Oscillations with TCP-like Flow Control in Networks of Queues

Matthew Andrews and Aleksandrs Slivkins

Abstract—We consider a set of flows passing through a set of servers. The injection rate into each flow is governed by a flow control that increases the injection rate when all the servers on the flow’s path are empty and decreases the injection rate when some server is congested. We show that if each server’s congestion is governed by the arriving traffic at the server then the system can oscillate. This is in contrast to previous work on flow control where congestion was modeled as a function of the flow injection rates and the system was shown to converge to a steady state that maximizes an overall network utility.

Categories and subject descriptors: F.2.2 [Nonnumerical Algorithms and Problems]: Sequencing and scheduling.

General Terms: algorithms.

Keywords: routing, scheduling, stability.

I. INTRODUCTION

In this paper we study how TCP-like mechanisms for flow control operate in a network of queues with feedback. We consider a set of flows passing through a network of finite capacity servers. Let $x_i(t)$ be the injection rate of data into flow $i$. A series of papers, (e.g. [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]) has shown that many variants of TCP can be viewed as solving optimization problems on the variables $x_i$ subject to the constraint that none of the server capacities are overloaded.

To give a concrete example, consider the following model of TCP-like additive-increase-multiplicative-decrease (AIMD) flow control considered by Kelly et al. in [2]. Let $\mu_Q$ be the service rate of server $Q$. Let $S(Q)$ be the set of flows that pass through server $Q$. In [2], the injection rate into flow $i$ is controlled by

$$\dot{x}_i(t) = \kappa \left( w_i - x_i(t) \sum_{Q \in S(Q)} \gamma_Q(t) \right)$$

$$\gamma_Q(t) = p_Q \left( \sum_{i \in S(Q)} x_i(t) \right).$$

In the above equations $\kappa$ and $w_i$ are parameters, $\dot{x}_i(t)$ represents $x_i(t)$, $\sum_{Q \in S(Q)} x_i(t)$ represents the current congestion at server $Q$ and $p_Q(y)$ is a price charged by server $Q$ when the congestion is $y$. An example given in [2] is

$$p_Q(y) = (y - \mu_Q + \varepsilon)^+ / \varepsilon^2.$$

For sufficiently small values of $\varepsilon$ this is approximately equal to solving the system,

$$\max \sum_i w_i \log x_i$$

subject to:

$$\sum_{i \in S(Q)} x_i(t) \leq \mu_Q \quad \forall Q.$$
In this paper we examine whether such oscillations can occur when the injection rates are governed by a flow control such as \[\text{[1]}\text{-}\text{[2]},\] in the case when the congestion at a link is governed by \(\sum_{i \in S(Q)} a_{i,Q}(t)\) rather than \(\sum_{i \in S(Q)} x_i(t)\). In particular we consider a network of servers, most of which serve data according to First-in-First-Out (FIFO). The rate at which data arrives at internal nodes of the network is governed by the dynamics of the queues. When data leaves the network the amount of delay that the data has experienced within the network affects the new injection rates into the network, in a manner similar to \[\text{[1]}\text{-}\text{[2]}\]. Our main result is that there exists an oscillating example of Choe and Low \[\text{[7]}\] and Liu et al. \[\text{[8]}\] in which oscillations occurred due to the introduction of propagation delays into an injection-rate based model. We discuss this distinction in more detail in Section \[\text{[1]}\text{-}\text{[2]}.\]

A. The model: network dynamics

We consider a set of servers and a set of flows. Each flow consists of a path through the network. For ease of analysis we model data as a continuous fluid. We begin by describing how we model the network dynamics. We then describe how we model the flow control.

Let us consider a given server \(Q\). We will use the following notation. We let \(\mu\) be the service rate of \(Q\), and \(S\) be the set of flows that pass through \(Q\). For a time \(t\), we define \(h(t)\) to be the height of server \(Q\), which is the amount of fluid queued at server \(Q\) normalized by the service rate. We also let \(a_i(t)\) and \(d_i(t)\) be the arrival and departure rates, respectively. of fluid on flow \(i\) at server \(Q\).

Most of the servers in our oscillating examples will be FIFO servers. A FIFO server \(Q\) is governed by the following equations (see e.g. [16]):

\[
d_i(t + h(t)) = \frac{a_i(t)}{h(t) + 1}. \tag{3}
\]

If \(h(t) > 0\),

\[
h(t) = \left(\frac{1}{\mu} \sum_{i \in S} a_i(t)\right) - 1. \tag{4}
\]

If \(h(t) = 0\),

\[
h(t) = \max \left\{ \left(\frac{1}{\mu} \sum_{i \in S} a_i(t)\right) - 1, 0 \right\}. \tag{5}
\]

That is, any fluid that arrives at server \(Q\) at time \(t\) will depart at time \(t + h(t)\). The departure rate of this fluid will equal the arrival rate divided by a quantity that is proportional to the aggregate arrival rate.

When fluid leaves server \(Q\) it goes to the next server on its path, call it \(Q'\). The arrival rate at \(Q'\) is determined by the departure rate from server \(Q\) as well as the propagation delay between the two servers, call it \(\tau\). In particular we have

\[
a_{i,Q'}(t) = d_{i,Q}(t - \tau). \tag{6}
\]

In our construction all propagation delays will be zero.

In addition to the FIFO servers, a small number of servers in the network divide the flows into two classes and give priority to the first class over the second class; we call them two-priority FIFO servers. For these servers the flows from the first class behave as if they are being served by a FIFO queue but the only arrivals come from the first class flows. The second class flows are served according to FIFO among themselves but the service rate available to them is limited to the amount of capacity that is not used by the first class flows.

In Section \[\text{VI}\] we conjecture that these priorities are not necessary and that our oscillating example can be modified so all the queues are FIFO. The difficulty in proving this conjecture lies in arrival rates that are negligibly small but cumbersome to analyze.

B. The model: parameterized flow control

We now describe our flow control mechanism. Consider some session \(S\) at a given time \(t\). Say it is \textit{blocked} if at least one server \(Q\) on its path stores either a positive amount of session \(S\) fluid, or a positive amount of some other session’s fluid whose priority at \(Q\) is higher than that of \(S\). Say session \(S\) is \textit{happy} if any server \(Q\) on its path does not store any session \(S\) fluid and, moreover, the bandwidth available to \(S\) at \(Q\) given this server’s priorities is strictly larger than the injection rate of \(S\) into \(Q\). Say session \(S\) is \textit{stable} if it is neither blocked nor happy, i.e. if it is bottlenecked at the current injection rate. We assume that the injection rate of \(S\) should increase when \(S\) is happy, decrease when \(S\) is blocked, and stay constant when \(S\) is stable.

For the purposes of this paper we will assume that for a given session the flow control mechanism is given by two functions, \(f(\varepsilon, t)\) and \(g(t)\), which control flow decrease and flow increase, respectively. Specifically:

- \(f(\varepsilon, t)\) is the injection rate at time \(t\) assuming that at time 0 the session becomes blocked and stays blocked till time \(t\), and at time 0 the injection rate is \(\varepsilon\).
- \(g(t)\) is the injection rate at time \(t\) assuming that at time 0 the injection rate is 0, and between times 0 and \(t\) the session is happy.

We will allow an arbitrary flow increase function \(g(t)\) as long as it is monotone and increases to infinity. We consider two main examples of flow decrease: additive decrease:

\[
f_\delta(\varepsilon, t) = \varepsilon - \delta t, \tag{6}
\]

and multiplicative decrease

\[
f_\delta(\varepsilon, t) = \begin{cases} \varepsilon e^{-\delta t} & t \leq t_0 \\ \alpha \varepsilon - \delta (t - t_0) & \text{otherwise} \end{cases} \tag{7}
\]

where \(\alpha \in (0, 1)\) is a small constant and \(t_0\) is the time such that \(e^{-\delta t_0} = \alpha\). We use \(\delta\) instead of the “simple” multiplicative decrease \(f_\delta(\varepsilon, t) = \varepsilon e^{-\delta t}\) because for technical reasons we need the injection rate to drop to 0 in some finite time.
In both examples $\delta > 0$ is the damping parameter which controls the speed at which the flow rate decreases. We will assume that we are given a pair of functions $(f, g)$ which all servers must use, but we are allowed to tune $\delta$. To make the model slightly more general, we will assume that the flow increase function $g(t)$ is also parameterized by this $\delta$.

With the above types of flow control we can model both additive-increase-additive-decrease (AIAD) and additive-increase-multiplicative-decrease (AIMD) types of flow control. We recall that AIMD schemes have been widely studied in connection with providing effective flow control in the internet. (See e.g. [17], [18].)

To generalize examples (6) and (7), and to treat them in a unified model, we define the flow control mechanism as a triple $(f_3, g_3, S_{\text{DEL}})$, where $S_{\text{DEL}} \subseteq \mathbb{R}$ is the set of possible values for the damping parameter $\delta$, and the functions $(f_3, g_3)$ satisfy the following set of axioms:

(A1) increase to $\infty$: $g_3(t)$ increases to infinity for any $\delta$.

(A2) decrease to 0: for any fixed $\delta$ and $\varepsilon$, function $f_3(\varepsilon, t)$ becomes 0 at some positive time, and then stays 0. 

(A3) monotonicity: function $f_3(\varepsilon, t)$ is increasing in variable $\varepsilon$ and decreasing in parameter $\delta$ and in variable $t$.

(A4) damping: for any fixed rates $0 < \varepsilon' < \varepsilon$ and any fixed time $t$, there exists some $\delta$ such that $f_3(\varepsilon', t) > \varepsilon'$, and there exists some other $\delta$ such that $f_3(\varepsilon, t) < \varepsilon'$.

(A5) continuity: for any fixed $\delta$ the function $f_3(\varepsilon, t)$ is a continuous function from $\mathbb{R}^2$ to $\mathbb{R}$.

Axioms (A1-A5) are very natural. Axiom (A1) expresses why it is useful to have a damping parameter. Finally, Axiom (A5) captures the intuition that in the fluid-based model everything ought to be continuous. All five axioms easily satisfied by additive decrease (6) and multiplicative decrease (7).

Axiom (A1) is the only axiom on flow increase; note that it says nothing about the damping parameter. We will tune $\delta$ solely to obtain the desired flow decrease. In particular, for flow increase we will tolerate an arbitrary dependence on $\delta$.

Note that the function $f_3(\varepsilon, t)$ does not need to be continuous in $\delta$, and the set $S_{\text{DEL}}$ of possible values of $\delta$ does not even need to be a continuous real interval. For example, both for additive decrease (6) and for multiplicative decrease (7), this set can be discrete, e.g. $S_{\text{DEL}} = \mathbb{N} \cup \{1/n : n \in \mathbb{N}\}$.

C. The main result and extensions

A network of servers and sessions is a network of servers connected by server-to-server links, together with a set of sessions; here each session is specified by a source-sink pair, a flow path, and a flow control mechanism. At a given time the state of such network includes:

- injection rates into each session,
- height and composition of each queue

Here the composition of a given queue $Q$ includes, for each height $h$ and each session $S$ passing through $Q$, the density of session $S$ fluid at height $h$. (This density is defined to be the rate at which session $S$ fluid leaves queue $Q$ after time $h$.)

We say that a network oscillates if starting from some state it eventually reaches this state again.

We say that at a given time the injection rates are feasible if they do not overload any server: for each server $Q$, the sum of the current injection rates of all sessions that go through $Q$ is no bigger than the service rate of $Q$.

Our main result is that given any parameterized flow control mechanism, we can create a network that exhibits wide oscillations during which the injection rates stay feasible. We state this result as follows:

**Theorem I.1** Suppose we are given an arbitrary flow control mechanism that satisfies properties (A1-A3), and we are allowed to choose the damping parameter separately for each session. Suppose all servers must be either FIFO or two-priority FIFO, and we are allowed to choose the service rate separately for each server.

Then there exists a network of servers and sessions that oscillates so that the injection rates are feasible at all times. The damping parameters take only three distinct values.

The oscillations are very wide, in the sense that all injection, arrival and departure rates, and all queue heights, oscillate between 0 and their respective maximal values.

It is interesting to ask if we can fine-tune the above result so that all sessions are really running the same flow control mechanism, i.e. have the same damping parameter. We can do it for the case of additive decrease (6) and more generally for all flow decrease functions of the form

$$f_3(\varepsilon, t) = \varepsilon - \delta \cdot \varepsilon^\alpha \cdot t^\beta,$$

where $\alpha < 1$ and $\beta > 0$. (8)

Note that any such function satisfies axioms (A1-A5). In particular, by differentiating we can see that it is increasing in $\varepsilon$ whenever it is non-negative.

**Theorem I.2** Suppose in Theorem I.1 the flow decrease function satisfies (8). Then we can choose the same damping parameter for all sessions. Moreover, there exists $\delta^* \in S_{\text{DEL}}$ with the following property: for any $\delta \in S_{\text{DEL}}$ such that $\delta \leq \delta^*$ we can choose the damping parameter to be $\delta$.

In fact, our result applies to a somewhat more general family of flow decrease functions, see Section IV for further discussion. Note that we can take any given $\delta$ as the common damping parameter, as long as it is sufficiently small.

D. Remarks

We stress that in our example it is the dynamics of the queues that cause the oscillations. This provides a contrast with the work of Choe and Low [7] and Liu et al. [8] that showed that injection-rate based models of TCP Vegas and a variant called Stabilized Vegas can both be unstable when feedback delays are present. These papers model TCP as a set of differential equations with feedback and show that the feedback delays can cause the equations to oscillate. In particular, for each server $Q$ the injection rates of flows in $S(Q)$ oscillate between values that overload $Q$ and values that underload $Q$. In contrast, in our oscillating examples the injection rates $\pi_i(t)$ are such that $\sum_{i \in S(Q)} \pi_i(t) \leq \mu_Q$ always holds and so the injection rates do not a priori overload any
server. The oscillations arise because the queueing dynamics mean that temporary congestion is continually being created in the network.

Another feature of our model is that when congestion occurs the injection rate into a session decreases continuously. Some previous work has looked at a contrasting model of TCP in which the window size (and hence the injection rate) of a session is instantaneously decreased by half whenever congestion occurs. Baccelli et al. [19] showed that these “jumps” can lead to oscillations when a system of TCP sessions interacts with a RED or drop-tail queue. Our example shows that even if these jumps are eliminated using a continuous decrease then the flow control can still oscillate when interacting with a network of servers.

One paper that considers a similar problem to ours is Ajmone Marsan et al. [20]. They present an example consisting of a network of finite-buffer servers in which the scheduling discipline is either strict-priority or Generalized Processor Sharing. The example is constructed in such a way that if the source rates are non-adaptive then the queue sizes will build up in an oscillating manner. The paper [20] demonstrates via simulation that if we instead use adaptive additive-increase-multiplicative-decrease sources then we still get queue buildups that lead to packet losses and oscillating behavior. There are a number of differences between the model of [20] and our model however. The main difference is that [20] uses finite buffers and the sources only adapt to packet losses. In our model we consider unbounded buffers but the sources respond directly to congestion. Our results therefore demonstrate that as long as sources respond to congestion on a link, we can still get oscillating behavior and hence suboptimal utilization, even if no packet losses occur. Another difference between the two models is that [20] uses networks of strict-priority or GPS servers whereas our servers are mostly FIFO.

Finally we remark that one of the main reasons that we are able to create oscillations is that the queueing disciplines at the servers are oblivious to the state of the flow control. In contrast, there exist schemes in which the flow control protocols and the queueing protocols work in combination and are able to ensure convergence and prevent oscillations. See for example the Greedy primal-dual algorithm of Stolyar [21].

E. Preliminaries

Without loss of generality we may assume that we can choose the maximal injection rate for a given session. Indeed, for any target maximal rate $\varepsilon$ we can attach a new server in the very beginning of the session path, with a service rate $\varepsilon$. For simplicity, in the following sections each session will have two flow control parameters $(\varepsilon, \delta)$, where $\varepsilon$ is the maximal injection rate and $\delta$ is the damping parameter.

In practical settings it makes no sense to have sessions with non-simple flow paths (i.e. flow paths that go through the same server more than once). However, non-simple flow paths are often useful theoretically since they can make a construction more compact and/or clear. It turns out that one can use non-simple flow paths without loss of generality, since any network with non-simple flow paths can be converted to an equivalent network where all flow paths are simple, e.g. see [22]. The basic idea is that if $Z$ is the maximum number of times that a session visits a server then we create $Z$ copies of the network. Whenever a session is about to visit a server for second time it simply moves to a new copy of the network. More precisely, we organize the copies into a ring: we number the copies from 0 to $Z-1$, and the next visit from copy $i$ goes into copy $(i+1) \pmod{Z}$. In this way we can create an example in which each session visits a server at most once. However, in order to avoid this extra complexity, we will use non-simple flow paths without any further notice.

F. Organization of the paper

In Section II we overview our construction. In Section III we describe the main building block in our construction, which we call the basic gadget. Then in Section IV we proceed to the full construction and the proof of its performance. In Section V we discuss the extension where we choose the same damping parameters for all session. In Section VI we conjecture an extension where all servers are FIFO. Finally, in Section VII we conclude and state some open questions.

II. OVERVIEW OF OSCILLATING EXAMPLE

The complete description of our oscillating example is somewhat complex and so we begin with a brief overview. Our construction is based on rows of gadgets. Each row of gadgets has a single horizontal session $H$ that passes through all the servers in the row multiple times. Each gadget in a row consists of four servers $Q_1, Q_2, Q_3$, and $Q_4$. (See Figure 1). The server $Q_4$ in one gadget is identified with the server $Q_1$ in the next gadget. The main aim of the gadget is to transfer session $H$ fluid from server $Q_1$ to server $Q_4$. In order that this can occur session $H$ loops multiple times through all of servers $Q_1$-$Q_4$ and then loops multiple times through server $Q_4$ only. We also have a session $S$ that goes through server $Q_4$ only.

Initially, server $Q_1$ contains session $H$ fluid and server $Q_4$ is empty. This means that session $S$ is not blocked and so it is injecting at its maximum rate. As session $H$ fluid loops through server $Q_4$ we get a buildup in server $Q_4$. This causes the injection rate of session $S$ to decrease. Eventually we reach a state in which all of the session $H$ fluid is now in $Q_4$, the injection rate of session $S$ is zero, and there is no session $S$ fluid in server $Q_4$.

Since server $Q_4$ in the current gadget is identified with server $Q_1$ in the next gadget we can repeat this process. Note that during the process the injection rate of session $S$ has decreased from its maximum rate to zero. Once the session $H$ fluid has left server $Q_4$ and moved to the next gadget, the injection rate of session $S$ increases again to its maximum rate. Hence we have created oscillations in the injection rate of session $S$.

Unfortunately, we cannot continue the above process indefinitely. This is because in a finite network the session $H$ has finite length and so the session $H$ fluid will eventually reach
its destination. We therefore need a mechanism to replenish the session \( H \) fluid. We do this as follows.

At the beginning of the row of gadgets we have a server called a replenishing server. Initially all the servers in the row gadgets are empty and so session \( H \) fluid is injected at its maximum rate. We are able to create a buildup of fluid in the replenishing server. This causes session \( H \) fluid to build up in the replenishing server and it also causes the session \( H \) injection rate to go to zero. When enough session \( H \) fluid has been collected it begins the process of passing through the row of gadgets.

It remains to describe how we create a buildup of fluid in the replenishing queue. We do this in the following manner. As the session \( H \) fluid passes through the server \( Q_2 \) in a gadget, it creates a buildup at server \( Q_2 \). We have a new vertical session \( V \) that also passes through server \( Q_2 \). Session \( V \) is blocked by session \( H \) at server \( Q_2 \) and so we get a buildup of session \( V \) fluid. Our complete example consists of multiple rows of gadgets. Each vertical session \( V \) passes through multiple rows and multiple copies of \( Q_2 \). Moreover, the session \( V \) fluid gets blocked at each copy of \( Q_2 \). Each session \( V \) eventually passes through a replenishing queue at the beginning of a row of gadgets. Since the session \( V \) fluid has been blocked at multiple copies of \( Q_2 \) it enters the replenishing queue at a high rate. This causes the required buildup in the replenishing queue.

The vertical sessions \( V \) are depicted in Figure 2. The interaction of the vertical sessions with the replenishing server is depicted in Figure 3. We remark that each server processes data in FIFO order with two exceptions. First, each server \( Q_2 \) gives strict priority to session \( H \) fluid over session \( V \) fluid. Second, each replenishing server gives priority to session \( V \) fluid over session \( H \) fluid. As we discuss in Section VI we made a step towards relaxing the first requirement: we are able to create a (more complex) basic gadget where the servers \( Q_2 \) are strictly FIFO. We are unfortunately unable to construct an example where all servers are strictly FIFO. In Section VII we conjecture that this is possible and we briefly discuss why we believe this is so.

III. BASIC GADGET

In this section we’ll define a basic gadget that will be used as a main building block in our construction. This gadget consists of four servers called \( Q_1, Q_2, Q_3 \) and \( Q_4 \), and two sessions. We’ll describe a phase in which we transfer fluid from server \( Q_1 \) to server \( Q_4 \). During this phase the height of server \( Q_2 \) will increase and will then decrease again. The behavior of server \( Q_2 \) will allow us (in the full construction) to eventually accumulate new fluid that we use to replenish fluid that eventually reaches its destination.

A. Basic gadget: construction

The gadget consists of the four servers \( Q_1, \ldots, Q_4 \), and two sessions: one horizontal session that we call \( H \), and one simple session that we call \( S \). The horizontal session \( H \) starts and ends outside the gadget; in fact, in the full construction one such session comes through multiple gadgets. For the simple session \( S \), the source and the sink lie inside the gadget.

![Fig. 1. A basic gadget. At server \( Q_2 \), session \( H \) has priority over session \( V \). At server \( Q_4 \), sessions \( S \) and \( H \) are served in FIFO order.](image1)

![Fig. 2. The grid-like structure](image2)

Session \( H \) enters the gadget at server \( Q_1 \) and leaves it at server \( Q_4 \). The path of \( H \) first goes \( K \) times through the big loop, which consists of servers \( Q_1, Q_2, Q_3 \) and \( Q_4 \) in this order. The path then goes \( K' \) more times through the small loop, which consists of server \( Q_4 \) only. Here \( K \) and \( K' \) are parameters that we’ll tune later. The path of the simple session \( S \) goes through server \( Q_4 \) only. The two sessions compete in server \( Q_4 \) in FIFO order. See Fig. 1 for the composition of a single basic gadget.

Let us briefly preview how the basic gadgets fit into the full construction. There we shall have rows of servers that consist of many blocks of three servers each. Servers \( Q_1, Q_2 \) and \( Q_3 \) constitute one such block; server \( Q_4 \) will be the first server of the next block. The horizontal session \( H \) goes consecutively through all blocks in a given row (more precisely, \( H \) enters the next basic gadget right after it leaves the previous one; the details are Section IV). The blocks of servers are also organized in columns; there will be vertical sessions that go through consecutive servers \( Q_2 \) in the same column. These flows will not mix with the horizontal session \( H \) since server \( Q_2 \) will strictly prefer \( H \). This grid-like construction is summarized in Fig. 2.

Now let us go back to the level of a single basic gadget. We shall have several parameters (apart from \( K \) and \( K' \)) that we’ll tune later. For the simple session \( S \), let \( \varepsilon_s < 1 \) be the maximal flow rate, and let \( \delta_s \) be the damping parameter. The flow control parameters of session \( H \) are irrelevant at this point, simply because its source is outside of the scope. Moreover, throughout this section we’ll assume that the injection rate of session \( H \) into the gadget is 0. Servers \( Q_1 \) and \( Q_4 \) both have service rate 1. Server \( Q_2 \) has service rate \( 1 - \mu_2 \) and server \( Q_3 \) has service rate \( 1 - \mu_3 \), for some parameters \( 0 < \mu_2 < \mu_3 \). We summarize the parameters in Table I.

B. Basic gadget: a single time phase

In this subsection we describe the workings of a single time phase where we transfer one unit of session \( H \) fluid from
server $Q_1$ to server $Q_1$.

Let us define fresh fluid as the fluid from session $H$ that has never come through server $Q_1$ within our basic gadget, i.e., as the session $H$ fluid that still needs to go $K$ times through the big loop. Also, let us define processed fluid as the fluid from session $H$ that either has come through our basic gadget, or just needs to come through server $Q_4$ exactly once to exit the gadget.

At the beginning of the phase we assume that server $Q_1$ contains one unit of fresh fluid, and the three other servers are empty. The injection rate of session $S$ is at its maximal value $\varepsilon_s$. Moreover, we assume that throughout the phase, the injection rate of session $H$ is zero. We finish with one unit of processed fluid at server $Q_4$.

The behavior of the servers during the phase is summarized in Table I. During the time interval $[0, T_1]$, the height of server $Q_1$ decreases and the height of servers $Q_2, Q_3$ and $Q_4$ increases. During the time interval $[T_1, T_2]$, server $Q_1$ is empty, the height of server $Q_2$ decreases and the height of servers $Q_3$ and $Q_4$ increases. During the time interval $[T_2, T_3]$, servers $Q_1$ and $Q_3$ are empty, the height of server $Q_3$ decreases and the height of server $Q_4$ increases. We ensure that $T_3 = K/(1-\mu_3)$ and that $\varepsilon(T_3) \geq \varepsilon_s/2$.

We state our result as follows:

**Lemma III.1** Consider the basic gadget with a given flow decrease mechanism for session $S$ that satisfies properties (A3, A4). Assume that the injection rate into session $H$ is 0 at all times. Furthermore, suppose that at time $t = 0$ server $Q_1$ contains one unit of fresh fluid, the three other servers are empty, and the injection rate into session $S$ is at its maximal value $\varepsilon_s$.

Then there exist parameters $(\varepsilon_s, \delta_s, \mu_2, \mu_3, K')$ and times $(T_1, T_2, T_3, T_4)$ such that the gadget functions as shown in Table I. Moreover, this holds for any $K \geq 20$, in which case we can choose $\varepsilon_s \in (5/3K; 20/K)$ and set $\mu_3 = 2\varepsilon_s/5$.

Moreover, it turns out that after the parameters $(K, \varepsilon_s, \mu_3)$ are chosen in the above fashion, one can pick any $\mu_2 \in (0, \mu_3)$. This enables us to fine-tune such quantities as $T_2$ and the maximal height of server $Q_2$. Although we do not use this feature in the present proof, it is useful in the all-FIFO setting; see Section VI for details.

**C. Basic gadget: proof of the main lemma**

Let $h_1(t)$ be the height of server $Q_1$ at time $t$, and let $\varepsilon(t)$ be the injection rate of session $S$ at time $t$.

We set $\mu_3 = 2\varepsilon_s/5$ and assume $0 < \mu_2 < \mu_3$. At first session $H$ fluid is served by server $Q_1$ at rate 1. Since $\mu_2 > 0$, we start to get a buildup in server $Q_2$. Session $H$ fluid leaves server $Q_2$ at rate $1 - \mu_2$ which is larger than $1 - \mu_3$, so we start to get a buildup in server $Q_3$, too. Then session $H$ fluid leaves server $Q_3$ at rate $1 - \mu_3$ and arrives at server $Q_4$. At the beginning of the phase fluid also arrives to server $Q_4$ on session $S$ at rate close to $\varepsilon_s$. Since $1 - \mu_3 + \varepsilon_s > 1$ we get a buildup at server $Q_4$.

As server $Q_4$ is congested, $\varepsilon(t)$ starts to decrease; however, we’ll make sure that it stays large enough – say, at least $\varepsilon_s/2$ – until time $T_3$ when server $Q_4$ empties. Since $1 - \mu_3 + \varepsilon_s/2 > 1$, it follows that (at least) until this time $h_1(t)$ keeps increasing.

Moreover, until time $T_3$ session $H$ fluid loops back to server $Q_1$ after being served by $Q_4$ at rate less than 1 - $\mu_3$ (since $Q_4$ is also serving session $S$). In particular, $h_1(t)$ decreases until it becomes 0.

To summarize, initially the height of server $Q_1$ decreases, and the heights of the three other servers increase. At some time $t = T_1$, server $Q_1$ empties. At this point, the fluid that loops back from server $Q_4$ passes directly through server $Q_1$ and arrives at server $Q_2$ at a rate less than $1 - \mu_3$, which is less than $1 - \mu_2$. Therefore at time $T_1$ the height of server $Q_2$ starts to decrease, until at some time $t = T_2$ it empties. At this point session $H$ fluid starts to enter server $Q_3$ at a rate smaller than $1 - \mu_3$, so its height starts to decrease. We use $T_3$ to denote the first time that server $Q_3$ empties. At this point all of the session $H$ fluid is stored in server $Q_4$.

By definition of time $T_3$, at any time $t \in (0, T_3)$ session $H$ fluid leaves server $Q_3$ at the rate $1 - \mu_3$, so exactly $(1 - \mu_3)t$ units of session $H$ fluid are served by $Q_3$ between time 0 and $t$. Since only $K$ units of session $H$ fluid are available (more precisely, one unit that passes through the big loop $K$ times), it follows that $T_3 \leq T$, where $T = K/(1 - \mu_3)$.

From now on, let us fix parameter $K$ and vary parameter $\varepsilon_s$. Define constants $\varepsilon_1 = 5/3K$ and $\varepsilon_2 = 20/K$. Later we will choose $\varepsilon_s$ from the interval $[\varepsilon_1; \varepsilon_2]$.

**Claim III.2** There exists a positive constant $\delta^*$ such that

$$\varepsilon(2K) = f_{\delta^*}(\varepsilon(2K)) \geq \varepsilon/2 \text{ for any } \varepsilon \in [\varepsilon_1; \varepsilon_2].$$

(9)
Proof: Let us denote $\alpha_i = \varepsilon_1 (3/2)^i$ and let us cover the interval $[\varepsilon_1; \varepsilon_2]$ with smaller intervals $[\alpha_i; \alpha_{i+1}]$. For each such interval let us choose $\delta = \delta_i$ such that $f_\delta(\alpha_i, 2K) \geq \alpha_{i+1}/2$; such $\delta$ exists by Axiom (G) since $\alpha_{i+1}/2 < \alpha_i$. Then (11) is satisfied for any $\delta \geq \delta_i$ and $\varepsilon \in [\alpha_i; \alpha_{i+1}]$. Finally, let us choose $\delta^*$ to be the smallest of the $\delta_i$'s. □

Let us set the damping parameter $\delta_s$ to $\delta^*$ from Claim III.2. Note that since $\mu_3 = 2\varepsilon_3/5$ it follows that $T < 2K$, so

$$\varepsilon(T) \geq \varepsilon_s/2 \text{ for any } \varepsilon_s \in [\varepsilon_1; \varepsilon_2].$$

(10)

We choose parameter $\varepsilon_s$ carefully to ensure that $T_3 = T$, or, equivalently, that at time $t = T_3$ all of the session $H$ fluid has passed through servers $Q_1$, $Q_2$ and $Q_3$, exactly $K$ times. More precisely, we have the following claim whose proof is deferred to Section III.C.

Claim III.3 For any $K \geq 20$ there exists $\varepsilon_s \in (5/3K; 20/K)$ such that for any $\mu_3 \in (0, \mu_3)$ we have $T_3 = K/(1 - \mu_3)$.

In the remainder of the phase, all of the session $H$ fluid has to do is loop through server $Q_4$ another $K' + 1$ times, which takes time at least $K' + 1$. For any given $K$ (and hence $\varepsilon_s, \delta_s$ and $T_3$), we choose parameter $K'$ to be the smallest integer such that by the time $T_3 + K'$ the injection rate of session $S$ has dropped to zero, and all of the session $S$ fluid has left the system. Clearly, such $K'$ exists for any given $K$.

The phase ends at some time $t = T_P$ when all of the session $H$ fluid has exactly one loop to go in $Q_4$. Obviously, at this time all of this fluid is still queued in $Q_4$; since all of the session $S$ fluid has already left the system, $h_4(T_P) = 1$ as required at the end of the phase.

To complete the proof of Lemma III.1 it remains to prove Claim III.3.

D. Basic gadget: proof of Claim III.3

Let us keep $K$ fixed and assume that $\varepsilon_s \in [5/3K; 20/K]$. Recall that $\varepsilon(t)$ is the injection rate into session $H$ at time $t$, and $h_i(t)$ is the height of server $Q_i$ at time $t$. Recall that we use $T_i$ to denote the first time that server $Q_i$ empties:

$$T_i = \min\{t > 0 : h_i(t) = 0\}.$$

We will also use $d_i(t)$ to denote the rate at which session $H$ fluid departs from server $Q_i$ at time $t$.

A key quantity in our analysis is the total amount of session $H$ fluid queued in servers $Q_1$, $Q_2$ and $Q_3$. We use $h^+(t)$ to denote this quantity at time $t$. We note that $h^+$ behaves like the amount of fluid in a single queue that is fed at rate $d_4(t)$ and has service rate $1 - \mu_3$. Since $\varepsilon(T_3) \geq \varepsilon_s/2$, by the analysis in the previous section server $Q_3$ drains after servers $Q_1$ and $Q_2$. It follows that

$$T_3 = \min\{t > 0 : h^+(t) = 0\}$$

and, in particular, $h^+(T_3) = 0$.

For any time $t \in [0; T_3]$ the following equations hold:

$$d_3(t) = 1 - \mu_3$$

(11)

$$h_4(t) = \int_0^t (d_3(t) + \varepsilon(t) - 1) \, dt$$

(12)

$$h^+(t) = 1 + \int_0^t (d_4(t) - 1 + \mu_3) \, dt$$

(13)

$$d_4(t + h_4(t)) = \frac{1 - \mu_3}{\varepsilon(t) + 1 - \mu_3}$$

(14)

Let $T = K/(1 - \mu_3)$ and recall that $T_3 \leq T$ and $\varepsilon(T)$ is at least $\varepsilon_s/2$ by (10). We'd like to show that $T = T_3$ if and only if $h^+(T) = 0$, and then solve $h^+(T) = 0$ to obtain the desired dependency between $\varepsilon_s$ and $K$. However, under the current definition of $h^+(t)$ it is trivially 0 for any $t \geq T_3$, including $t = T$.

To remedy this, let us formally extend equations (11)-(13) to $t \in [0; 2K]$. More precisely, let us forget the original definitions of functions $d_3(t)$, $d_4(t)$, $h_4(t)$ and $h^+(t)$, and redefine them by these equations; note that these functions are uniquely determined by these equations. Obviously, the new definitions coincide with the old ones for $t \in [0, T_3]$.

By (9) it follows that $h^+(t)$ is strictly decreasing for all $t \in [0; 2K]$, so $t = T_3$ is its only root. Therefore $T = T_3$ if and only if $h^+(T) = 0$. By (14),

$$h^+(T) = 1 - (1 - \mu_3)T + \int_0^T d_4(t) \, dt.$$ (15)

Suppose we keep $K$ fixed and vary $\varepsilon_s$. Then the integral in (15) becomes a function of $\varepsilon_s$; let us denote it by $F(\varepsilon_s)$. This function is continuous; this is intuitively clear but nevertheless requires a rigorous proof, see Appendix II for details.

Let us define a function

$$g(y) = \frac{1 - \mu_3}{y + 1 - \mu_3}$$

and observe that it is strictly decreasing since $g'(y) < 0$ for all $y$. We claim that

$$d_4(t) \in [g(\varepsilon_s); g(\varepsilon_s/2)] \text{ for any } t \in [0, T].$$ (16)

Indeed, since $\varepsilon(t) \in [\varepsilon_s/2; \varepsilon_s]$ for all $t \in [0, T]$, it is easy to see that $h_4(t)$ is strictly increasing. Therefore the function $x(t) = t + h_4(t)$ is continuous and strictly increasing, so for any $x \in [0; T]$ there exists some $t \in [0; T]$ such that $x = x(t)$. It follows that

$$d_4(x) = d_4(x(t)) = g(\varepsilon(t)) \in [g(\varepsilon_s); g(\varepsilon_s/2)],$$

claim proved.

Then $F(\varepsilon) \in [F^-(\varepsilon); F^+(\varepsilon)]$, where

$$\begin{cases}
F^-(\varepsilon) = T \times g(\varepsilon) = K/(1 + 3\varepsilon/5), \\
F^+(\varepsilon) = T \times g(\varepsilon/2) = K/(1 + \varepsilon/10)
\end{cases}$$
It follows that
\[
\begin{cases}
F\left(\frac{20}{17}\right) \leq F^+\left(\frac{20}{17}\right) = \frac{K^2}{K+2} < K - 1 \\
F\left(\frac{17}{20}\right) \geq F^-\left(\frac{17}{20}\right) = \frac{K^2}{K+1} > K - 1
\end{cases}
\]

Therefore by continuity of \(F(\varepsilon)\) for any \(K \geq 20\) there exists some \(\varepsilon_0 \in (5/3K; 20/K)\) such that \(F(\varepsilon_0) = K - 1\). Let us set \(\varepsilon_a = \varepsilon_0\).

IV. FULL CONSTRUCTION

In this section we’ll define the full construction and prove that it has the periodicity claimed in Theorem [1]. We’ll use the basic gadget from Section [III] as the main building block. Recall that this gadget has several tunable parameters (see Table [I]). All gadgets we use will be identical, i.e. will have the same parameters. Specifically, we set \(K = 40\) and use Lemma [III] to get parameters \((\varepsilon, \delta, \mu_1, \mu_2, \mu_3, K')\) such that \(\varepsilon_a \leq 1/2\), and the corresponding times \((T_1, T_2, T_3, T_4)\).

A. Full construction: the layout

As we mentioned in Section [III-A] we organize gadgets in rows, so that the same horizontal session \(H\) goes consecutively through all gadgets in a given row. Formally, a sequence of basic gadgets is merged into a row of gadgets as follows.

Recall that each basic gadget \(G\) contains four servers \(Q_i = Q_i(G)\), and the horizontal session path that we denote \(P(G)\); recall that this path includes cycles and loops. Let \(G_0, G_1, \ldots, G_n\) be a sequence of gadgets. Firstly, two new nodes \(s, t\) are introduced; they designated as the source and the sink of the horizontal session \(H\), respectively. Secondly, for each \(i\) we identify server \(Q_i(G_i)\) from gadget \(G_i\) with server \(Q_1(G_{i+1})\) from gadget \(G_{i+1}\). Thirdly, we define the session path for \(H\) as
\[
(s, q) \oplus P(G_0) \oplus \ldots \oplus P(G_n) \oplus (q', t),
\]
where \(q = Q_1(G_0)\) and \(q' = Q_4(G_n)\) are two servers, and \(P \oplus P'\) denotes the concatenation of two paths \(P\) and \(P'\) such that the last node of \(P\) is the first node of \(P'\). In particular, session \(H\) goes through each server \(Q = Q_4(G_i)\) for the total of \(2K + K'\) times: first \(K\) times in the big loop of gadget \(G_i\), then \(K'\) times in the small loop of the same gadget, and then another \(K\) times in the big loop of gadget \(G_{i+1}\).

Several consecutive rows will form a super-row. We will have \(N_{\text{sup}}\) super-rows, of \(N_{\text{rows}}\) rows each; each row will consist of \(N_{\text{cols}}\) gadgets. Here \(N_{\text{sup}}, N_{\text{rows}}\) and \(N_{\text{cols}}\) are parameters that we will tune later. Let us number rows and super-rows starting from 0, in the downward direction. In a given row, let the horizontal session flow from left to right; let us number gadgets from left to right, starting from 0. Let \(G_{(i,j,k)}\) be the \(k\)-th gadget in the \(j\)-th row of super-row \(i\). The \(N_{\text{rows}}\) gadgets \(G_{(i,j,k)}, j \in [N_{\text{rows}}]\) will form a column \(C_{ik}\). So each super-row \(i\) consists of \(N_{\text{cols}}\) columns \(C_{ik}, k \in [N_{\text{cols}}]\).

Recall that we have a distinct horizontal session for each row. All horizontal sessions have the same flow control parameters: we let \(\varepsilon_h\) be the maximal flow rate, and \(\delta_h\) be the damping parameter. In addition to horizontal sessions and simple sessions from Section [III] we’ll have a new type of session: vertical sessions. All such sessions have the same flow control parameters: we let \(\varepsilon_v\) be the maximal flow rate, and \(\delta_v\) be the damping parameter.

Each server \(Q_1\) in the very first gadget of each row is called a replenishing server. Each vertical session \(V\) goes through exactly one such server, call it \(Q_V\). It will be the case that exactly \(N_{\text{vert}}\) vertical sessions go through each replenishing server, periodically flooding it with vertical fluid, causing build-up (replenishing) of the blocked horizontal fluid. See Section [IV-H] for more details.

For each replenishing server \(Q\) there is a distinct new server with service rate \(\mu_D \geq N_{\text{vert}}\varepsilon_v\) that we call the decelerating server. For each vertical session \(V\) such that \(Q = V_Q\) this is the last server that \(V\) goes through before it reaches its sink. In case there is a build-up of vertical fluid, this server acts as a bottleneck, making sure that the build-up persists (and thus session \(V\) stays blocked) for a sufficiently long time.

Each vertical session \(V\) is associated with some column \(C_{ik}\). Session \(V\) injected into server \(Q_2\) of the first gadget of this column, and goes consecutively through servers \(Q_2\) of all gadgets in the column. Then \(V\) goes through the corresponding replenishing server \(Q_V\), then it goes through the corresponding decelerating server, and then it proceeds to its sink. At most one vertical session passes through a given column. Some columns do not contain a vertical session; those are called blank. We need to specify which columns are blank, and which vertical sessions connect to which replenishing queues. Let us denote this information as a collection \(C_{\text{prof}}\) of all 4-tuples \((i, k, l, j)\) such that column \(C_{ik}\) contains a vertical session \(V\), and the corresponding replenishing server \(Q_V\) lies in row \(j\) of super-row \(l\). Collection \(C_{\text{prof}}\) describes connectivity between the super-rows; thus we call it connectivity profile.

We review our construction in Figure [3]. We will specify the connectivity profile in Section [IV-H]. We also need to choose the values for various parameters, see Table [III] for a list. We do it later in this section.

B. Full construction: the max-stable flow constraints

We say that at a given time a session \(S\) is max-stable if it is stable and is injecting at its maximal injection rate. Let us say that a super-row \(R\) is in a max-stable state if all horizontal sessions in \(R\) are in a max-stable state, and all vertical sessions in the columns of \(R\) are in a max-stable state. To ensure

| Parameter       | Description                                                                 |
|-----------------|------------------------------------------------------------------------------|
| \((\varepsilon_v, \delta_v)\) | flow control parameters for the vertical sessions                            |
| \((\varepsilon_h, \delta_h)\) | flow control parameters for the horizontal sessions                          |
| \(\mu_D\)       | service rate for the decelerating servers, \(\mu \geq \varepsilon_v\)         |
| \(N_{\text{vert}}\) | #vertical sessions in each replenishing server                              |
| \(N_{\text{cols}}\) | #columns                                                                    |
| \(N_{\text{rows}}\) | #rows in a super-row                                                        |
| \(N_{\text{sup}}\) | #super-rows                                                                 |
| \(\Delta t\)    | time shift between two consecutive rows in a super-row                      |
| \(\Delta T\)    | time shift between two consecutive super-rows                               |
| \(\Delta T^*\)  | duration of a row replenishing process                                      |

TABLE III
TUNABLE PARAMETERS IN THE FULL CONSTRUCTION
Let us introduce some machinery needed to describe and reason about such processes. Let us start by defining formally what is a state in our construction. The state of session \( S \) at a given time \( t \) is a record \((Q, t, . . . )\) recording the injection rate of \( S \) at time \( t \), whether at time \( t \) the session is blocked or happy or stable, and (if it has a positive injection rate and it is not stable) the last time it switched between these three modes. The state of server \( Q \) at a given time \( t \) is a set of records \((Q, t, S, . . . )\) which for each session \( S \) coming through server \( Q \) describes the session \( S \) fluid buffered at \( Q \) at time \( t \).

Let \( N \) be the set of all servers and sessions. For each subset \( N' \subset N \) the state of \( N' \) is the union of states of all elements of \( N' \). Note that the state of \( N \) at time \( t_0 \) determines the state of \( N \) at any future time \( t > t_0 \). We will also argue about the dependencies between the states of different rows and super-rows in our construction. Finally, let us define the process in \( N' \subset N \) in time interval \( I \) as the union of states of \( N' \) for all times \( t \in I \).

Now we can introduce a more concrete definition of the basic time phase. In a gadget, the \( B(t) \)-process is the basic time phase which starts at time \( t \) and ends at time \( t + T_p \). We will ensure that each gadget \( G_{(i,j,k)} \) undergoes the \( B(t) \)-process for each

\[
t = (nN_{sup} + i)\Delta T + j\Delta t + kT_p, \quad n \in \mathbb{N},
\]

where parameters \( \Delta T \) and \( \Delta t \) are the same for all gadgets. Each gadget will undergo exactly one \( B(t) \)-process every \( N_{sup}\Delta T \) time units. The rest of the time it will be “idle”; we’ll make it more precise later. In particular, all gadgets in the same row will follow the same process, but the processes in two consecutive gadgets are shifted by time \( T_p \). Similarly, all rows in the same super-row will follow the same process, but the processes in two consecutive rows are shifted by time \( \Delta t \); all super-rows will follow the same process, but the processes in two consecutive super-row are shifted by time \( \Delta T \).

Recall that we fixed the value for \( T_p \) when we invoked Lemma[11]. We want to define \( \Delta t \) such that \( 0 < \Delta t < T_2 \) and the ratio \( T_p/\Delta t \) is an odd integer; we will use these properties in the proof of Lemma[IV.8] to enable equation [17]. Let us set

\[
\Delta t := T_p/r, \quad \text{where } r := 2\lfloor T_p/T_2 \rfloor + 1.
\]

We will specify \( \Delta T \) later in this subsection.

We introduced rows of gadgets in order to enable the following intuitive claim:

**Claim IV.1** If for time \( t \) some gadget in a row undergoes the \( B(t) \)-process, and the next gadget in this row is ‘nice’, then this next gadget undergoes the \( B(t+T_p) \)-process.

Here the next gadget being ‘nice’ means that in this gadget, servers \( Q_2, Q_3 \) and \( Q_4 \) are empty of horizontal and simple sessions’ fluid, and the simple session is max-stable. We will need to use this intuitive claim in the subsequent proofs about states of the full construction. For this purpose we need to express what happens in this lemma in terms of states, and to do that we need a somewhat more complicated formalism.

To define precisely what we mean by the state of row \( R \), let us think of \( R \) as a system that contains the horizontal

---

**Fig. 3.** Super-rows for \( N_{rows} = 4 \) and \( N_{vert} = 2 \).

that such state is possible, we need to satisfy the max-stable flow constraints: for each server \( Q \), the sum of the maximal injection rates of all sessions served by \( Q \) is no more than this server’s service rate. Let us list these constraints:

\[
\begin{align*}
K\varepsilon_h + \varepsilon_v N_{vert} &\leq 1 & (\text{at each replenishing server } Q_1) \\
(2K + K')\varepsilon_h + \varepsilon_v &\leq 1 & (\text{at each non-replenishing } Q_1) \\
K\varepsilon_h + \varepsilon_v &\leq 1 - \mu_2, & (\text{at each server } Q_2) \\
K\varepsilon_h &\leq 1 - \mu_3, & (\text{at each server } Q_3) \\
N_{vert}\varepsilon_v &\leq \mu_0 & (\text{at each decelerating server})
\end{align*}
\]

To satisfy these constraints, let us set

\[
\begin{align*}
N_{vert} &= \lfloor 1/(\mu_3 - \mu_2) \rfloor \\
\varepsilon_h &= 1/(8K + 4K') \\
\varepsilon_v &= 1/4N_{vert}
\end{align*}
\]

and let us enforce that

\[
\varepsilon_h \leq \varepsilon_h \quad \text{and} \quad \varepsilon_v \leq \varepsilon_v \quad \text{and} \quad \mu_0 \geq 1/4.
\]

Since we chose \( \varepsilon_v \), the maximal flow rate for a simple session, to be at most 1/2, [17] implies all max-stable flow constraints.

**C. Full construction: the replenishing process**

In this subsection we will consider the dynamics of our construction. We will build on the dynamics of the basic time phase described in Section III. In particular, in a given row this process will happen in each gadget, one gadget at a time, consecutively from left to right. Different rows will follow the same process, shifted in time by a certain amount.
Claim IV.2 Consider a row $R$ in our construction. For convenience, let us denote $t_s = N_{\text{cols}}T_p + 1$.

(a) Suppose at time $t$ row $R$ is in a $P_k$-state, for some $k$ such that $1 \leq k < N_{\text{cols}}$. Then row $R$ undergoes the $B(t)$-process, and at time $t + T_p$ it is in a $P_{k+1}$-state.

(b) Suppose at time $T_p$ row $R$ is in the partial state $P_1$. Then for each time $t \in [T_p, t_s]$ the partial state of this row is fixed, call it $P(t)$. In particular, $P(kT_p) \in P_k$ for each $k$ such that $1 \leq k \leq N_{\text{cols}}$, and $P(t_s) \in P_{\text{empty}}$.

(c) Suppose at some time $t \in [T_p, t_s]$ row $R$ is in the partial state $P(t)$. Then for each time $t' \in [t, t_s]$ the partial state of this row is $P(t')$.

In fact each row will be initialized in partial state $P_0$, but in this case the behavior of (the partial state of) the row depends on what happens outside this row; this is why we chose partial state $P_1$ as the starting point for $P(t)$.

Claim IV.3 Suppose at time 0 row $R$ is in partial state $P_0$, and that during time interval $[0; T_p]$ no vertical fluid enters the replenishing server of $R$. Then this row undergoes the $B(0)$-process, and at time $T_p$ it is in partial state $P_1$.

Let us say that a given super-row $i$ includes the corresponding rows, decelerating servers in these rows, and all vertical sessions $V_{ik}$. Let us define the partial state of this super-row as the partial state of all rows. For each $k \leq N_{\text{cols}}$ such that $kT_p - N_{\text{rows}} \Delta t \geq T_p$

let us define a partial state $S_k$ such that each row $j$ is in partial state $P(kT_p - j \Delta t)$. (See Figure 5)

Let us translate Claim IV.2 into the corresponding claim into partial states of a super-row. For convenience let us introduce the following conventions:

$$\begin{align*}
  t_s &= N_{\text{cols}}T_p + 1 \\
  k_0 &= 1 + \lfloor N_{\text{rows}}/r \rfloor \\
  t_0 &= k_0T_p
\end{align*}$$

Claim IV.4 Consider a super-row $i$ in our construction.

(a) Suppose at some time $t$ super-row $i$ is in partial state $S_k$, for some $k$ such that $k_0 \leq k < N_{\text{cols}}$. Then at time $t + T_p$ it is in partial state $S_{k+1}$.

(b) Suppose at time $k_0T_p$ a given super-row is in partial state $S_{k_0}$. Then for each time $t \in [t_0; t_s]$ the partial state of this super-row is fixed, call it $S(t)$. Moreover, it is the case that $S(kT_p) = S_k$ for all $k$ such that $k_0 \leq k \leq N_{\text{cols}}$.

(c) Suppose at some time $t \in [t_0; t_s]$ a given super-row is in partial state $S(t)$. Then for each time $t' \in [t, t_s]$ this super-row is in partial state $S(t')$.

Let us say that a partial process in a super-row in time interval $I$ is a union of all partial states in $I$. The above claim defines the partial process

$$\{S(t) : t \in [t_0; t_s]\}$$

For each non-blank column $C_{ik}$, let $V_{ik}$ be the vertical session that goes through $C_{ik}$. For each such session, we would like to understand how it interacts with partial process (22) in super-row $i$. Intuitively, for each row $R$ of this super-row all interesting interaction with session $V_{ik}$ happens when row $R$ is between partial states $P_k$ and $P_{k+1}$. For technical convenience we want partial process (22) to be well-defined whenever any row is between these two partial states. Therefore we will enforce that for any vertical session $V_{ik}$ we have

$$k_0 \leq k < N_{\text{cols}} - k_0$$
In partial process \(\mathcal{R}\), all horizontal fluid eventually drains to its sinks. As we want our construction to exhibit oscillations, we need a way to get back to partial state \(S_k\). Let us consider this problem for each row separately: we start with an empty row, and we want to get back to partial state \(\mathcal{P}_1\); most crucially, we need to replenish the horizontal fluid in server \(Q_1\) in column 0.

Before we can state our basic result on replenishing, we need to build some machinery. Say row \(R\) is max-stable if the horizontal session and all simple sessions are max-stable, and all servers contain no horizontal or vertical fluid. We would like to define a process that turns row \(R\) from a max-stable state to partial state \(\mathcal{P}_1\); we will call it a row replenishing process. Apart from partial state \(\mathcal{P}_1\), we need to ensure that row \(R\) behaves ‘nicely’ towards vertical sessions that come through \(R\). Specifically, say row \(R\) is open if for each non-blank column in this row, server \(Q_2\) (in the gadget in this column) has at least \(\varepsilon_v\) units of capacity for the vertical session that flows through this column.

Let us formulate the initial conditions for our replenishing process. Recall that each vertical session \(V\) goes through some replenishing server \(Q_v\). For a given replenishing server \(Q\) in row \(R\), let us define the inverse mapping \(V_Q\) as the set of all vertical sessions \(V\) such that \(Q = Q_v\). Say our construction is \(Q\)-synchronized if at time \(t\) the following conditions hold:

- Row \(R\) is max-stable: for each session \(V_{ik} \in V_Q\), \(V_{ik}\) is max-stable and satisfies (24)
- and super-row \(i\) is in partial state \(S_k\).

The main result of this subsection is (essentially) that if our construction is \(Q\)-synchronized, then a certain time later row \(R\) is in partial state \(\mathcal{P}_1\).

**Lemma IV.5** There exist parameters \((\varepsilon_h, \delta_h, \varepsilon_v, \delta_v, \mu_D)\) and positive integers \((L_1, L_2)\) such that the max-stable flow constraints hold, and for \(N_{rows} = rL_1\) and any choice of \((\Delta T, N_{cols}, N_{sup}, C_{prof})\) we have the following property: (a) Let \(Q\) be a replenishing server such that \(|V_Q| = N_{vert}\). Suppose our construction is \(Q\)-synchronized at time \(t\). Then at time \(t + \Delta T^*\) server \(Q\)’s row is in partial state \(\mathcal{P}_1\), and each session \(V_{ik} \in V_Q\) is happy. Moreover, between time \(t\) and \(t + \Delta T^*\) this row is open. Here

\[
\Delta T^* = (L_1 + L_2 + 1) T_p - \Delta t.
\]

We defer the proof to Section IV-D. We would like to make several remarks. First, note that property (a) is conditional: in this lemma we do not prove that our construction ever becomes \(Q\)-synchronized. Second, we achieved this property by tuning parameters \((\varepsilon_h, \delta_h, \varepsilon_v, \delta_v, \mu_D, N_{rows})\). Note that we did not impose any constraints on parameters \((\Delta T, N_{sup}, N_{cols}, C_{prof})\). Third, we had a lot of flexibility in choosing the separation time \(\Delta T^*\). We used this flexibility to enforce (25), which we will use in order to make our construction \(Q\)-synchronized at the appropriate times. Specifically, we will use (25) in the proof of Lemma IV.8 in order to pass from \(25\) to \(26\).

Let us conclude this subsection by attaching appropriate names to the process described in Lemma IV.5. Let us say that a \((Q, t)\)-replenishing process is (any) process that happens in our construction between times \(t\) and \(t + \Delta T^*\) if at time \(t\) it becomes \(Q\)-synchronized. By Lemma IV.5 during a \((Q, t)\)-replenishing process the corresponding row goes from being empty at time \(t\) to partial state \(\mathcal{P}_1\) at time \(t + \Delta T^*\). Informally, we say that this row gets replenished.

Let us define the analogous replenishing process for a given super-row \(i\). Let \(Q^{(j)}\) be the replenishing server in row \(j\) of this super-row, so that all rows experience the same replenishing process, appropriately time-shifted. Let us say that a \((i, t)\)-replenishing process is (any) process in our construction that starts at time \(t\) and proceeds for time \((N_{rows} - 1) \Delta t + \Delta T^*\), such that for each row \(j\) in super-row \(i\), a \((Q^{(j)}, t + j \Delta t)\)-replenishing process happens.

**D. Full construction: proof of Lemma IV.5**

Consider a session with maximal flow rate \(\varepsilon_v\) and damping parameter \(\delta_v\). Recall that \(f_{\mathcal{R}}\) is the parameterized flow decrease function (see Section I-B). Suppose at time \(0\) the session is injecting at rate \(\varepsilon_v\), then becomes blocked and stays blocked. Then by axiom (A2) the injection rate starts decreasing until at some finite time \(T_{stop} = T_{stop}(\varepsilon, \delta)\) it becomes 0. Let

\[
A(\varepsilon, \delta, t) := \int_0^t f_{\mathcal{R}}(\varepsilon, t) dt
\]

be the amount of fluid that is injected by time \(t\). Let

\[
B(\varepsilon, \delta) := \int_0^\infty f_{\mathcal{R}}(\varepsilon, t) dt = A(\varepsilon, \delta, T_{stop})
\]

be the amount of fluid that is injected by the time the flow stops. Let \(B_v = B(\varepsilon_v, \delta_v)\) and \(B_h = B(\varepsilon_h, \delta_h)\) be these amounts for vertical and horizontal sessions, respectively.

Let us fix a choice of \((N_{cols}, N_{sup}, C_{prof})\), suppose \(Q\) is a replenishing server such that with respect to this choice \(|V_Q| = N_{vert}\), and assume that our construction is \(Q\)-synchronized at time \(t\). Let us focus on some vertical session \(V = V_{ik} \in V_Q\). The path of this session goes through \(N_{rows}\) consecutive gadgets in column \(C_{ik}\). Let \(G_j = G(i, j, k)\) be the \(j\)-th such gadget, and let \(Q^{(j)}\) be the server \(Q_2\) from \(G_j\). Say such server is blocked if it stores a positive amount of horizontal fluid.

Then, since at servers \(Q^{(j)}\) the horizontal fluid has priority over the vertical fluid, session \(V\) is blocked if at least one server \(Q^{(j)}\) is blocked.

Recall that at time \(t\) super-row \(i\) is in partial state \(S_k\). Let us consider what happens with session \(V\) after time \(t\). Until time \(t\) session \(V\) is in the max-stable state. Then each server \(Q^{(j)}\) is blocked starting from time \(t + j \Delta t\) for time \(T_2\), so session \(V\) is blocked starting from time \(t\) for time

\[
t^* := (N_{rows} - 1) \Delta t + T_2.
\]

We will make sure that by this time the injection rate into session \(V\) goes from \(\varepsilon_v\) to 0:

\[
T_{stop}(\varepsilon_v, \delta_v) \leq t^*.
\]

Then \(B_v\) units of session \(V\) fluid is accumulated in server \(Q\) at time \(t + t^*\). From this time on, all servers \(Q^{(j)}\) become unblocked; in each of them, horizontal fluid eats up at most
1 – \(\mu_3\) units of bandwidth (this is by Lemma \(\text{III.1}\)), so at least \(\mu_3 - \mu_2\) units of bandwidth are available to session \(V\). Therefore, starting from time \(t + t^*\) session \(V\) fluid drains down into server \(Q\) at rate at least \(\mu_3 - \mu_2\), until at some time \(t_2\) all \(B_3\) units of fluid are gone.

Since this happens for all \(N_{vert}\) vertical sessions \(V \in V_Q\), between times \(t + t^*\) and \(t_2\) the total incoming rate of vertical sessions into server \(Q\) will be at least \(N_{vert}(\mu_3 - \mu_2)\), which is at least \(1\) by definition of \(N_{vert}\). Therefore at time \(t + t^*\) vertical fluid will immediately start building up in server \(Q\). Since at \(Q\) the vertical fluid has priority over the horizontal fluid, this build-up will persist exactly for time \(N_{vert}B_v\). During all this time vertical fluid exits server \(Q\) and enters the corresponding decelerating server \(D_Q\) at rate \(1\). We will choose the service rate \(\mu_D\) for \(D_Q\) so that \(\mu_D < 1\). Therefore after leaving server \(Q\), vertical fluid builds up in server \(D_V\) and, moreover, this build-up drains down to the sink only after all vertical session fluid drains down from \(Q\). Therefore a total of \(N_{vert}B_v\) units of vertical fluid drains down server \(D_V\), at a constant rate \(\mu_D\), taking time \(N_{vert}B_v/\mu_D\). During this time interval, the injection rate into each session \(V \in V_Q\) is \(0\). We will ensure that this time interval is sufficiently long, namely that

\[
N_{vert}B_v/\mu_D = N_{vert}B_v + T_p. \tag{28}
\]

Note that if (28) holds then \(\mu_D < 1\).

Let \(H\) be the horizontal session coming through server \(Q\). Before time \(t + t^*\) session \(H\) is in the max-stable state. Recall that server \(Q\) holds a non-zero amount of vertical fluid starting from time \(t + t^*\), for time \(N_{vert}B_v\). During this time interval, session \(H\) is blocked. We will tune the parameters so that during this time interval the injection rate of session \(H\) goes from \(\varepsilon_h\) down to 0:

\[
T_{stop}(\varepsilon_h, \delta_h) \leq N_{vert}B_v, \tag{29}
\]

and meanwhile exactly 1 unit of session \(H\) fluid accumulates in server \(Q\):

\[
B_h = 1. \tag{30}
\]

If equations (28, 30) hold, then at time \(t + t^* + N_{vert}B_v\) the row \(\mathcal{R}\) containing server \(Q\) in partial state \(P_0\). By Claim \(\text{IV.3}\) this row is in partial state \(P_1\) time \(T_p\) later, i.e. at time \(t + \Delta T^*\) where

\[
\Delta T^* = t^* + N_{vert}B_v + T_p. \tag{31}
\]

Moreover, by (28) at this time the decelerating server of row \(\mathcal{R}\) is empty, and therefore all vertical sessions \(V \in V_Q\) are happy. Row \(\mathcal{R}\) is open between time \(t\) and \(t + \Delta T^*\) since there was no build-up of horizontal fluid at time \(t\), and the horizontal session stayed blocked between time \(t\) and \(t + \Delta T^*\).

To complete the proof of Lemma \(\text{IV.5}\) it remains to match (31) with the definition (25) of \(\Delta T^*\).

Note that by definition (26) of \(t^*\) and since

\[
N_{vert} = L_1[T_p/T_2] \Delta t = L_1T_p,
\]

we can write \(t^*\) as a function of \(L_1\):

\[
t^* = L_1T_p + T_2 - \Delta t.
\]

In particular, by (31) equation (25) becomes equivalent to

\[
N_{vert}B_v = L_2T_p - T_2. \tag{32}
\]

Now to prove Lemma \(\text{IV.5}\) it remains to choose parameters \((\varepsilon_h, \delta_h, \varepsilon, \delta, \mu_D)\) and integers \((L_1, L_2)\) so that equations (17), (27), (30) and (32) hold. We need to be careful in order to avoid circular dependencies between the parameters. This is how we overcome this hurdle:

1. choose \((\varepsilon_h, \delta_h)\) so that \(\varepsilon_h \leq \varepsilon_h^*\) and \(B_h = 1\).
2. define

\[
L_2 := 1 + [(T_{stop}(\varepsilon_h, \delta_h) + T_2)/T_p]
\]

\[
T_0 := (L_2T_p - T_2)/N_{vert}.
\]

3. choose \((\varepsilon, \delta, \mu)\) so that \(\varepsilon \leq \varepsilon^*\) and \(B_v = T_0\).
4. choose \(L_1\) large enough so that (27) holds.
5. choose \(\mu_D\) so that (28) holds.

Note that at the third step we ensure that (29) and (32) hold. At the fifth step, by (32) we have \(N_{vert}B_v > T_p\), so \(\mu_D > 1/2\) as required. It remains to show that we can indeed do the first and the third step. We establish this via the following claim:

**Claim IV.6** If axioms (A1, A5) hold, then for any given positive \((\varepsilon^*, b)\) there exist flow control parameters \((\varepsilon, \delta)\) such that \(\varepsilon \leq \varepsilon^*\) and \(B(\varepsilon, \delta) = b\).

**Proof:** Choose parameter \(\delta\) so that \(B(\varepsilon^*, \delta) \geq b\), i.e. at least \(b\) units of fluid are injected if the session becomes blocked when it is transmitting at rate \(\varepsilon^*\); such \(\delta\) exists by axiom (A4). Now that \(\delta\) is fixed, let us tune \(\varepsilon\) in the interval \([0, \varepsilon^*]\) so that exactly \(b\) units of fluid are injected. We do it rigorously as follows.

Let us write \(T_{stop}(\varepsilon) = T_{stop}(\varepsilon, \delta)\); by Axiom (A5) this is an increasing function of \(\varepsilon\). The function

\[
A(\varepsilon) := A(\varepsilon, \delta, T_{stop}(\varepsilon^*))
\]

is continuous by axiom (A5), see Fact 2.3 for rigor. Since \(A(0) = 0\) and \(A(\varepsilon^*) \geq b\), we can choose \(\varepsilon \leq \varepsilon^*\) such that \(A(\varepsilon) = b\). Since \(T_{stop}(\varepsilon) \leq T_{stop}(\varepsilon^*)\), it follows that \(B(\varepsilon, \delta) = b\), too. \(\square\)

This completes the proof of Lemma IV.5.

**E. Full construction: the connectivity profile**

In this subsection we will choose the connectivity profile \(C_{prof}\). We will assume that the parameters are chosen as per Lemma IV.5 and that \(\Delta T = 2\Delta T^*\).

Recall that \(C_{prof}\) is the collection of all 4-tuples \((i, k, l, j)\) such that vertical session \(V_{ik}\) goes through the replenishing server which lies in row \(j\) of super-row \(l\). Let us impose some simple conditions on the connectivity profile.

**Definition IV.7** Let us say the connectivity profile is well-formed if it has all of the following properties:

(a) for each pair \((i, k)\) there is at most one \((i, k, l, j) \in C_{prof}\).
(b) for each tuple \((i, k, l, j) \in C_{prof}\) we have \(k \geq k_0 + 1\).
(c) for each replenishing server \(Q\) we have \(|V_Q| = N_{vert}|.\)
(d) $C_{\text{prof}}$ is not changed by a circular permutation of super-rows: $(i, k, l, j) \in C_{\text{prof}}$ if and only if $(0, k, l^*, j) \in C_{\text{prof}}$, for $l^* = (l - j) \pmod{N_{\text{sup}}}$. Let us reverse-engineer some other useful properties of $C_{\text{prof}}$ from the requirement that for each replenishing queue $Q$, our construction needs to be $Q$-synchronized at the appropriate time. Let us go back to [13], this is when we would like a given gadget to undergo a $\text{B}(t)$ process. It follows that for each column $C_{ik}$ we would like the corresponding super-row $i$ to be in partial state $S_k$ at time

$$\tau_{ik} := i \Delta T + k T_P.$$ 

Similarly, for each replenishing server $Q$ in row $j$ of super-row $i$ we would like this row to be in partial state $P_1$ at time

$$\tau_Q := i \Delta T + j \Delta t + T_P.$$ 

It follows that we would like our construction to be $Q$ synchronized at time $\tau_Q - \Delta T^*$. In particular, for each vertical session $V_{ik} \in V_Q$ we want

$$\tau_{ik} \equiv \tau_Q - \Delta T^* \pmod{N_{\text{sup}}\Delta T}. \quad (33)$$

Say the $C_{\text{prof}}$ is synchronized if (33) holds for each replenishing queue $Q$ and each vertical session $V_{ik} \in V_Q$.

Note that a well-formed connectivity profile $C_{\text{prof}}$ is completely determined by $N_{\text{sup}}$ and the set $C_{\text{prof}}^*$ of triples $(k, l, j)$ such that $(0, k, l, j) \in C_{\text{prof}}$. Let us call this set the reduced connectivity profile.

For a given reduced connectivity profile $C_{\text{prof}}^*$ let us define

$$N_{\text{sup}}(C_{\text{prof}}^*) := \max\{l \in \mathbb{N} : (k, l, j) \in C_{\text{prof}}^*\}$$

$$N_{\text{cols}}(C_{\text{prof}}^*) := \max\{k \in \mathbb{N} : (k, l, j) \in C_{\text{prof}}^*\}$$

to be, respectively, the maximal super-row that super-row 0 is connected to, and the maximal non-blank column. Then we can choose any $N_{\text{sup}} > N_{\text{sup}}(C_{\text{prof}}^*)$ and reverse-engineer the corresponding connectivity profile $C_{\text{prof}}$. Note that if it is well-formed then we satisfy (23) if and only if

$$N_{\text{cols}} \geq N_{\text{cols}}(C_{\text{prof}}^*) + L_1 + 1. \quad (34)$$

Clearly, being well-formed and being synchronized are properties of $C_{\text{prof}}^*$. More precisely, for a given $C_{\text{prof}}^*$ the connectivity profile is well-formed (resp. synchronized) for some $N_{\text{sup}} > N_{\text{sup}}(C_{\text{prof}}^*)$ then it is well-formed (resp. synchronized) for all such $N_{\text{sup}}$; in this case say that $C_{\text{prof}}^*$ is well-formed (resp. synchronized). It turns out that these two properties are all we need from $C_{\text{prof}}^*$.

**Lemma IV.8** If $\Delta T = 2\Delta T^*$, where $\Delta T^*$ is given by (25), then there exists a reduced connectivity profile $C_{\text{prof}}^*$ which is well-formed and synchronized.

**Proof:** Recall that within a given super-row, rows are numbered from 0 to $N_{\text{rows}} - 1$. Let us write row numbers as $ir + j$, where $j \in [r]$. Note that $C_{\text{prof}}^*$ is synchronized if and only if for each triple $(k, l, ir + j) \in C_{\text{prof}}^*$ we have

$$k T_P + \Delta T^* = l \Delta T + (ir + j) \Delta t + T_P. \quad (35)$$

Indeed, this matches (23) since $\tau_{ik} = k T_P$ and the right-hand side of (35) is $\tau_Q$, where $Q$ is the replenishing server in row $ir + j$ of super-row $l$. Let $L = L_1 + L_2$. Then by equations (19) and (25) we can rewrite (35) as follows:

$$kr = (2l - 1)Lr + (2l + i)r - (2l - 1 - j). \quad (36)$$

Equivalently, for some $\alpha \in \mathbb{N}$ we have

$$\begin{cases} 2l = \alpha r + j + 1 \\ k = L(\alpha r + j) + \alpha (r - 1) + (i + j + 1). \end{cases} \quad (37)$$

Note that in the above equation we want $l$ and $k$ to be integer; this is why in (19) we defined $r$ to be integer. Also, note that the first equation in (37) cannot hold for odd $j$ if $r$ is even; this is why in (19) we defined $r$ to be odd.

Let $A_j$ be the set of the first $N_{\text{vert}}$ integers that are at least 3 and have the same parity as $1 + j$. Let us define $C_{\text{prof}}^*$ as the collection of all triples $(k, l, ir + j)$ such that $\alpha \in A_j$, $i \in [L_1]$, $j \in [r]$, and $(k, l)$ are given by (37).

Since $r$ is odd, such $C_{\text{prof}}^*$ is synchronized. Let us check that $C_{\text{prof}}$ is well-formed. We need to check the four properties in Definition IV.7. Properties (b) and (d) are easy: the former holds since the $\alpha$’s are at least 3, and the latter holds since we reverse-engineer $C_{\text{prof}}$ from $C_{\text{prof}}^*$.

To prove property (a), let us consider equation (35). By (25) we have $\Delta T > N_{\text{rows}} \Delta t$, so for a given choice of $k$ there can be at most one value $l$ such that (35) holds, hence at most one triple $(k, l, ir + j) \in C_{\text{prof}}$. Therefore property (a) holds.

It remains to check property (c), i.e. that for each replenishing server $Q$ we have $|V_Q| = N_{\text{vert}}$. Indeed, say server $Q$ lies in row $ir + j$ of super-row $l$; for simplicity let us assume $l > N_{\text{sup}}(C_{\text{prof}}^*)$. Then the induced connectivity profile $C_{\text{prof}}$ contains a tuple $(n, k, n', ir + j)$ if and only if for some $\alpha \in A_j$ the pair $(l, k)$ satisfies (37) where the super-row number $l$ is

$$l = (n' - n) \pmod{N_{\text{sup}}}.$$ 

Claim follows since there are exactly $N_{\text{vert}}$ such pairs. $\square$

**F. Full construction: safe states**

Consider a given vertical session $V_{ik}$. Suppose super-row $i$ is in partial state $S(t_0)$ at time $\tau_{i0} + t_0$. As far as we are concerned, Lemma IV.5 specifies the behavior of session $V_{ik}$ between times $\tau_{ik}$ and $\tau_{ik} + \Delta T^*$. Here we investigate what happens with $V_{ik}$ during the rest of the time interval $[t_0; t_*]$, and also when super-row $i$ is empty.

Suppose a vertical session $V$ comes through a server $Q$. Let us say that at a given time server $Q$ is open for session $V$ if $Q$ has at least $\varepsilon_V$ units of capacity available for $V$. Suppose a given super-row contains session $V$. Let us say that at a given time this super-row is open for session $V$ if each server in this super-row is open for $V$.

Say a vertical session $V$ is safe at a given time if it enters the corresponding replenishing server $Q_V$ at rate at most $\varepsilon_V$. Say a replenishing server $Q$ is safe at a given time if it is empty of vertical fluid, and all vertical sessions $V \in V_Q$ are safe. Say a super-row is safe if each replenishing server in this super-row is safe. Intuitively, if vertical sessions, replenishing
servers and super-rows are safe then they do not cause any perturbations in our construction.

Let $\tau_{ik}':=\tau_{ik}+(N_{\text{rows}}-1)\Delta t$.

**Lemma IV.9** Consider a vertical session $V_{ik}$. Suppose for some $t \in [t_0; t_*]$ at time $\tau_{i0}+t$ super-row $i$ is in partial state $S(t)$ and session $V_{ik}$ is max-stable. Then

(a) session $V_{ik}$ is safe during the time interval(s)

$$[\tau_{i0}+t; \tau_{i0}+t_*] \setminus [\tau_{ik}'; \tau_{ik}+\Delta T^*],$$

(b) super-row $i$ is open for $V_{ik}$ during the time interval(s)

$$[\tau_{i0}+t; \tau_{i0}+t_*] \setminus [\tau_{ik}'; \tau_{ik}],$$

(c) if at time $\tau_{i0}+t_*$ super-row $i$ is safe and session $V_{ik}$ becomes unsafe at a later time $t'$, then super-row $i$ must have become unsafe during time interval $(\tau_{i0}+t_*; t')$.

**Proof:** The proof of parts (ab) is in-lined in the proof of Lemma [IV.8]. For part (c), note that at time $t_*$ both super-row $i$ and session $V_{ik}$ are safe, and by (24) there is no build-up of horizontal fluid in columns $k' \leq k$ of super-row $i$. Session $V_{ik}$ can become unsafe only (a positive time after) some build-up of horizontal fluid appears in some row $R$ in column $k$. The latter can happen only if at some earlier time, in row $R$ the horizontal session is injecting at the positive rate, then becomes blocked by vertical fluid in its replenishing server $Q$. The latter can happen only after server $Q$ becomes unsafe. □

Say a super-row is *empty* at a given time if all rows in this super-row are empty. We will also need a version of Lemma [IV.9] where initially super-row $i$ is empty, as opposed to being in one of the partial states $S(t)$.

**Lemma IV.10** Consider a vertical session $V_{ik}$. Suppose at some time $t$ super-row $i$ is safe and empty, and session $V_{ik}$ is safe. If session $V_{ik}$ becomes unsafe at a later time $t'$, then super-row $i$ must have become unsafe in time interval $(t; t')$.

The proof follows that of Lemma [IV.9] (c). The way we are going to apply Lemma [IV.9] (c) and Lemma [IV.10] it is crucial that super-row $i$ becomes unsafe strictly before time $t'$.

**G. Full construction: the high-level layout**

In this subsection we complete the specification of our construction. We choose parameters $(\varepsilon_h, \delta_h, \varepsilon_v, \delta_v, \mu_D, N_{\text{rows}})$ and positive integers $(L_1, L_2)$ from Lemma [IV.5]. We set $\Delta T = 2\Delta T^*$, where $\Delta T^*$ is defined by (12). We choose $C_{\text{prof}}$ from Lemma [IV.8]. We define $N_{\text{cobs}}$ to be the smallest integer such that

$$\Delta T \text{ divides } N_{\text{cobs}}\,T_P.$$  

Such $N_{\text{cobs}}$ exists because $L_1\Delta T$ is divisible by $T_P$ by (25). We need (38) for technical convenience, so that in later proofs we could consider moments of time when every super-row is either empty or in some well-defined partial state $S(t)$.

We choose $N_{\text{sup}} > N_{\text{sup}}(C_{\text{prof}})$ large enough so that all sessions are in the happy state long enough to build up the injection rate to the maximal value. Specifically, we let

$$N_{\text{sup}} = N_{\text{cobs}}\,T_P / \Delta T + n + 1,$$

where $n$ is the smallest integer such that if a horizontal session and a vertical session and a simple session are in a happy state from time 0 onward, then by time $n\Delta T$ they will be injecting at their respective maximal rates. This completes the construction.

Let us check that $N_{\text{sup}} \geq N_{\text{sup}}(C_{\text{prof}})$, as required by definition of $C_{\text{prof}}$. Indeed, let $N_{\text{sup}}(C_{\text{prof}}) = l$ so that $(k_1, l, \cdot, \cdot) \in C_{\text{prof}}$ for some $k$. Since the connectivity profile is synchronized, (35) holds. It follows that

$$N_{\text{sup}} - 1 \geq kT_P/\Delta T \geq l - 1,$

claim proved.

**H. Full construction: the high-level dynamics**

In this subsection we put all pieces together and show that starting from a suitable initial state our construction exhibits oscillations, with period $T_{\text{big}} = N_{\text{sup}}\Delta T$. To distinguish it from the 'local' period $T_P$, let us call $T_{\text{big}}$ the big period.

For $n \in N$, define $i_n = n \,(\text{mod } N_{\text{sup}})$. Recall the conventions (21). We want our system to have the following behavior:

For each $n \in N$, at time $n\Delta T + k_0T_P$

super-row $i_n$ is in partial state $S_{k_0}$. (39)

Moreover, we want super-row $i_n$ to get to partial state $S_{k_0}$ as a result of an $(i, t)$-replenishing process, for the appropriately chosen time $t$.

If (39) holds, then by Claim [IV.4] it is the case that at each time $t \in [n\Delta T + [k_0T_P, t_*]$ super-row $i_n$ is in state $S(t)$, so that $S(t_*) \in S_{\text{empty}}$. Moreover, we want this super-row to stay empty (almost) till time $n\Delta T + T_{\text{big}}$.

We want to prove oscillations via an inductive argument. Generally, for some time $t_0$ we want to define some state $H_n$ such that at time $t_0 + n\Delta T$ we want our construction to be in state $H_n$; for a given $n$ this will be our inductive hypothesis. The sequence $\{H_n : n \in N\}$ should have period $T_{\text{big}}$. We need to prove the inductive step from $n = m$ to $n = m + 1$. Then we get oscillations if at time $t_0$ we initialize our construction in state $H_0$.

With the above plan in mind, we have three difficulties to overcome. First, to define any reasonable $H_n$ we need to define states of super-rows, not partial states. For this we need to specify exactly the behavior of the vertical sessions so that this behavior is consistent across different super-rows; it is somewhat non-trivial to specify this behavior in a brute-force way. Second, we know how to argue about partial states $S(t)$, so in $H_n$ we would like each super-row to be in one of these partial states. For example, if (39) holds for $n = 0$, then the earliest time super-row 0 is in one of these partial states (namely, $H_{k_0}$) is time $k_0T_P$. Third, for technical convenience in a given inductive step we want to worry only about a single replenishing process, namely the one for super-row $i_n$. In particular, we would like it to be the case that at time $t_0 + n\Delta T$ the corresponding replenishing process for super-row $i_n + 1$ has not yet started. These considerations motivate the choice of $t_0 := k_0T_P$ and $\Delta T := 2\Delta T^*$.

For a given super-row, we are going to define a process $\{S^*(t) : t \geq t_0\}$ such that state $S^*(t)$ induces partial state
\(S(t)\) for any time \(t \in [t_0, t_*]\). We want this process to be consistent with the behavior of our construction if \(m^m_i\) holds for all \(n \in N\). To achieve this, we will modify a super-row so that it becomes a (suitable) stand-alone system, and in this system we will define a process by specifying the (suitable) initial state at time \(t_0\) and letting the system run. We let \(S^i(t)\) be the state of this system at time \(t\).

Let us consider a super-row \(S\) as a stand-alone system that includes all vertical sessions coming through the columns of \(S\), and all decelerating servers corresponding to the replenishing servers located in \(S\). Note that system \(S\) does not include the vertical sessions that go to these replenishing servers. Recall that in the full construction for each vertical session \(V\) in \(S\) there would be a decelerating server \(D\) (in a different super-row) which session \(V\) goes to. Essentially this server determines for how long session \(V\) remains blocked. Since here we are trying to emulate the behavior of a super-row in the full construction, let us add such servers artificially. Specifically, let us assume that for each vertical session \(V\) in \(S\) there is a distinct new quasi-decelerating server \(D_V\) with service rate \(\mu_D/N_{\text{vert}}\), such that session \(V\) leaves \(S\), then goes to \(D_V\), and then goes to its sink. Then \(S\) is a fully specified dynamical system.

Let us define the initial state \(S^0\) for system \(S\) such that it is in partial state \(S^0_{i_0}\), all replenishing servers are empty, and all vertical sessions in \(S\) are in the max-stable state. (It follows that in state \(S^0\) all servers in \(S\) are empty of vertical fluid.) Suppose we initialize system \(S\) at time \(t_0\) in state \(S^0\), and let it run according to its control mechanism. Define \(S^\Delta(t)\) to be the state of this system at time \(t\). Note that at time \(t_0\) system \(S\) is in partial state \(S^0_{i_0}\), so by Claim \(IV.4\) at any time \(t \in [t_0, t_*]\) it is in partial state \(S(t)\), as required.

Let us go back to our construction and ask how can we make sure that if we start a given super-row in state \(S^0\), it will indeed follow the process \(S^\Delta(t)\). Essentially, it will happen as long as the super-row is safe.

**Claim IV.11** Suppose super-row \(i\) is in state \(S^i(t)\) at some time \(t \geq t_0\), and it is safe between time \(t\) and time \(t' > t\). Then at time \(t'\) it is in state \(S^i(t')\).

Now we can formulate our inductive hypothesis, i.e. specify what is state \(H_n\). We define it by specifying separately the state of each super-row: \(H_n\) is the state of our construction such that each super-row \(i\) is in state \(S^i(t_0 + i_n \Delta T)\). Note that the sequence \(\left\{H_n : n \in \mathbb{N}\right\}\) has period \(T_{\text{big}}\), as required.

Finally, we are ready to state and prove the theorem that our construction exhibits oscillations.

**Theorem IV.12** Suppose we initialize our construction at time \(t_0\) in state \(H_0\). Then at time \(t_0 + T_{\text{big}}\) it is in the same state.

We formulate the proof as the next subsection.

**I. Full construction: proof of the main theorem**

It suffices to prove the inductive step. Let us assume that for some \(n \in \mathbb{N}\), at time \(t_1 := t_0 + n \Delta T\) our construction is in state \(H_n\), i.e. each super-row \(i\) is in state \(S^i(t_0 + i_n \Delta T)\). We need to prove that at time \(t_1 + \Delta T\) it is in state \(H_{n+1}\). Equivalently, we need to prove that super-row \(n + 1\) is in state \(S^*_{n+1}\), and any other super-row \(i\) is in state \(S^i(t_0 + i_{n+1} \Delta T + \Delta T)\). In short, we will get the former via replenishing processes (one for each row in super-row \(n + 1\)), and the latter via Claim \(IV.11\).

For simplicity let us assume that \(N_{\sup} < n < N_{\sup}\). Then at time \(t_1\) each super-row \(i \leq n\) is in state \(S^i(t_1 - i \Delta T)\), and super-row \(n + 1\) is in state \(S^*_{n+1}(t_0 + (N_{\sup} - 1) \Delta T)\). Note that by the choice of parameter \(N_{\sup}\) at time \(t_1\) all rows in super-row \(n + 1\) are max-stable.

Let us consider row \(R\) of super-row \(n + 1\). Let \(Q\) and \(D\) be the corresponding replenishing and decelerating servers, and let \(t_2 := \tau_Q - \Delta T^\ast\). Note that \(t_2 > t_1\). Indeed, by (25)

\[
\Delta T^\ast \geq (L_1 + 1) T_p = t_0 \\
\Delta T^\ast \geq n \Delta T + T_p - T_p = (n \Delta T + t_0) + (\Delta T^\ast - t_0) + T_p \\
\geq t_1 + T_p.
\]

**Claim IV.13** At time \(t_2\) our construction is \(Q\)-synchronized.

**Proof:** We need to check the conditions in the definition (24). Indeed, by the induction hypothesis at time \(t_1\) row \(R\) is max-stable, each vertical session \(V_{ik} \in V_Q\) is max-stable, and the corresponding super-row \(i \leq n\) is in partial state \(S(t_1 - i \Delta T)\). Therefore by Claim \(IV.11\) at time \(t_2\) it is in partial state \(S(t_2 - i \Delta T)\). Note that \(t_2 = \tau_{ik}\) since the connectivity profile \(C_{\text{prod}}\) is synchronized. It follows that at time \(t_2\) super-row \(i\) is in partial state \(S(\tau_{ik})\).

By Lemma \(IV.2\) a) during time interval \([t_1, t_2]\) session \(V_{ik}\) stays safe. Since this happens for all sessions \(V_{ik} \in V_Q\), during this time interval servers \(Q\) and \(D\) are empty, so row \(R\) stays max-stable. Moreover, by Lemma \(IV.2\) b) between times \(t_1\) and \(t_2\) super-row \(i\) stays open for \(V_{ik}\), so all servers on the flow path of \(V_{ik}\) are open for \(V_{ik}\), so it stays max-stable. We have checked all conditions in definition (24).

By Lemma \(IV.3\) it follows that at time \(\tau Q\) row \(R\) is in partial state \(P(T_p)\), each session \(V_{ik} \in V_Q\) is happy, and moreover row \(R\) is open between times \(t_2\) and \(\tau_Q\). We use this to prove the following two claims.

**Claim IV.14** At time \(t_1 + \Delta T\) replenishing server \(Q\) and decelerator server \(D\) are empty, and row \(R\) is in partial state \(P(t_0 - j \Delta T)\), where \(j\) is the row number of \(R\).

**Proof:** At time \(\tau Q\) all vertical sessions \(V_{ik} \in V_Q\) are happy. It follows that they are safe and servers \(Q\) and \(D\) are empty. By Lemma \(IV.9\) these sessions are safe during time interval \([\tau Q; t_1 + \Delta T]\). Therefore at time \(t_1 + \Delta T\) servers \(Q\) and \(D\) are empty.

Since at time \(\tau Q\) row \(R\) is in partial state \(P(T_p)\), by Claim \(IV.2\) c) at time \(t_1 + \Delta T\) this row is in partial state \(P(t)\), where \(t = T_p + t_1 + \Delta T - \tau Q = t_0 - j \Delta T\).

**Claim IV.15** Row \(R\) is open between times \(t_1\) and \(t + \Delta T\).
Proof: We already proved that row $R$ stays max-stable, hence open, between times $t_1$ and $t_2$. By Lemma IV.5, it is open between times $t_2$ and $\tau_Q$. Finally, at any time $t \in [\tau_Q, t_1 + \Delta T]$ this row is in partial state $P(t')$, $t' \leq t_0$, and therefore it is open by (2).

Since Claim IV.14 holds for all rows in super-row $n + 1$, at time $t + \Delta T$ this super-row is in partial state $S_{n_0}$. We need some more work to go from statements about partial states of super-rows to statements about their states.

Claim IV.16 Let $V$ be a vertical session $V$ in super-row $n + 1$. Then between time $t_1$ and $t_1 + \Delta T$ (a) this super-row is open for $V$, and (b) session $V$ is safe.

Proof: Part (a) follows by applying Claim IV.15 to every row in super-row $n + 1$. Part (b) follows from part (a) since by induction hypothesis at time $t_1$ session $V$ is max-stable.

Say a replenishing server is essential if it lies in super-row $n + 1$. Say a vertical session $V$ is essential if $V \in V_Q$ for some essential replenishing server $Q$. We show that non-essential vertical sessions and replenishing servers are safe as far as we are concerned.

Claim IV.17 For each non-essential vertical session $V_{ik}$, intervals $[t_1; t_1 + \Delta T]$ and $[\tau_{ik}; \tau_{ik} + \Delta T^*]$ are disjoint.

Proof: Suppose $V_{ik} \in V_Q$ for some replenishing server $Q$ in row $j$ of super-row $l \neq n + 1$. If $l \leq n$ then
\[
\tau_{ik} + \Delta T^* = l \Delta T + j \Delta t < n \Delta T + t_0 = t_1.
\]
If $l \leq n + 2$ then $\tau_{ik} > (n + 2) \Delta T - \Delta T^* > t_1 + \Delta T$.

Claim IV.18 Between time $t_1$ and $t_1 + \Delta T$, all non-essential vertical sessions and replenishing servers are safe.

Proof: By induction hypothesis this condition holds at time $t_1$. Suppose it fails at time $t \in (t_1, t_1 + \Delta T)$, and let us assume this is the first time it fails in this time interval.

If a non-essential replenishing server $Q$ becomes unsafe at time $t$, then at some time $t' \in (t_1, t)$ some vertical session $V \in V_Q$ must have become unsafe. Therefore without loss of generality a non-essential vertical session $V_{ik}$ becomes unsafe at time $t \in (t_1, t_1 + \Delta T)$. Let us argue to the contradiction.

By Claim IV.16(b) session $V_{ik}$ cannot belong to super-row $n + 1$. By induction hypothesis and (38), at time $t_1$ either super-row $i$ is empty or it is in a well-defined partial state $S_{i}(t_1 - \tau_{i0})$. In the first case by Lemma IV.10 super-row $i$ becomes unsafe at some time $t' \in (t_1, t)$, contradiction. So we are in the second case. If $t - \tau_{i0} > t_1$ then by Lemma IV.9(c) super-row $i$ becomes unsafe at some time $t' \in (t_1, t)$, contradiction. Else by Lemma IV.9(a) we must have $t \in [\tau_{ik}; \tau_{ik} + \Delta T^*]$, which contradicts Claim IV.17.

Claim IV.19 At time $t + \Delta T$ super-row $n + 1$ is in state $S_{n_0}$.

Proof: We know that at time $t + \Delta T$ this super-row is in partial state $S_{n_0}$, and (by Claims IV.14) all replenishing and decelerating servers in this super-row are empty.

It remains to prove that at this time each vertical session $V$ in this super-row is max-stable. This is the case because it is max-stable at time $t_1$ by the induction hypothesis, and during time interval $[t_1; t_1 + \Delta T]$ all servers on the flow path of $V$ are open for $V$. Namely, during this time interval the corresponding replenishing and decelerating servers are open for $V$ by Claim IV.18 and super-row $n + 1$ is open for $V$ by Claim IV.16(a).

For each super-row $i \neq n + 1$, note that by Claim IV.16(b) and Claim IV.18 it is safe between time $t_1$ and $t_1 + \Delta T$. At time $t_1$ it is in state $S^*(t_0 + i_{n-1} \Delta T)$ by the induction hypothesis, so by Claim IV.11 at time $t_1 + \Delta T$ it is in state $S^*(t_0 + i_{n-1} \Delta T + \Delta T)$ as required.

This completes the proof of Theorem IV.12 and therefore the proof of Theorem IV.1.
We will transform our construction so that all sessions have damping parameter $\delta^*$. (The same transformation also works if we replace $\delta^*$ by any given smaller value.)

Let us focus on one type of sessions. Let $(\varepsilon, \delta)$ be its flow control parameters. If $\delta = \delta^*$ then we do not need to do anything. Else, we will replace each session of this type with several parallel sessions with damping parameter $\delta^*$ and maximal injection rates $(\varepsilon_1, \varepsilon_1, \ldots, \varepsilon_k)$ that sum up to $\varepsilon$, where the rates $\varepsilon_i$ are chosen so that

$$f_\delta(\varepsilon, t) = \sum_{i=1}^k f_\varepsilon(\varepsilon_i, t) \quad \text{for all times } t \geq 0. \quad (42)$$

Assume such rates exist; we will prove it later. Then if these $k$ new sessions all become blocked when they are sending at their respective maximal injection rate, they decrease their sending rate exactly as the original session did. It is crucial that in our construction each session becomes blocked only when it is at the maximal injection rate.

For the modified construction, we choose $N_{\text{sup}}$, the number of super-rows, exactly as before, except now we want it to be large enough so that all new sessions have time to build up their injection rate to the maximum values. Then the modified construction works exactly as the original one.

It remains to prove that for any pair $(\varepsilon, \delta)$, there exist rates $(\varepsilon_1, \varepsilon_1, \ldots, \varepsilon_k)$ that sum up to $\varepsilon$ and satisfy (42). Indeed, whenever the rates are non-negative and sum up to $\varepsilon$, we have

$$\sum_{i=1}^k f_\varepsilon(\varepsilon_i, t) = \varepsilon - f(t) \sum_{i=1}^k h(\varepsilon_i, \delta^*).$$

If $\varepsilon_i = \varepsilon/k$ for each $i$, then

$$\sum_{i=1}^k h(\varepsilon_i, \delta^*) = k h(\varepsilon, \delta^*) = \varepsilon h(\varepsilon, \delta^*)/\varepsilon_1,$$

which is greater than $h(\varepsilon, \delta)$ if $k$ is large enough. Let us choose the smallest such $k$; note that $k \neq 1$ since $\delta^* < \delta$.

Now let us perturb the rates so that the above sum is exactly equal to $h(\varepsilon, \delta)$. With this goal in mind, let us define a one-dimensional function

$$H(\varepsilon_1) := h(\varepsilon_1, \delta^*) + \sum_{i=2}^k h(\varepsilon - \varepsilon_1/k, \delta^*) - \varepsilon h(\varepsilon, \delta^*)/\varepsilon_1,$$

Then $H(\varepsilon_1)$ is continuous on $[0, \varepsilon]$, and $H(\varepsilon/k) > h(\varepsilon, \delta)$. Moreover,

$$H(0) \geq \sum_{i=1}^{k-1} h(\varepsilon/k, \delta^*),$$

which is less than or equal to $h(\varepsilon, \delta)$ by the choice of $k$. Therefore there exists $\varepsilon_1 \in [0, \varepsilon/k]$ such that $H(\varepsilon_1) = h(\varepsilon, \delta)$, as required. For this value of $\varepsilon_1$ we define the other rates accordingly: $\varepsilon_i = (\varepsilon - \varepsilon_1)/(k - 1)$ for each $i \geq 2$. This completes the proof of the theorem. \hfill $\square$

VI. FURTHER RESEARCH: THE ALL-FIFO SETTING

We conjecture that Theorem \[11\] extends to the case when all servers must be FIFO. We have a promising preliminary result in this direction, namely an all-FIFO version of the basic gadget in which server $Q_2$ is strictly FIFO and does not give session $H$ priority over session $V$.

We consider the basic gadget from Section \[11\] with one modification that all four servers are now FIFO. Refer to Table \[11\] for the list of all relevant parameters. As before, we will have vertical session(s) going through server $Q_2$. Since now horizontal fluid does mix with vertical fluid, we have to consider the vertical session(s) explicitly inside the gadget.

We will allow $L$ vertical sessions coming through server $Q_2$ (in the full construction, parameter $L$ should define $N_{\text{vert}}$ and $N_{\text{rows}}$). We denote these sessions by $V_i, 1 \leq i \leq L$. We do not want to characterize the arrival pattern of sessions $V_i$ at this point. Instead, we shall simply place a bound on the arrivals. In particular, we let $f_i(t)$ represent the rate at which fluid arrives at server $Q_2$ on session $V_i$ at time $t$, and let $f(t) = \sum f_i(t)$. We shall assume that

$$\mu_2 < \mu_3 + \varepsilon_2^2 \quad \text{(43)}$$

$$f(t) \leq \varepsilon^2 \quad \text{for all times } t \quad \text{(44)}$$

In general, the vertical sessions are supposed to enter server $Q_2$ right after they leave from a similar server from some other gadget. For this reason we need to assume that the rates $f_i$ may depend on $\mu_2$. To this extent, we will assume the following:

the integral $\int_0^T f(t) \, dt$ is continuous in $\mu_2$ \[45\] for any fixed time $T$.

Lemma VI.1 Consider the basic gadget under all assumptions in Lemma \[11\] but define all servers to be FIFO. Then:

(a) There exist parameters $(\varepsilon_2, \delta_2, \mu_3, K, K')$ and times $(T_1, T_3, P)$ such that given \[43, 44\] the basic gadget functions as shown in Table \[11\].

(b) Part (a) holds for any given $K \geq 20$, in which case we can choose $\varepsilon_2 \in (5/3K'; 20/K')$ and set $\mu_3 = 2\varepsilon_2/5$.

(c) Moreover, for any $K \geq 100$ and the corresponding choice of parameters $(\varepsilon_2, \delta_2, K')$, and any rates $f_i(t)$ satisfying \[44, 45\] there exists $\mu_2 \in [\varepsilon_2/60, \varepsilon_2/5]$ satisfying \[43\] such that the maximal height of server $Q_2$ is $P/[5P]$.

Note that in part (a) one choice of parameters $(\varepsilon_2, \delta_2, \mu_3, K, K')$ works for any parameter $\mu_2$ satisfying \[43\], and any arrival rates $f_i(t)$ satisfying \[44\]. The idea is that first we choose the above five parameters using part (a), and then fine-tune $\mu_2$ using (c). Part (b) could be useful to make sure that $\varepsilon_2$ is as small as desired.

Part (c) is important because in the all-FIFO setting the time separation between two consecutive rows should be equal to the maximal height of server $Q_2$: making this height equal to $P/b$, $b \in N$ makes row $i + b$ synchronized with row $i$.

Proof: The proof of parts (ab) follows that of Lemma \[11\] almost word-by-word, with two modifications:

- In the second paragraph of Section \[11\] we need to note that the session $H$ traffic mixes with the “vertical” traffic in session $Q_2$. However, by \[44\] session $H$ fluid leaves server $Q_2$ at rate at least $1 - \mu_2 - \varepsilon_2^2$. Since this is larger than $1 - \mu_3$, we start to get a buildup in server $Q_3$.
- For part (a) we need to observe that times $T_1$, $T_3$ and $T_5$ depend only on the parameters $(K, \varepsilon_s, \delta_b)$ and are independent of $\mu_2$ and the flow rates $f_i$. To see this for time $T_1$, note that for any time $t \leq T_1$, height $h_1(t)$ is given by

$$h_1(t) = 1 + \int_0^t (d_s(t) - 1) \, dt,$$

where $d_s(t)$ is uniquely determined by system (14), and therefore by parameters $(K, \varepsilon_s, \delta_b)$.

We need more work to prove part (c). Recall that we assume $K \geq 100$; accordingly, by part (b) we can choose $\varepsilon_s \leq 1/60$. We'll state explicitly where we need this condition.

We vary $\mu_2$ in the interval $[\varepsilon_s/60; \varepsilon_s/5]$, keeping all other parameters fixed; note that any such $\mu_2$ satisfies $4/15$ as long as $\varepsilon_s \leq 1/5$, so all of the previous analysis applies.

The maximal height of server $Q_2$ becomes a function of $\mu_2$; we denote this height by $h_2(\mu_2)$. Intuitively, this is a continuous function. Let us quickly verify that this is indeed so. Recall that $T_1$ is fixed. For any time $t \in [0; T_1]$, we have

$$h_2(t) = \frac{1}{1 - \mu_2} \int_0^t (\mu_2 + f(t)) \, dt,$$

where $f(t) = \sum_{i=1}^P f_i(t)$. Since $h_2(\mu_2) = h_2(T_1)$, it is a continuous function of $\mu_2$ as long as we assume $45$.

We claim that $1/\varepsilon_s \leq T_1 \leq 3/\varepsilon_s$. Indeed, at any time $t \in [0; T_1]$, server $Q_1$ drains at rate $1 - d_1(t)$, which by (16) lies between $\frac{1}{\mu_2}(1 + \frac{1}{\mu_2} - \mu_3)$ and $\varepsilon_s/(1 + \varepsilon_s - \mu_3)$. Hence the time $T_1$ at which server $Q_1$ drains satisfies,

$$\frac{1}{\varepsilon_s} \leq \frac{1 + \frac{1}{\mu_2} - \mu_3}{\varepsilon_s} \leq T_1 \leq \frac{1}{\mu_2} \frac{1 + \frac{1}{\mu_2} - \mu_3}{\varepsilon_s} \leq \frac{3}{\varepsilon_s},$$

claim proved.

By (47), at any time $t \in [0; T_1]$ height $h_2(t)$ increases at rate that lies between $\mu_2/(1 - \mu_2)$ and $(\mu_2 + \varepsilon_s^2)/(1 - \mu_2)$. The latter number is at most $2\mu_2$ whenever $\varepsilon_s \leq 1/60$ and $\mu_2 \in [\varepsilon_s/60; \varepsilon_s/5]$, which matches our initial assumptions; the proof is a simple but tedious calculation which we omit. Therefore

$$\mu_2/\varepsilon_s \leq H_2(\mu_2) \leq 6\mu_2/\varepsilon_s,$$

so, in particular, $H_2(\varepsilon_s/60) \leq 1/10$, and $H_2(\varepsilon_s/5) \geq 1/5$.

Finally, let us choose an integer $b = \lceil 5P \rceil$. Since $P \geq 1$, we have $P/b \in [1/10; 1/5]$. Since $H_2(\mu_2)$ is continuous, there exists a $\mu_2 \in [\varepsilon_s/60; \varepsilon_s/5]$ such that $H_2(\mu_2) = P/b$.

Unfortunately, we are unable to construct an example where the replenishing servers are strictly FIFO. The main difficulty is that for a FIFO replenishing queue, when the vertical sessions are creating a build up of fluid a small amount of session $H$ fluid will be served. Since we would like the session $H$ fluid to start traversing the row only after all the vertical session fluid has left the replenishing queue, this small amount of session $H$ fluid is served too early.

By choosing parameters appropriately we can ensure that the amount of session $H$ fluid that is served early is arbitrarily small. However, as long as the amount is nonzero it will affect the operation of all the subsequent gadgets in the row. We conjecture that the effect will be minimal and will not affect the final result. We believe that an interesting open problem is to prove this conjecture.

VII. CONCLUSIONS

In this paper we have shown that natural flow control schemes can create oscillations when interacting with networks of queues. These oscillations occur because the queueing dynamics affect the rate at which data passes through a server and so the arrival rate of a session at a server can be different than the external session injection rate. In particular, our example is always feasible in the sense that the total injection rate of all sessions passing through a server is never bigger than the service rate of that server. This provides a contrast with previous work showing oscillations of TCP.

A number of open questions remain. The first set of questions concerns elaborating our construction. As already mentioned, we would like to extend our example to the case in which all servers are FIFO. Also, we would like to extend Theorem 1 (‘same damping parameter for all sessions’) to a wider class of flow control functions, most notably to multiplicative decrease. Lastly, we would like to accommodate TCP Vegas-type schemes (see [23], [3], [7]) where the flow rate is increased/decreased at a rate of $1/(\Gamma_i(t))^2$ depending on whether or not there is congestion on the path of flow $i$. Here $\Gamma_i(t)$ is the end-to-end-delay on the path of flow $i$.

On the other hand, we wonder what would be a good way to break our construction. In particular, it would be nice to prove a complimentary convergence result for our model under minor restrictions on the initial state, the way we fine-tune parameters, or the underlying network topology.

Another area for future study is reducing the complexity of our example. We note that much of this complexity arises from the fact that we wish to recreate our initial conditions exactly so that we can create a infinite sequence of oscillations. We are curious if there is a much simpler example in which it is possible to create a large finite number of oscillations. In particular, we wonder if it might be possible to create such examples on more natural topologies.

Another open question relates to the “stability” of our oscillating example. That is, if we slightly perturb the state of the system, will the oscillations persist? We would also like to investigate if our example can be applied to networks utilizing different scheduling disciplines. For example, we wonder whether TCP-like flow control can exhibit oscillations when interacting with a network of servers that schedule flows according to the Generalized processor sharing scheme (see [24], [25], [26]). Another feature of our example is that it utilizes large buffers. Whenever a queue builds up the participating sessions recognize congestion and then reduce their injection rates. It would be interesting to know if our example can be adapted to a small buffer scenario in which fluid is dropped when congestion occurs.

Lastly, we would like to investigate to best way to prevent oscillations via joint flow control and network scheduling. A promising approach in this direction is the Greedy primal-dual algorithm of Stolyar [21].
REFERENCES

[1] F. Kelly, “Charging and rate control for elastic traffic,” European Trans. on Telecomm., vol. 8, pp. 33–37, 1997.
[2] F. Kelly, A. Maulloo, and D. Tan, “Rate control for communication networks: shadow prices, proportional fairness and stability,” Journal of the Operations Research Society, vol. 49, no. 3, pp. 257–282, 1998.
[3] S. Low, L. Peterson, and L. Wang, “Understanding Vegas: a duality model,” Journal of the ACM, vol. 49, no. 2, pp. 207–235, 2002.
[4] C. Holotl, V. Misra, D. Towsley, and W. Gong, “A control theoretic analysis of RED,” in Proceedings of IEEE INFOCOM ’01, 2001.
[5] S. Rybko and A. Stolyar, “Ergodicity of stochastic processes describing RED,” in Proceedings of IEEE INFOCOM ’99, 1999.
[6] H. Choe and S. Low, “Stabilized Vegas,” in Proceedings of IEEE INFOCOM ’00, 2000.
[7] S. Low, L. Peterson, and L. Wang, “Understanding Vegas: aduality analysis of RED,” in Proceedings of IEEE INFOCOM ’00, 2000.
[8] M. Andrews, B. Awerbuch, Anna Fernández, J. Kleinberg, T. Leighton, and M. Andrews, “Instability of FIFO in session-oriented networks,” in Proceedings of IEEE INFOCOM ’00, 2000.
[9] M. Bramson, “Convergence to equilibria for fluid modelsof FIFO networks with flow schedulers,” in Proceedings of IEEE INFOCOM ’00, 2000.
[10] A. Demers, S. Keshav, and S. Shenker, “Analysis and simulation of RED,” in Proceedings of IEEE INFOCOM ’00, 2000.
[11] Alexander L. Stolyar, “Maximizing queueing network utility subject to constraints: A generalized primal-dual algorithm,” Queueing Systems, vol. 4, no. 4, pp. 401–457, 2005.
[12] Marco Ajmone Marsan, Mirko Franceschinis, Emilio Leonardi, and ISDN Systems, “Congestion avoidance on a global internet,” IEEE/ACM Transactions on Networking, December 2004, vol. 12, pp. 1131–1143, Preliminary version in IEEE INFOCOM, 2003.
[13] Alexander L. Stolyar, “Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm.” Queueing Systems, vol. 50, no. 4, pp. 401–457, 2005.
[14] Marius Bramson, “Instability of FIFO in the permanent sessions model of a network,” Queueing Systems, vol. 36, no. 2, pp. 187–207, 2000.
[15] F. Kelly, A. Maulloo, and D. Tan, “Rate control for communication networks: shadow prices, proportional fairness and stability,” Journal of the Operations Research Society, vol. 49, no. 3, pp. 257–282, 1998.
[16] M. Andrews, B. Awerbuch, Anna Fernández, J. Kleinberg, T. Leighton, and M. Andrews, “Instability of FIFO in session-oriented networks,” in Proceedings of IEEE INFOCOM ’00, 2000.
[17] M. Bramson, “Convergence to equilibria for fluid modelsof FIFO networks with flow schedulers,” in Proceedings of IEEE INFOCOM ’00, 2000.
[18] A. Demers, S. Keshav, and S. Shenker, “Analysis and simulation of RED,” in Proceedings of IEEE INFOCOM ’00, 2000.
[19] A. K. Parekh and R. G. Gallager, “A generalized processor sharing approach to flow control in integrated services networks: The single-node case,” IEEE/ACM Transactions on Networking, vol. 1, no. 3, pp. 341 – 357, 1993.
[20] A. K. Parekh and R. G. Gallager, “A generalized processor sharing approach to flow control in integrated services networks: The multiple-node case,” IEEE/ACM Transactions on Networking, vol. 2, no. 2, pp. 137 – 150, 1994.

APPENDIX I
CONTINUITY ISSUES

Let us return to Section III-C and consider the integral in (15) as a function of \( \varepsilon \) that we denote \( F(\varepsilon) \). We show that it is continuous in \( \varepsilon \). We will use the notation from Section III.

Claim I.1 \( F(\varepsilon) \) is continuous on \([\varepsilon_1, \varepsilon_2]\).

We will need some basic facts from two-variable calculus.

Fact I.2 For sets \( S, S' \subset \mathbb{R}^2 \), consider functions \( f : S \to \mathbb{R} \) and \( g : S' \to \mathbb{R} \). Let \( h(x, y) = f(x, g(x, y)) \) and assume that this function is well-defined on \( S' \), i.e. that \((x, g(x, y)) \in S \) for any \((x, y) \in S'\).

If \( g \) is continuous at some point \((x, y) \in S'\), and \( f \) is continuous at the corresponding point \((x, g(x, y)) \in S \), then the composition \( h \) is continuous at \((x, y) \).

Fact I.3 For a closed rectangle \( S \subset \mathbb{R}^2 \), let \( x(\varepsilon, t) \) be a function \( S \to \mathbb{R} \) which is continuous on \( S \) and increasing in \( t \). Then writing \( x_\varepsilon(t) := x(\varepsilon, t) \), the function \( f(\varepsilon, x) := x_\varepsilon^{-1}(x) \) is continuous on the set \( \{ (\varepsilon, x) : x = x(\varepsilon, t) \text{ and } (\varepsilon, t) \in S \} \).

Fact I.4 Consider a compact set \( S \subset \mathbb{R}^2 \) such that for any point \((x, y) \in S\), the segment \([x \in \varepsilon, x \in \varepsilon] \) lies in \( S \). Suppose a function \( f : S \to \mathbb{R} \) is continuous on \( S \). Then the integral \( g(x, y) := \int_0^y f(x, y) \, dy \) is continuous on \( S \).

Equipped with these facts, let us prove Claim I.1. Recall that \( \varepsilon(t) \), the injection rate into session \( H \) at time \( t \), is really a function of two variables, time \( t \) and parameter \( \varepsilon \). For clarity, let us denote \( f(\varepsilon, t) := \varepsilon(t) \). This function is defined on the closed rectangle \( R = [\varepsilon_1; \varepsilon_2] \times [0; 2K] \); it is continuous by (A5).

By (III2) and plugging in \( \mu_2 = 2\varepsilon/5 \) we have \[ x_\varepsilon(t) := t + h_\varepsilon(t) = t(1 - 2\varepsilon/5) + \int_0^t f(\varepsilon, t) \, dt. \]

This is a function of two variables \((\varepsilon, t)\); by Fact I.4 it is continuous on \( R \).

Consider the set \( R^* := \{ (\varepsilon, x) : x = x_\varepsilon(t) \text{ and } (\varepsilon, t) \in S \} = \{ (\varepsilon, x) : \varepsilon \in [\varepsilon_1; \varepsilon_2] \text{ and } x \in [0; x_\varepsilon(2K)] \} \).

Let us treat \( x_\varepsilon(t) \) as a (parameterized) function of one variable, \( t \), and consider its inverse \( x_\varepsilon^{-1}(x) \). This inverse is well-defined for all \((\varepsilon, x) \in R^*\). We can view it as a function of these two variables; by Fact I.3 this function is continuous on \( R^* \). By Fact I.2 it follows that \( g(\varepsilon, x) := f(\varepsilon, x_\varepsilon^{-1}(x)) \) is continuous on \( R^* \), too.

Let us recall that the rate \( d_4(t) \) depends on time \( t \) and parameter \( \varepsilon \); accordingly, let us write \( d_4(t) = d_4(\varepsilon, t) \). Then we can re-write (14) as

\[ d_4(\varepsilon, x) = \frac{1 - 2\varepsilon/5}{g(\varepsilon, x) + 1 - 2\varepsilon/5}. \]
By Fact $\ref{fact2}$ this is also a continuous function on $R^*$. By
Fact $\ref{fact4}$ it follows that the function
\[ F(\varepsilon, x) := \int_0^1 d_4(\varepsilon, x) \, dx \]
is continuous on $R^*$. Finally, recall that
\[ F(\varepsilon) = F(\varepsilon, T(\varepsilon)), \text{ where } T(\varepsilon) = K/(1 - 2 \varepsilon/5). \]
By Fact $\ref{fact2}$ $F(\varepsilon)$ is continuous on $[\varepsilon_1, \varepsilon_2]$, as required. This
completes the proof of Claim $\ref{claim1}$. 

20