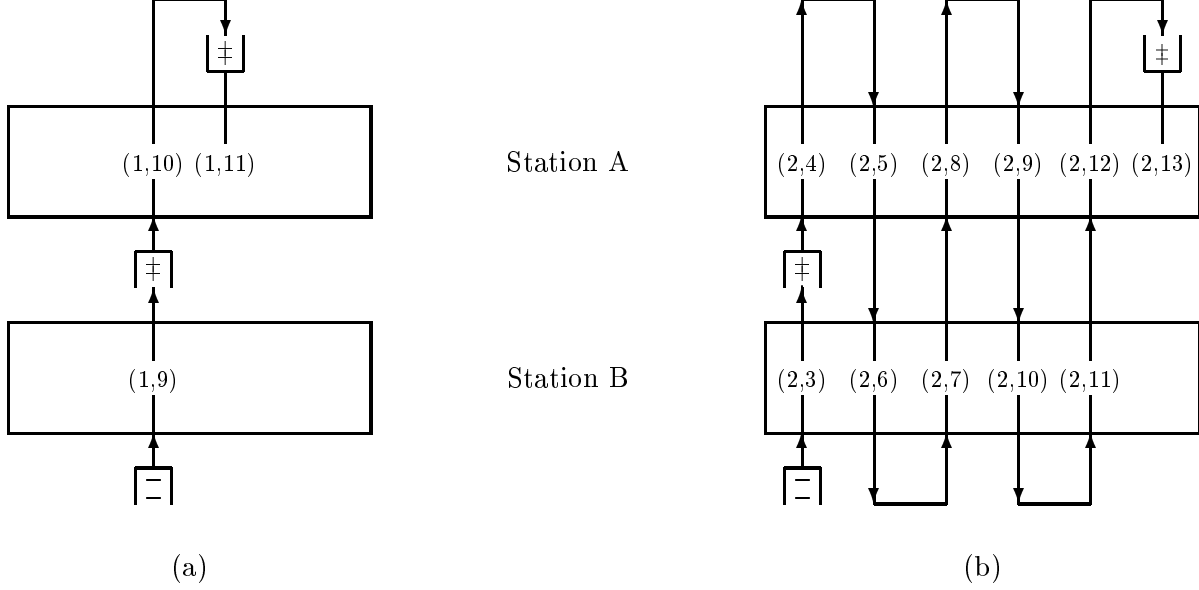


Figure 8: This figure depicts the flows when Step 1 is applied to X_3^1 , shown in part (a), and to X_2^2 , shown in part (b), in the example of Figure 3. Only the relevant classes are shown. Arrows depict the flows and an open box in front of a class denotes a buffer that may contain a positive quantity of fluid during the step. Plus signs inside a buffer indicate that it is accumulating fluid while minus signs indicate that it is draining. All other indicated buffers remain empty throughout the step.

which, by (8.6), is at least as great as

$$\frac{m_{\ell[s,i]}^i}{m(C_A(X_r^i))}.$$

Thus,

$$\mu_{\ell[i,s]}^i \theta_{\ell[s,i]}^i \geq \frac{1}{m(C_A(X_r^i))} = \mu_{\ell[s,i]+1}^i \theta_{\ell[s,i]+1}^i$$

and so fluid is arriving at the second buffer of $C(X_r^i)$ at least as fast as the server at Station A processes it. Thus, both stations are busy until the first buffer in $C(X_r^i)$ is emptied. The contents of the buffer have moved to the first buffer of the next excursion and the first buffer of $C(Y_{r+1}^i)$ — both at Station A .

Both servers are fully busy during this entire step and so any fluids arriving to the system during this period simply accumulate at their first buffers.

Step 2. Next, for each non-input A -section $X_r^i = [i, s, t)$ one at a time in any order, drain the first buffer of $C_A(X_r^i)$ into the first buffer of $C(Y_{r+1}^i)$ (or out of the system if this is a last section).

When the first buffer of $C_A(X_r^i)$ is emptied, we have accumulated in the first buffer of $C(Y_{r+1}^i)$ (if there is one) all the flow originally in the first buffer of $C(X_r^i)$. Figure 9 illustrates this step for the A -sections X_3^1 and X_2^2 in the example of Figure 3.

The system defining the flows for each of these A -sections is:

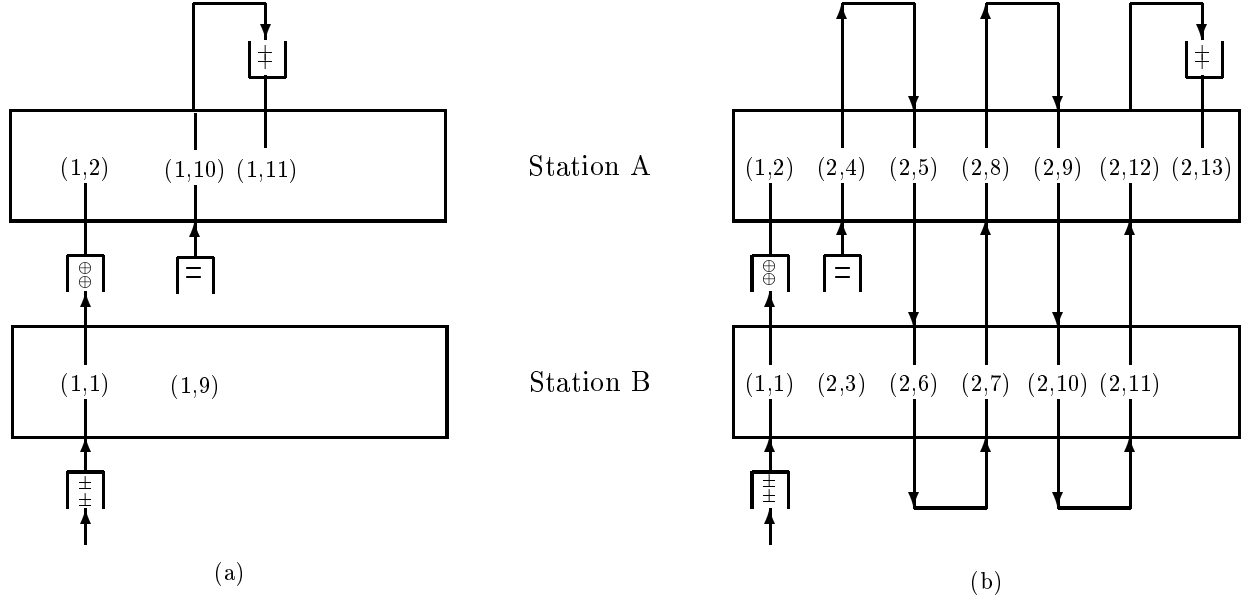


Figure 9: This figure depicts the flows when Step 2 is applied to X_3^1 , shown in part (a), and to X_2^2 , shown in part (b). Only the relevant classes are shown. Arrows depict the flows and an open box in front of a class denotes a buffer that may contain a positive quantity of fluid during the step. Plus signs inside a buffer indicate that it is accumulating fluid while minus signs indicate that it is draining. The signs \pm inside a buffer indicate that it may either accumulate fluid or drain during the step, depending on the balance of flows into and out of the buffer. The signs \oplus inside a buffer indicate that it may either accumulate fluid or be empty. All other indicated buffers remain empty throughout the step.

$$\begin{aligned}
 \theta(C_A(X_r^i)) &= 1 && \text{Since work is accumulated at Station A,} \\
 \mu_k^i \theta_k^i - \mu_{k-1}^i \theta_{k-1}^i &= 0 && \text{For each class } (i, k) \text{ in } C(X_r^i) \text{ except the first two. We emptied} \\
 &&& \text{the buffer of the first class in Step 1 and we are draining the} \\
 &&& \text{buffer of the second class.}
 \end{aligned}$$

This system has the unique solution:

$$\theta_k^i = \frac{m_k^i}{m(C_A(X_r^i))}$$

for each class in the block except the first, which has $\theta_{\ell[s,i]}^i = 0$.

These flows keep the server at Station A busy, but, by Lemma 8.1, may not keep the server at Station B busy. Whenever these flows do not keep the server at Station B busy, it can process fluids that have accumulated at classes $(1, i)$ for $i \in I_B$. However, since the server at Station A is busy, these fluids will accumulate at the first buffer of their second excursion.

At the end of Step 2, all the fluid initially in the first buffer of $C(X_r^i)$ for each non-input A-section X_r^i has been moved to the first buffer of $C(Y_{r+1}^i)$. For the example in Figure 3, fluid in buffers $(1, 9)$ and $(2, 3)$ has been moved to buffers $(1, 11)$ and $(2, 13)$. For each A-section X_r^i , moving

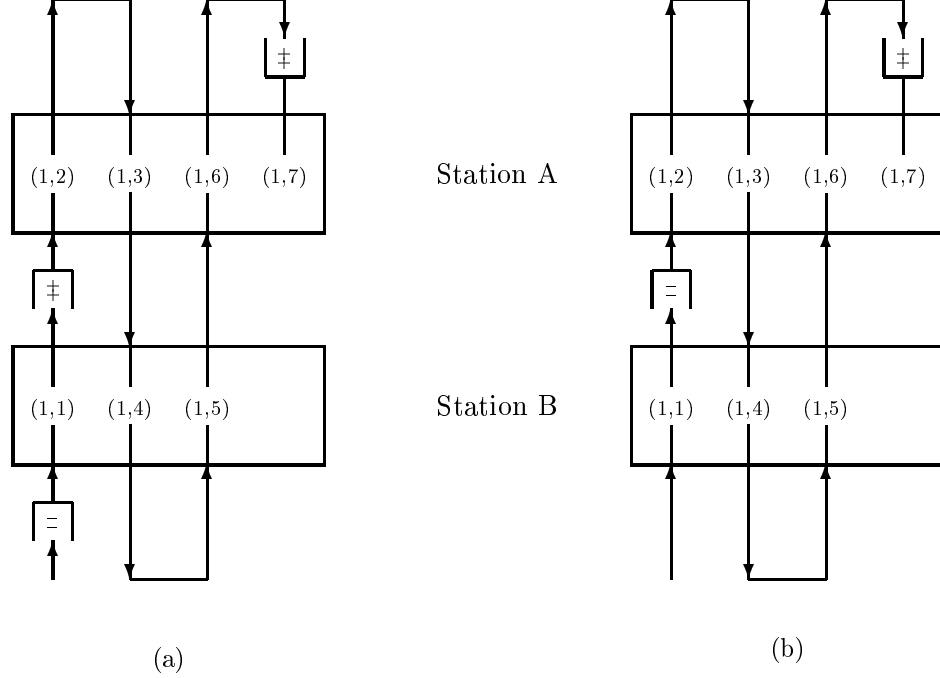


Figure 10: This figure depicts the flows when Steps 3 and 4 are applied to the example in Figure 3. Only the relevant classes are shown. Arrows depict the flows and an open box in front of a class denotes a buffer that may contain a positive quantity of fluid during the step. Plus signs inside a buffer indicate that it is accumulating fluid while minus signs indicate that it is draining. All other indicated buffers remain empty throughout the step.

u_r^i units through the classes of $C(X_r^i)$ requires time $u_r^i m(C_A(X_r^i))$. Thus, we spend

$$(8.10) \quad \tau_1 = \sum_{i \in I_A} \sum_{r=1}^{b_i} u_{2r}^i m(C_A(X_{2r}^i)) + \sum_{i \in I_B} \sum_{r=2}^{b_i} u_{2r-1}^i m(C_A(X_{2r-1}^i))$$

on Steps 1 and 2.

Note that these two steps are consistent with a static buffer priority dispatch policy, which uses the last-buffer-first-served dispatch policy within the block of each non-input A -section.

Step 3. One at a time, for each type $i \in I_B$ in order, drain the the fluid that has accumulated in class $(i, 1)$ of $C(X_1^i)$ into $(i, 2)$, the first buffer of $C_A(X_1^i)$, and the first buffer of $C(Y_2^i)$ (or out of the system if X_1^i is the last section for type i), while keeping the buffers of $C(X_1^j)$ empty for all earlier types $j \in I_B$, with $j < i$. This process continues until buffer $(i, 1)$ is empty. Figure 10 (a) illustrates this step for type 1 in the example of Figure 3.

At the end of this step all the buffers at Station B are empty. To see that this is consistent with a non-idling dispatch policy, observe that the system defining the flows for each type $i \in I_B$ is:

$$\begin{aligned}
\sum_{j \in I_B, j \leq i} \theta(C_A(X_1^j)) &= 1 && \text{Since work is accumulating at Station } A, \\
\sum_{j \in I_B, j \leq i} \theta(C_B(X_1^j)) &= 1 && \text{Since there is work accumulated at Station } B, \\
\mu_1^j \theta_1^j &= \lambda_j && \text{For each type } j \in I_B \text{ with } j < i, \\
\mu_k^j \theta_k^j - \mu_{k-1}^j \theta_{k-1}^j &= 0 && \text{For each class } (j, k) \text{ in } C(X_1^j) \text{ where } j \in I_B, j < i \text{ and } k > 1, \\
\mu_k^i \theta_k^i - \mu_{k-1}^i \theta_{k-1}^i &= 0 && \text{For each class } (i, k) \text{ in } C(X_1^i) \text{ except the first two. The first} \\
&&& \text{class's buffer is draining, so the rate of flow out is faster than} \\
&&& \text{the rate of flow in. The second class's buffer is accumulating} \\
&&& \text{fluid, so the rate of flow in is greater than the rate of flow} \\
&&& \text{out.}
\end{aligned}$$

This system has the unique solution:

$$\begin{aligned}
\theta_k^j &= \lambda_k^j m_k^j \text{ for each class } (j, k) \in C(X_1^j) \text{ where } j \in I_B \text{ and } j < i \\
\theta_k^i &= \frac{m_k^i \left(1 - \sum_{j \in I_B, j < i} \lambda m(C_A(X_1^j))\right)}{m(C_A(X_1^i))} \text{ for each class } (i, k) \in C(X_1^i) \text{ except class } (i, 1) \\
\theta_1^i &= \frac{m_1^i \left(1 - \sum_{j \in I_B, j < i} \lambda m(C_A(X_1^j))\right)}{m(C_A(X_1^i))} + \sum_{j \in I_B, j < i} \left(\lambda m(C_A(X_1^j)) - \lambda m(C_B(X_1^j))\right),
\end{aligned}$$

which, by (8.6), is at least as great as

$$\frac{m_1^i \left(1 - \sum_{j \in I_B, j < i} \lambda m(C_A(X_1^j))\right)}{m(C_A(X_1^i))}.$$

Thus,

$$\mu_1^i \theta_1^i \geq \frac{1 - \sum_{j \in I_B, j < i} \lambda m(C_A(X_1^j))}{m(C_A(X_1^i))} = \mu_2^i \theta_2^i$$

and so fluid is arriving at class $(i, 2)$ at least as fast as the server at Station A processes it. Thus, both stations are busy until buffer $(i, 1)$ is emptied. The contents of the buffer have moved to buffer $(i, 2)$ and the first buffer of $C(Y_2^i)$ — both at Station A .

Both servers are fully busy during this entire step and so any fluids arriving to Station A during this period simply accumulate at their first buffers.

Step 4. One at a time, for each type $i \in I_B$ in order, empty buffer $(i, 2)$ while keeping empty both buffer $(i, 1)$ and the buffers of $C(X_1^j)$ for all earlier types $j \in I_B$, with $j < i$. Figure 10 (b) illustrates this step for type 1 in the example of Figure 3.

The system defining the flows is:

$$\begin{aligned}
\sum_{j \in I_B, j \leq i} \theta(C_A(X_1^j)) &= 1 && \text{Since work is accumulating at Station A,} \\
\mu_1^j \theta_1^j &= \lambda_j && \text{For each type } j \in I_B \text{ with } j \leq i, \\
\mu_k^j \theta_k^j - \mu_{k-1}^j \theta_{k-1}^j &= 0 && \text{For each class } (j, k) \text{ in } C(X_1^j) \text{ where } j \in I_B, j < i \text{ and } \\
&&& k > 1, \\
\mu_k^i \theta_k^i - \mu_{k-1}^i \theta_{k-1}^i &= 0 && \text{For each class } (i, k) \text{ in } C(X_1^i) \text{ except the first two. The first} \\
&&& \text{class's buffer is empty and, to keep it that way, its output} \\
&&& \text{rate must match the rate of exogenous arrivals. The second} \\
&&& \text{class's buffer is draining, so the rate of flow out is greater} \\
&&& \text{than the rate of flow in.}
\end{aligned}$$

This system has the unique solution:

$$\begin{aligned}
\theta_k^j &= \lambda_k^j m_k^j \text{ for each class } (j, k) \in C(X_1^j) \text{ where } j \in I_B \text{ and } j < i \\
\theta_k^i &= \frac{m_k^i \left(1 - \sum_{j \in I_B, j < i} \lambda m(C_A(X_1^j))\right)}{m(C_A(X_1^i))} \text{ for each class } (i, k) \in C(X_1^i) \text{ except class } (i, 1) \\
\theta_1^i &= \lambda_i m_1^i.
\end{aligned}$$

Note that since the arrival rates and service times satisfy the nominal workload conditions, θ is between 0 and 1. Further, as in Step 2, the server at Station A is fully busy during this entire step, but the server at Station B may not be. To see this observe that

$$\theta(E_B) = \sum_{j \in I_B, j < i} \lambda m(C_B(X_1^j)) + \frac{(m(C_B(X_1^i)) - m_1^i) \left(1 - \sum_{j \in I_B, j < i} \lambda m(C_A(X_1^j))\right)}{m(C_A(X_1^i))} + \lambda m_1^i.$$

By (8.6),

$$\frac{m(C_B(X_1^i))}{m(C_A(X_1^i))} \leq 1$$

and since the arrival rates and service times satisfy the nominal workload conditions,

$$\frac{1 - \sum_{j \in I_B, j < i} \lambda m(C_B(X_1^j))}{m(C_A(X_1^i))} > \lambda_i.$$

Thus,

$$\theta(E_B) < 1.$$

Whenever the server at Station B has remaining capacity, he again drains the contents of subsequent buffers $(j, 1)$ with $j \in I_B$ and $j > i$. Since the server at Station A is busy, however, these fluids accumulate in their second buffers.

Continue in this way until all buffers in $C(X_1^i)$, $i \in I_B$, are empty. Let τ_2 be the additional time needed to complete Steps 3 and 4, then

$$\tau_2 = (\tau_1 + \tau_2) \sum_{j \in I_B} \lambda m(C_A(X_1^j)) + \sum_{j \in I_B} u_1^j m(C_A(X_1^j))$$

or

$$\tau_2 = \frac{\tau_1 \sum_{j \in I_B} \lambda m(C_A(X_1^j)) + \sum_{j \in I_B} u_1^j m(C_A(X_1^j))}{1 - \sum_{j \in I_B} \lambda m(C_A(X_1^j))}.$$

At the end of this step, the buffers of each block $C(X_r^i)$ are empty and only the first buffer of each block $C(Y_r^i)$ has any accumulated fluid. Thus, all the WIP is at Station A .

Let

$$(8.11) \quad \tau = \tau_1 + \tau_2 = \frac{\sum_{i \in I_A} \sum_{r=1}^{b_i} u_{2r}^i m(C_A(X_{2r}^i)) + \sum_{i \in I_B} \sum_{r=1}^{b_i} u_{2r-1}^i m(C_A(X_{2r-1}^i))}{1 - \sum_{j \in I_B} \lambda m(C_A(X_1^j))}.$$

For each $i \in I_A$, the first buffer in the block $C(Y_1^i)$, $(i, 1)$, has accumulated $\lambda_i \tau$ fluid, i.e., $Q_1^i(\tau) = \lambda_i \tau$. For each $i \in I_B$, the first buffer in the block $C(Y_2^i)$ has accumulated $\lambda_i \tau + u_1^i$ fluid. For each remaining B -section, the first buffer of $C(Y_r^i)$ has accumulated the u_{r-1}^i fluids that were initially in the first buffer of $C(X_{r-1}^i)$.

It is easy to check that the fluid solution $(Q(\cdot), T(\cdot))$ constructed in (8.8)–(8.9) satisfies (1.5)–(1.10) for $t \in [0, \tau]$. Furthermore, none of the classes in blocks of B -sections has ever been worked on. That is, $\theta(C(Y_r)) = 0$ for each B -section Y_r . Therefore, $(Q(\cdot), T(\cdot))$ satisfies (7.1) for t in $[0, \tau]$ and, by Lemma 7.1, there is more work at the virtual station $V(S)$ at time τ than at time 0.

We next repeat Steps 1–4 interchanging the roles of Station A and Station B . The fluid network completes the four steps at time t_1 in a state very similar to its initial state: there are \tilde{u}_r^i units in the first buffer of $C(X_r^i)$ for each A -section X_r^i and all the other buffers are empty. Further, the resulting fluid solution $(Q(\cdot), T(\cdot))$ satisfies (1.5)–(1.10) and (7.1) for all times $t \in [0, t_1]$.

To complete the argument, we would like to conclude that repeating these eight steps over and over produces a fluid solution $(Q(\cdot), T(\cdot))$ that satisfies (1.5)–(1.10) and (7.1) for all times $t \geq 0$. We could then invoke Lemma 7.1 to argue that the workload for the classes of $V(S)$ grows linearly with time and the fluid solution diverges. To reach this conclusion, however, we must first show that $\sum_{n=1}^{\infty} t_n \rightarrow \infty$, where t_n is the time required to complete the n th iteration of these eight steps. Otherwise, $\sum_{n=1}^{\infty} t_n \rightarrow t^*$ for some finite t^* and repeating these eight steps only produces a fluid solution for times $t \in [0, t^*]$.

To this end, let u be an arbitrary non-negative vector of initial fluid levels for the first buffers in the blocks of the A -sections X_r^i . From Equation (8.11) it is clear that the time required to complete the eight steps is a positive linear combination of the initial buffer levels u , i.e.,

$$t_1(u) = \sum c_r^i u_r^i,$$

where each coefficient $c_r^i > 0$. Similarly, the initial total workload at the virtual station $V(S)$ (as defined in (7.2) with $C = V(S)$) is a positive linear combination of the initial buffer levels u , i.e.,

$$W(u) = \sum a_r^i u_r^i,$$

where each coefficient $a_r^i > 0$.

Thus, the shortest time t_1^* to complete the eight steps given an initial workload $W(u) \geq 1$ can be found by solving the simple linear program:

$$\begin{aligned} t_1^* &= \min \sum c_r^i u_r^i \\ \text{s.t. } &\sum a_r^i u_r^i \geq 1 \\ &u_r^i \geq 0 \end{aligned}$$

Since all the data are positive, $t_1^* = \min\{c_r^i/a_r^i\} > 0$. Thus, if we begin with initial fluid levels u with $W(u) \geq 1$, we know from Lemma 7.1 that the workload at the virtual station $V(S)$ will not decrease and so each iteration will require at least time t_1^* . Hence, $\sum_{n=1}^{\infty} t_n \rightarrow \infty$ and repeating the eight steps over and over produces a fluid solution with buffer levels that cycle to infinity. \square

9 Concluding Remarks

This paper not only provides an explicit characterization of the global stability region of two-station multiclass fluid networks, it also offers intuitive explanations of how the constraints defining the global stability region arise. Namely, the two phenomena of virtual stations and push starts explain all the global stability conditions of two-station fluid networks.

Under certain assumptions on the interarrival and service time distributions as specified in Dai (1995), a queueing network is stable or positive Harris recurrent if the corresponding fluid network is stable. Thus, the workload conditions are sufficient to ensure the global stability of two-station multiclass queueing networks with deterministic routing. In Dai and Vande Vate (1996), we show that under some weaker distributional assumptions, the virtual station conditions are also necessary to ensure the global stability of the queueing network. The push start conditions, on the other hand, are not generally necessary for global stability of the queueing network.

Weak stability, also called *rate stability*, is a less restrictive form of stability in queueing networks (see for example El-Taha and Stidham 1994). Chen(1995) and Dai and Vande Vate (1996) show that weak stability of the fluid model implies weak stability of the queueing network. A fluid network is said to be *weakly stable* under non-idling dispatch policies, or simply *globally weakly stable*, if any fluid solution $(Q(\cdot), T(\cdot))$ with initial WIP zero will never have any WIP. The following theorem is a weak stability version of Theorem 2.1. Its proof is parallel to the development in this paper.

Theorem 9.1. *A two-station fluid network is globally weakly stable if and only if*

$$\rho_A \leq 1, \quad \rho_B \leq 1,$$

and for each vector $\mathbf{e} = (e_i)_{i \in I}$ of excursions and each strictly separating set S , we have

$$(9.1) \quad \frac{\lambda m(V_A(S) \setminus F_A^<(\mathbf{e}))}{1 - \lambda m(F_A^<(\mathbf{e}))} + \frac{\lambda m(V_B(S) \setminus F_B^<(\mathbf{e}))}{1 - \lambda m(F_B^<(\mathbf{e}))} \leq 1.$$

This paper characterizes the global stability region for fluid models of two-station queueing networks with deterministic routing. Hasenbein (1998) extended these results to the fluid models of certain two-station queueing networks with probabilistic routing.

Theorem 2.1 demonstrates that the global stability region of a two-station fluid network is monotone, i.e., reducing service times or arrival rates maintains global stability.

Corollary 9.1. *The global stability region of a two-station fluid network is monotone.*

This is not the case for fluid networks with more than two stations. Humes (1994) and Dai, Hasenbein and Vande Vate (1998) showed that the global stability region is *not* monotone. Dumas (1997) showed that the stability region of a priority network is not monotone. These examples indicate a daunting challenge inherent in managing complex re-entrant manufacturing systems like wafer fabrication plants: increasing the processing rate for a class can reduce the capacity of the system!

The fact that the global stability region of a fluid network with more than two stations can be non-monotone suggests that determining the global stability region for these networks will be considerably more difficult. For example, the problem of determining the coefficients of a piecewise linear Lyapunov function for a fluid network with more than two stations does not generally reduce to a linear program. Nevertheless, the example studied in Dai, Hasenbein and Vande Vate (1998) is one of an important family of n -station fluid networks for which this problem is linear and, in fact,

does reduce to a parametric network flow problem. Thus, the piecewise linear Lyapunov function of this paper does lead to sufficient conditions for global stability for this family of n -station networks.

The four steps of Section 8 prove that the static buffer priority policies are the “worst” dispatch policies for two-station fluid networks.

Corollary 9.2. *A two-station fluid network is globally stable if and only if it is stable under every static buffer priority policy.*

This is not the case for n -station fluid networks. Dai, Hasenbein and Vande Vate (1998) demonstrated arrival rates and service times for which their three-station fluid network is not globally stable even though it is stable under all static buffer priority disciplines.

Acknowledgements

This research was initiated when J. G. Dai was visiting the Institute of Mathematics and Its Applications at the University of Minnesota in the winter quarter of 1994. Partial financial support from the Institute is acknowledged. This research is supported in part by National Science Foundation grants DDM-9215233, DMI-94-57336, US-Israel Binational Science Foundation grant 94-00196, Airforce Office of Scientific Research grant F49620-95-1-0121 and a grant from Harris Semiconductor.

A Appendix

Proof of Lemma 7.2. Assume that the fluid model (7.5)–(7.10) is unstable. Then there is a fluid solution $(\tilde{Q}_k^i(\cdot), \tilde{T}_k^i(\cdot))_{(i,k) \in R(\mathbf{e})}$ satisfying (7.5)–(7.10) that is unstable. That is, there is a sequence $\{t_n\}$ with $t_n \rightarrow \infty$ such that $\tilde{Q}(t_n) \neq 0$ for each n . For $(i, k) \in R_A(\mathbf{e})$, let $Q_k^i(t) = \tilde{Q}_k^i(t)$ and

$$T_k^i(t) = (1 - \lambda m(F_A^<(\mathbf{e}))) \tilde{T}_k^i(t).$$

For $(i, k) \in R_B(\mathbf{e})$, let $Q_k^i(t) = \tilde{Q}_k^i(t)$ and

$$T_k^i(t) = (1 - \lambda m(F_B^<(\mathbf{e}))) \tilde{T}_k^i(t).$$

For $(i, k) \in F^<(\mathbf{e})$, let $Q_k^i(t) = 0$ and $T_k^i(t) = \lambda_i m_k^i t$. We show that

$$(Q(\cdot), T(\cdot)) = (Q_k^i(\cdot), T_k^i(\cdot))_{i \in I, k=1, \dots, c_i}$$

is a fluid solution to (1.5)–(1.10).

First, it is easy to check that (1.5) and (1.6) hold for $(Q(\cdot), T(\cdot))$. Second, notice that

$$\begin{aligned} t - \sum_{i \in I, k \in A_i} T_k^i(t) &= t - \sum_{(i,k) \in F_A^<(\mathbf{e})} \lambda_i m_k^i t - \sum_{(i,k) \in R_A(\mathbf{e})} T_k^i(t) \\ &= (1 - \lambda m(F_A^<(\mathbf{e}))) t - (1 - \lambda m(F_A^<(\mathbf{e}))) \sum_{(i,k) \in R_A(\mathbf{e})} \tilde{T}_k^i(t) \\ &= (1 - \lambda m(F_A^<(\mathbf{e}))) \left(t - \sum_{(i,k) \in R_A(\mathbf{e})} \tilde{T}_k^i(t) \right), \end{aligned}$$

which is nondecreasing and hence (1.7) holds. Similarly, we see that

$$t - \sum_{i \in I, k \in B_i} T_k^i(t)$$

is non-decreasing.

Assume that t is a regular point of $(Q(\cdot), T(\cdot))$. When

$$\sum_{i \in I, k \in A_i} Q_k^i(t) > 0,$$

we have

$$\sum_{(i,k) \in R_A(\mathbf{e})} \tilde{Q}_k^i(t) > 0.$$

Hence

$$\sum_{(i,k) \in R_A(\mathbf{e})} \dot{T}_k^i(t) = 1,$$

or

$$\sum_{(i,k) \in R_A(\mathbf{e})} \dot{T}_k^i(t) = 1 - \sum_{(i,k) \in F_A^<(\mathbf{e})} \lambda_i m_k^i.$$

The last equation is equivalent to

$$\sum_{i \in I, k \in A_i} \dot{T}_k^i(t) = 1.$$

Hence we have proved (1.9). Similarly, we can show that (1.8) and (1.10) hold. Therefore, $(Q(\cdot), T(\cdot))$ is a fluid solution to (1.5)–(1.10). It is obvious that $(Q(\cdot), T(\cdot))$ is unstable. \square

Proof of Lemma 8.1. Consider an A -section $X = [i, s, t)$ of S^i with $s > e_i$. Let $S \oplus X = (S \setminus X) \cup (X \setminus S)$ be the symmetric difference of S and X . Because $s - 1 \notin S$ and $t \notin S$, $S \oplus X$ is a strictly separating set. Thus, by (8.2),

$$\begin{aligned} & \left(\frac{\lambda m(V_A(S) \setminus F_A^<(\mathbf{e}))}{1 - \lambda m(F_A^<(\mathbf{e}))} + \frac{\lambda m(V_B(S) \setminus F_B^<(\mathbf{e}))}{1 - \lambda m(F_B^<(\mathbf{e}))} \right) - \\ & \left(\frac{\lambda m(V_A(S \oplus X) \setminus F_A^<(\mathbf{e}))}{1 - \lambda m(F_A^<(\mathbf{e}))} + \frac{\lambda m(V_B(S \oplus X) \setminus F_B^<(\mathbf{e}))}{1 - \lambda m(F_B^<(\mathbf{e}))} \right) = \\ & \frac{\lambda m(C_A(X))}{1 - \lambda m(F_A^<(\mathbf{e}))} - \frac{\lambda m(C_B(X))}{1 - \lambda m(F_B^<(\mathbf{e}))} \geq 0. \end{aligned}$$

From which (8.4) follows immediately. The proof for B -sections $Y = [i, s, t)$ of S^i , where $s > e_i$ is similar.

The argument for sections $[i, s, t)$, where $s \leq e_i$, is more involved. Rather than change the separating set S , we change the push start set $F^<(\mathbf{e})$. Consider an A -section $X = [i, s, t)$ of S^i , where $s \leq e_i$, and let $\mathbf{e}' = (e'_j)_{j \in I}$, where $e'_i = t$ and for each $j \neq i$, $e'_j = e_j$.

By (8.2),

$$\frac{\lambda m(V_A(S) \setminus F_A^<(\mathbf{e}))}{1 - \lambda m(F_A^<(\mathbf{e}))} + \frac{\lambda m(V_B(S) \setminus F_B^<(\mathbf{e}))}{1 - \lambda m(F_B^<(\mathbf{e}))} \geq \frac{\lambda m(V_A(S) \setminus F_A^<(\mathbf{e}'))}{1 - \lambda m(F_A^<(\mathbf{e}'))} + \frac{\lambda m(V_B(S) \setminus F_B^<(\mathbf{e}'))}{1 - \lambda m(F_B^<(\mathbf{e}'))}$$

and so,

$$(A.1) \quad \frac{\lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e}))}{1 - \lambda m(F_A^{\leq}(\mathbf{e}))} - \frac{\lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e}'))}{1 - \lambda m(F_A^{\leq}(\mathbf{e}'))} \geq \frac{\lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}'))}{1 - \lambda m(F_B^{\leq}(\mathbf{e}'))} - \frac{\lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}))}{1 - \lambda m(F_B^{\leq}(\mathbf{e}))}.$$

Now, since

$$\lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e}')) = \lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e})) - \lambda m(C_A(X) \setminus F_A^{\leq}(\mathbf{e}))$$

and

$$\begin{aligned} \lambda m(F_A^{\leq}(\mathbf{e}')) &= \lambda m(F_A^{\leq}(\mathbf{e})) + \lambda m(C_A(X) \setminus F_A^{\leq}(\mathbf{e})), \\ \frac{\lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e}))}{1 - \lambda m(F_A^{\leq}(\mathbf{e}))} - \frac{\lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e}'))}{1 - \lambda m(F_A^{\leq}(\mathbf{e}'))} &\geq \frac{\lambda m(C_A(X) \setminus F_A^{\leq}(\mathbf{e})) \left(1 - \lambda m(F_A^{\leq}(\mathbf{e})) - \lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e}))\right)}{(1 - \lambda m(F_A^{\leq}(\mathbf{e}')))(1 - \lambda m(F_A^{\leq}(\mathbf{e})))}. \end{aligned}$$

Similarly, since

$$\lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}')) = \lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}))$$

and

$$\begin{aligned} \lambda m(F_B^{\leq}(\mathbf{e}')) &= \lambda m(F_B^{\leq}(\mathbf{e})) + \lambda m(C_B(X) \setminus F_B^{\leq}(\mathbf{e})), \\ \frac{\lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}'))}{1 - \lambda m(F_B^{\leq}(\mathbf{e}'))} - \frac{\lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}))}{1 - \lambda m(F_B^{\leq}(\mathbf{e}))} &= \frac{\lambda m(C_B(X) \setminus F_B^{\leq}(\mathbf{e})) \lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}))}{(1 - \lambda m(F_B^{\leq}(\mathbf{e}')))(1 - \lambda m(F_B^{\leq}(\mathbf{e})))}. \end{aligned}$$

Thus, by (A.1)

$$(A.2) \quad \frac{\lambda m(C_A(X) \setminus F_A^{\leq}(\mathbf{e})) \left(1 - \lambda m(F_A^{\leq}(\mathbf{e})) - \lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e}))\right)}{(1 - \lambda m(F_A^{\leq}(\mathbf{e}')))(1 - \lambda m(F_A^{\leq}(\mathbf{e})))} \geq \frac{\lambda m(C_B(X) \setminus F_B^{\leq}(\mathbf{e})) \lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}))}{(1 - \lambda m(F_B^{\leq}(\mathbf{e}')))(1 - \lambda m(F_B^{\leq}(\mathbf{e})))}.$$

Since,

$$\begin{aligned} \frac{\lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e}))}{1 - \lambda m(F_A^{\leq}(\mathbf{e}))} + \frac{\lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}))}{1 - \lambda m(F_B^{\leq}(\mathbf{e}))} &\geq 1, \\ \frac{1 - \lambda m(F_A^{\leq}(\mathbf{e})) - \lambda m(V_A(S) \setminus F_A^{\leq}(\mathbf{e}))}{1 - \lambda m(F_A^{\leq}(\mathbf{e}))} &\leq \frac{\lambda m(V_B(S) \setminus F_B^{\leq}(\mathbf{e}))}{1 - \lambda m(F_B^{\leq}(\mathbf{e}))} \end{aligned}$$

and so from (A.2),

$$\frac{\lambda m(C_A(X) \setminus F_A^<(\mathbf{e}))}{1 - \lambda m(F_A^<(\mathbf{e}'))} \geq \frac{\lambda m(C_B(X) \setminus F_B^<(\mathbf{e}))}{1 - \lambda m(F_B^<(\mathbf{e}'))},$$

from which it follows that

$$\frac{\lambda m(C_A(X) \setminus F_B^<(\mathbf{e}))}{1 - \lambda m(F_A^<(\mathbf{e}))} \geq \frac{\lambda m(C_B(X) \setminus F_B^<(\mathbf{e}))}{1 - \lambda m(F_B^<(\mathbf{e}))}.$$

The argument for B -sections $Y = [i, s, t)$ with $s \leq e_i$ is similar. \square

References

- Ahuja, R. K., T. L. Magnanti, and J. B. Orlin (1993). *Network flows: theory, algorithms, and applications*. Englewood Cliffs, N.J.: Prentice Hall.
- Bertsimas, D., D. Gamarnik, and J. N. Tsitsiklis (1996). Stability conditions for multiclass fluid queueing networks. *IEEE Transactions on Automatic Control* **41**, 1618–1631. Correction: **42**, 128, 1997.
- Botvich, D. D. and A. A. Zamyatin (1992). Ergodicity of conservative communication networks. Rapport de recherche 1772, INRIA.
- Bramson, M. (1994a). Instability of FIFO queueing networks. *Annals of Applied Probability* **4**, 414–431.
- Bramson, M. (1994b). Instability of FIFO queueing networks with quick service times. *Annals of Applied Probability* **4**, 693–718.
- Bramson, M. (1997). Convergence to equilibria for fluid models of head-of-the-line proportional processor sharing queueing networks. *Queueing Systems: Theory and Applications* **23**, 1–26.
- Bramson, M. (1998a). Stability of two families of queueing networks and a discussion of fluid limits. *Queueing Systems: Theory and Applications* **28**, 7–31.
- Bramson, M. (1998b). A stable queueing network with unstable fluid network. Preprint.
- Chen, H. (1995). Fluid approximations and stability of multiclass queueing networks I: Work-conserving disciplines. *Annals of Applied Probability* **5**, 637–665.
- Chen, H. and H. Zhang (1997). Stability of multiclass queueing networks under FIFO service discipline. *Mathematics of Operations Research* **22**, 691–725.
- Chen, H. and H. Zhang (1998). Stability of multiclass queueing networks under priority service disciplines. *Operations Research*. To appear.
- Dai, J. G. (1995). Stability of open multiclass queueing networks via fluid models. In F. Kelly and R. J. Williams (Eds.), *Stochastic Networks*, Volume 71 of *The IMA volumes in mathematics and its applications*, New York, pp. 71–90. Springer-Verlag.
- Dai, J. G. (1996). A fluid-limit model criterion for instability of multiclass queueing networks. *Annals of Applied Probability* **6**, 751–757.
- Dai, J. G., J. Hasenbein, and J. H. Vande Vate (1998). Stability of a three-station fluid network. Under revision for *QUESTA*.

- Dai, J. G. and S. P. Meyn (1995). Stability and convergence of moments for multiclass queueing networks via fluid limit models. *IEEE Transactions on Automatic Control* **40**, 1889–1904.
- Dai, J. G. and J. VandeVate (1996). Virtual stations and the capacity of two-station queueing networks. Under revision for *Operations Research*.
- Dai, J. G. and G. Weiss (1996). Stability and instability of fluid models for re-entrant lines. *Mathematics of Operations Research* **21**, 115–134.
- Down, D. and S. Meyn (1994). Piecewise linear test functions for stability of queueing networks. *Proceedings of the 33rd Conference on Decision and Control*, 2069–2074.
- Dumas, V. (1996). Essential faces and stability conditions of multiclass networks with priorities. Rapport de recherche 3030, INRIA.
- Dumas, V. (1997). A multiclass network with non-linear, non-convex, non-monotonic stability conditions. *Queueing Systems: Theory and Applications* **25**, 1–43.
- El-Taha, M. and S. Stidham Jr. (1994). Sample-path stability conditions for multiserver input-output processes. *Journal of Applied Mathematics and Stochastic Analysis* **7**, 437–456.
- Foss, S. and A. Rybko (1995). Stability of multiclass Jackson-type networks. Preprint.
- Harrison, J. M. and V. Nguyen (1995). Some badly behaved closed queueing networks. In F. P. Kelly and R. J. Williams (Eds.), *Stochastic Networks*, Volume 71 of *The IMA volumes in mathematics and its applications*, New York, pp. 117–124. Springer-Verlag.
- Hasenbein, J. (1998). *Capacity and Scheduling of Multiclass Queueing Networks*. Ph. D. thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology.
- Hasenbein, J. J. (1997). Necessary conditions for global stability of multiclass queueing networks. *Operations Research Letters* **21**, 87–94.
- Humes Jr., C. (1994). A regulator stabilization technique: Kumar-Seidman revisited. *IEEE Transactions on Automatic Control* **39**, 191–196.
- Kumar, P. R. and S. Meyn (1996). Duality and linear programs for stability and performance analysis of queueing networks and scheduling policies. *IEEE Transactions on Automatic Control* **41**, 4–17.
- Kumar, P. R. and S. P. Meyn (1995). Stability of queueing networks and scheduling policies. *IEEE Transactions on Automatic Control* **40**, 251–260.
- Kumar, P. R. and T. I. Seidman (1990). Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *IEEE Transactions on Automatic Control* **AC-35**, 289–298.
- Lu, S. H. and P. R. Kumar (1991). Distributed scheduling based on due dates and buffer priorities. *IEEE Transactions on Automatic Control* **36**, 1406–1416.
- Meyn, S. P. (1995). Transience of multiclass queueing networks via fluid limit models. *Annals of Applied Probability* **5**, 946–957.
- Morrison, J. R. and P. R. Kumar (1998). On the guaranteed throughput and efficiency of closed re-entrant lines. *Queueing Systems: Theory and Applications* **28**, 33–54.
- Rybko, A. N. and A. L. Stolyar (1992). Ergodicity of stochastic processes describing the operation of open queueing networks. *Problems of Information Transmission* **28**, 199–220.
- Seidman, T. I. (1994). ‘First come, first served’ can be unstable! *IEEE Transactions on Automatic Control* **39**, 2166–2171.

- Stolyar, A. (1994). On the stability of multiclass queueing networks. In *Proceeding of the 2nd International Conference on Telecommunication Systems-Modeling and Analysis*, Nashville, Tennessee, pp. 23–35.
- Winograd, G. L. and P. R. Kumar (1996). The FCFS service discipline: stable network topologies, bounds on traffic burstiness and delay, and control by regulators. *Mathematical and Computer Modeling* 23(11/12), 115–129.