A Theory of Packet Flows Based on Law-of-Mass-Action Scheduling

Thomas Meyer
Department of Mathematics and Computer Science
University of Basel, Switzerland
Email: th.meyer@unibas.th

Christian Tschudin
Department of Mathematics and Computer Science
University of Basel, Switzerland
Email: christian.tschudin@unibas.th

***Abstract*—Designing dynamically robust protocols is not a simple task with current classic work-conserving scheduling, where packets are sent out as soon as processing and transmission capacity is available. In this paper, we show that deviating from this fundamental queuing assumption leads to much more controllable and analyzable forms of protocols. At the core of our work is a queue-scheduling discipline based on the chemical "Law of Mass Action" (LoMA) that serves a queue with a rate proportional to its fill level.**

In this paper we introduce our LoMA-scheduling approach and provide a solid mathematical framework adopted from chemistry that simplifies the analysis of the corresponding queueing networks, including the prediction of the underlying protocols' dynamics. We demonstrate the elegance of our model by implementing and analyzing a TCP-compatible "chemical" congestion control algorithm C3A with only a few interacting queues (another novelty of our approach). We also show the application of our theory to gossip protocols, explain an effective implementation of the scheduler and discuss possibilities of how to integrate mass-action scheduling into traditional networking environments.

I. Introduction

A major problem in engineering communication protocols is to control the network-level dynamics. Early protocol designs in the 1960s and 70s aimed at logical functionality only. Dynamics started to play a major role at MAC (Ethernet) and transport level (TCP) in the 1980s: queuing theory was adopted and congestion control was developed in parallel. In the recent two decades, the network community has recognized that predictable dynamic behavior is a requirement for obtaining robust protocols. However it is still hard to bridge the gap between the micro-behavior of arbitrary protocol logic and its macroscopical dynamics. Even though theories such as the *Network Calculus* [18] help to reason about the *worst-case* behavior of large work-conserving queueing networks, we currently have no means to characterize the *average* dynamic trajectory of a system, which is essential in proving its default behavior. Ideally, an engineer would co-design functional and dynamic aspects of a new protocol, and conveniently, the protocol's dynamics would be expressed in terms of average flow properties and their robustness, rather that low-level queue parameters. In such an environment, the microscopic packet-delay details would depend on – and would be derived from – the macroscopic flow behavior, rather than being the starting point for an analysis.

To couple the micro- with the macro-level behavior and to be able to analyze a system's equilibrium is the core property of our "Law of Mass Action" (LoMA) approach. LoMA scheduling belongs to the class of non-work-conserving scheduling disciplines (which are rarely considered in computer networks: non-work-conserving queuing adds delays to packets, which for efficiency reasons seems to be a thing more to avoid rather than to embrace). Hence, before explaining LoMA in more detail, we briefly review research and applications that considered non-work-conserving scheduling.

Related Work: Several queueing disciplines introduce a small delay to packet streams with the aim of shaping traffic and guaranteeing fairness among competing flows. Delayed Frame Queueing (DFQ) [19] for ATM networks, for example, sticks to work-conserving forwarding in the intermediate nodes but allows for non-work-conserving transmission at the network's edge. This leads to guaranteed upper bounds of delay and jitter for virtual circuits (VC), without relying on individual VC-queueing. Traffic shaping is also a hot topic for jitter-sensitive multi-media streams: The authors of [16] propose to equip Ethernet hubs with a Constant Delay Queueing (CDQ) policy. CDQ afflicts each packet of the constant-rate IPTV stream with a constant delay and forwards other best-effort packets during this waiting time.

Another application area is layer 2, where wireless MAC protocols such as 802.11 make use of guard time intervals (because one cannot sense the wire as in 802.3), including random wait times, which leads to deliberate delays, too. It has also been shown that throughput-optimal allocation of the wireless channel capacity can be achieved by a work-conserving approach, in which a maximum-weight-scheduling discipline takes the queue's fill levels into account (e.g. in Maximum Weight Scheduling [31], Maximum Weight Matching Scheduling (MWMS) or Queue Proportional Scheduling (QPS) [2], [29]). What is noteworthy here, is the prioritization of queues in proportion to their fill levels, which is similar to the behavior of our LoMA scheduling discipline.

The common goal of these application fields is to make packet flows "more fluid", more predictable, and to provide strict or stochastic delay guarantees. However, to our knowledge, the corresponding non-work-conserving and fill-level-proportional scheduling disciplines are rarely discussed in the literature: At best they are mentioned as an exotic variant

without being analyzed thoroughly (e.g. in [32, Chap. 13]).

Law-of-Mass-Action scheduling in a nutshell: LoMA scheduling specifies in a rigid way the delay with which a packet is afflicted, such that some given macroscopic *rate* will result. More specifically, the LoMA states an inverse dependency between a queue's fill level and its inter-packet service time: the more packets are in the queue, the quicker the system processes them. As a consequence, the *waiting time* of a packet is constant, regardless of the queue's fill level. By designing local or remote packet processing loops and by coupling them, complex dynamic behavior can be built, for example rate limiters, rate differentiators, integrators and much more. As we will show, this is already sufficient to design and formally analyze complex flow behavior such as gossip protocols, or the well-known additive-increase / multiplicative-decrease congestion control as well as refinements thereof.

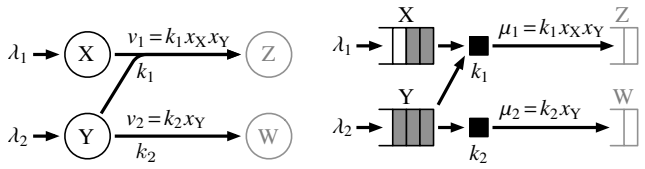
Contributions: The *first contribution* of this paper is a technical as well as abstract framework based on interacting packet queues that permits to construct complex packet flows whose dynamic properties can be formally analyzed and that helps protocol designers to study the dynamic's robustness early in the design phase. This result rests on the import of concepts from chemistry, among them the mass-action scheduling regime, which is non-work-conserving (Sect. II and III). Our *second contribution* is to show that LoMA scheduling can be done efficiently, which means that one can "run" corresponding protocols in an efficient way, turning our conceptual framework into an executable model that can operate at wire speed (Sect. IV). The *third contribution* lies in reformulating two well-known protocols (one from the world of gossip protocols, the other being the famous TCP protocol for which we provide a "chemical" congestion control algorithm that is TCP fair) and to analyze their well-known dynamics in new and elegant forms (Sect. V). *Fourth*, we discuss how (classical) packet flows can be steered by our chemically-inspired approach using a chemically modeled control plane and how compatible our packet flows are with the existing packet-forwarding infrastructure (Sect. VI).

II. Mass-Action Scheduling

In this section, we describe the law of mass action from chemistry and compare it to the well-known M/M/1 queueing model. We start by mapping concepts from chemistry ("molecular reactions") to packet queue events and then give a first example of how to use a LoMA-scheduled queueing network to solve a classical distributed consensus problem.

A. Molecules as Packets, Reactions as Queues

Figure 1(a) shows a system of two chemical reactions r_1 and r_2 having two reactants X and Y and two products Z and W. More precisely, r_1 is a bimolecular reaction $\mathrm{X} + \mathrm{Y} \xrightarrow{k_1} \mathrm{Z}$ while the competing reaction r_2 simply turns a Y-molecule into a W-molecule. The rate at which these reactions happen is controlled by the respective coefficients k_1 and k_2 called rate constants.



(a) Chemical reaction model (b) Corresponding FIFO queues

Fig. 1. Bimolecular reaction: extracts an instance of two chemical species (=packet queues), X and Y. At the same time, queue Y is drained by a second server (=reaction).

We map above configuration to a system of two queues X and Y (Fig. 1(b)) and make an analogy between chemistry and networking by treating packets as molecules. A molecular species can be viewed as a buffer that temporarily stores molecule instances until they are being consumed by an egress reaction. In terms of networking, we have two packet queues X and Y served by two servers. The upper server is special in that it extracts a packet from two queues at the same time. Table I summarizes how we translate chemical ideas to queueing theory.

Chemical metaphor…	**…for**	**Symbol**
Molecule instance	Packet	
Molecular species	Queue	$\mathrm{X}, \mathrm{Y}, \mathrm{Z}, \ldots$
Concentration	Fill level	$x_\mathrm{X}, x_\mathrm{Y}, x_\mathrm{Z}, \ldots$
Reaction	Server	$r_1, r_2, r_3, \ldots$
Reaction rate	Service rate	$v_1, v_2, v_3, \ldots$
Inflow	Fill rate	$\lambda_1, \lambda_2, \lambda_3, \ldots$

TABLE I
Chemical queueing metaphor

B. The Law of Mass Action (LoMA)

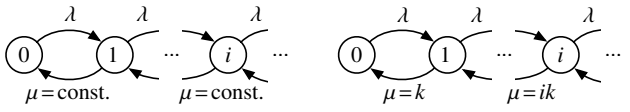
The "Law of Mass Action" in chemistry stipulates that the reaction rate depends proportionally to the abundance of the involved reactants.[1] According to this law, the reaction rate of r_1 in Fig. 1 is $v_1 = k_1 x_\mathrm{X} x_\mathrm{Y}$, where x_X and x_Y are the concentrations of the molecules X and Y, respectively. The rate $v_2 = k_2 x_\mathrm{Y}$ of the decay reaction r_2 depends only on the concentration of Y, i.e., it is linear w.r.t. x_Y.

We map this behavior to our packet queues in Fig. 1. The arrival rates λ_1 and λ_2 express how fast packets are put into the queues, while the concentrations x_X and x_Y become the fill level of the (potentially infinite) queues X and Y.

C. Properties of LoMA-Scheduled Queues

Like the classical M/M/1 queue, our LoMA-scheduled queue is modeled as a Markov birth-death chain (see Fig. 2). But unlike the classical queue, the mean death transition rate μ is now proportional to the fill level, meaning that waiting packets build up a "pressure" and accelerate the service rate.

[1]This principle is rooted in chemical kinetics, which explains the random movement of molecules by Brownian motion. The more molecules a fixed volume contains, the more frequently they collide and react in turn. At the macroscopic level, this manifests in the law of mass action [1], [33].



(a) M/M/1 queue (b) LoMA-scheduled queue

Fig. 2. Markov birth-death chain

For the simple case of a queue serviced by a single uni-molecular "reaction", the following properties hold (see also Tab. II): The expected fill level, $\hat{x}$, is proportional to the packet arrival rate λ. This leads to a constant waiting time T in steady state, meaning that the *latency* of a packet is independent on the arrival rate and independent on the fill level. These two quantities are related by Little's Law, which still holds for the mass-action scheduling regime.

Property of Queue	**M/M/1**	**LoMA**
Expected fill level $\hat{x}$	$\lambda/(\mu-\lambda)$	λ/k
Expected waiting time T	$1/(\mu-\lambda)$	$1/k$

TABLE II
QUEUE PROPERTIES: Properties of the traditional M/M/1 queue compared to a LoMA-scheduled queue.

D. LoMA-Scheduled Queueing Networks

While a uni-molecular exhibits a constant waiting time for traversing packets, the timing can be shaped differently by introducing multi-molecular reactions. A server that serves two or more queues aggregates the consumed packets (see reaction r_1 in Fig. 1). Similar to a Petri-Net transition, such a server synchronizes multiple packet streams, as a packet in one queue can only be processed when there are packets in the other reactant queues. Such a bimolecular reaction acts like a "chemical transistor": it allows one packet flow to control another one.

Note that in our model, a queue may be drained by multiple servers: Reaction r_2 in Fig. 1 for example drains packets from queue Y independent of reaction r_1. This requires that the competing servers coordinate their action. As we will see in Sect. IV-A, the servers are communicating indirectly through the fill level of the queues.

These two novel elements, multi-molecular and parallel reactions – or in networking terms: synchronizing and parallel servers – are powerful extensions, and are required for building complex protocols by just combining queues.

E. Example: Disperser – *Consensus and Distributed Computation through Interacting Packet Flows*

As a first example, we present *Disperser* – a LoMA-scheduled queueing network based on uni-molecular reactions that computes the average of distributed values. It performs distributed computation by disseminating information to randomly selected neighbors, similar to gossip-based or epidemic protocols such as *Push-Sum* [17].

Figure 3 depicts *Disperser's* reaction network that consists of one packet queue per node. Each queue is drained in parallel by one separate server per network link. In chemical terminology, there is one reaction per node-link pair that consumes a local molecule and produces it in the corresponding neighbor node.

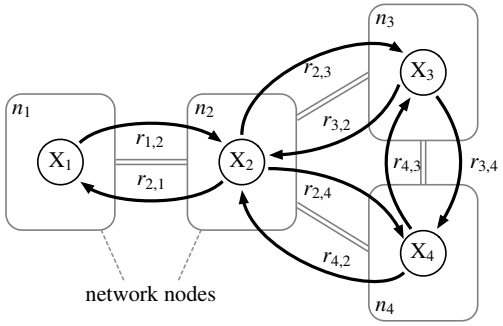


Fig. 3. *Disperser:* A distributed LoMA-scheduled queueing network that globally averages the fill levels of the queues.

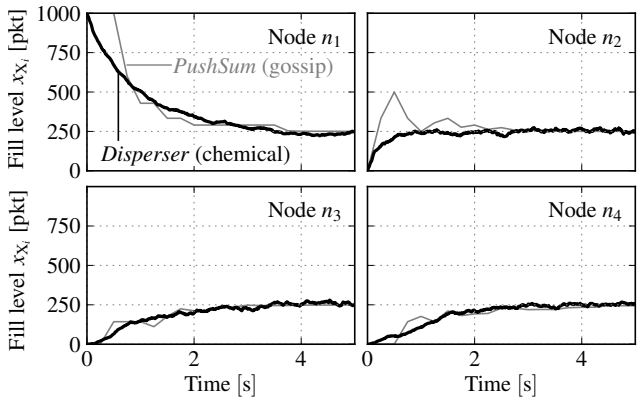


Fig. 4. OMNeT++ simulation of *Dipserser* and *Push-Sum:* The queueing network has a global attractor where the packets are distributed uniformly across the queues (here for the topology depicted in Fig. 3).

If we would run this queuing network with traditional work-conserving scheduling, molecules would accumulate at the node with the highest degree, as all other nodes are faster to shuffle packets to it. However, when queues are scheduled according to the law of mass action, the system has a global attractor that is independent of the topology. Figure 4 shows a typical simulation run, as well as a one-to-one comparison with a corresponding *Push-Sum* implementation. At equilibrium, each queue contains the average number of packets in the system. Thus our simple LoMA-scheduled queueing network disperses the packets uniformly across the network, hence its name.

Interestingly and typically for "chemical" algorithms, the computation is not carried out symbolically by explicitly calculating "the average". The result is rather emerging from the dynamic interaction of packet flows. Protocol states are represented by the multiplicity of packets in queues whereas information is conveyed to remote nodes via the packet *rate*. The law of mass action plays an important role by mapping fill levels to packet rates and vice-versa.

III. Formal Analysis

Chemists analyze chemical reaction networks and their robustness for over a century with remarkable progress in the past decades. In this section, we review some results and make them available to LoMA-scheduled queuing networks.

A. Microscopic Stochastic Analysis

The dynamic behavior of chemical reaction networks is described by the Chemical Master Equation (CME) [22], which can be derived directly from collision theory [7]. The master equation describes a continuous-time discrete-space Markov jump process, which is similar to that of classical queueing networks, but with fill-level proportional transition rates (see Fig. 2).

B. Macroscopic Deterministic Analysis

Like for classical queuing theory, a detailed stochastic treatment is only feasible for simple networks. Chemists rarely study the dynamics of reaction networks at the microscopic stochastic level. Instead, they approximate the average trajectory of the system by Ordinary Differential Equations (ODE) – one equation for each species. This also has a tradition in networking, where a fluid model is often built in order to analyze the dynamics of protocols and packet flows [20], [25], [30]. The problem of fluid network models is that they have to be extracted manually from the protocol's source code and that it is not always clear whether they accurately model the real behavior.

Fluid model: For chemical queueing networks the fluid model is accurate and easy to obtain: Gillespie showed that a simple ODE approximation accurately predicts the systemic trajectory of arbitrary chemical reaction networks, the dynamics of which are correctly described by the stochastic master equation [8]. A LoMA queueing network actually performs a distributed simulation of a chemical reaction system. Thus its ODE-model can be generated directly and automatically from the topology of the corresponding reaction network.

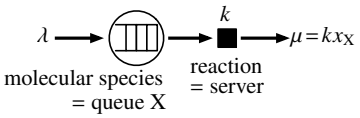


Fig. 5. LoMA-scheduled queue = unimolecular reaction

Let us analyze the fluid model of a single queue as depicted in Fig. 5. The fill level x_{X} of queue X is increased at rate λ and decreased at rate μ, which yields the differential equation

$$\dot{x}_{\mathrm{X}} = \overbrace{\lambda}^{\text{inflow}} - \overbrace{kx_{\mathrm{X}}}^{\text{outflow}} \tag{1}$$

Since the outflow rate is proportional to the fill level x_{X} there is always a stable fixed point: For $\dot{x}_{\mathrm{X}} \equiv 0$, we obtain a steady-state fill level of $\hat{x}_{\mathrm{X}} = \lambda/k$ (also compare to Tab. II).

Kirchhoff's current law: The LoMA scheduling leads to further observations on the macroscopic flow level that simplify the intuitive understanding of queuing networks by just looking at the corresponding reaction network graphs. One such observation is Kirchhoff's current law in electronic circuits, which also holds for chemical reaction networks at equilibrium [27]. It states that the total rate of packets flowing into a queue is equal to the total rate of packets drained from the queue. This law can be derived naturally from the queue's ODE and is very helpful in designing and analyzing chemical queueing networks with pen and paper.

Transient analysis: There are also well-known methods to analyze the transient behavior of chemical reaction networks that can be applied directly to LoMA-scheduled queueing networks: The Metabolic Control Analysis [3], [9], [10], [28] studies the sensitivity of states (fill levels) to internal and external perturbations. The transient behavior of fill levels and packet flows can also be analyzed by signal and control theory [11]. Perturbation analysis, for example, linearizes the ODE system around the fixed point and studies the system's stability or frequency response.

C. Formal Convergence Proof of Disperser

In this subsection, we use a perturbation analysis on the fluid model of our *Disperser* protocol (see Sect. II-E and Fig. 3) to prove that the system is robust in the sense that it always converges to a stable fixed point. We show that the system presents the expected result at this fixed point, i.e., that each queue contains the average number of packets in the network. This results is valid for arbitrary topologies if the network is symmetric and connected.

In general, such a convergence proof is carried out in three steps: (a) we write down the differential equations of all queues, (b) we find the global fixed point and show that the packets are uniformly distributed, and (c) we prove that this fixed point is asymptotically stable.

a) Fluid model: Each queue receives packets from its neighbors at a rate equal to the fill level the neighbor's queue. Concurrently, one reaction per neighbor drains the local queue. The differential equation describing the fill level x_i of queue X_i in node $i \in \mathcal{V}$ is

$$\dot{x}_i = \overbrace{\sum_{k \in \mathcal{N}_i} x_k}^{\text{inflow}} - \overbrace{\deg(i) x_i}^{\text{outflow}} \quad \forall i \in \mathcal{V} \tag{2}$$

where $\deg(i)$ is the degree[2] of node i, and where the set $\mathcal{N}_i$ contains the neighbors of node i.

b) Fixed point: We find the fixed point by assuming that all fill levels have settled, i.e. by setting all $\dot{x}_i \equiv 0$:

$$\hat{x}_i = \frac{\sum_{k \in \mathcal{N}_i} \hat{x}_k}{\deg(i)} \quad \forall i \in \mathcal{V} \tag{3}$$

That is, the equilibrium fill level of queue X_i is equal to the average fill level in node i's neighborhood. In a connected

[2]A node's in-degree is equal its out-degree in symmetric networks.

network this only holds iff all queues contain the same number of packets: $\hat{x} = \hat{x}_i$. Since all reactions conserve the total number of packets ($x_{\text{tot}} = \sum_{i\in\mathcal{V}} x_i = \text{const.}$) the fill level of all queues is equal to the average number of packets.

$$\hat{x} = \hat{x}_i = \frac{x_{\text{tot}}}{|\mathcal{V}|} \quad \forall i \in \mathcal{V} \tag{4}$$

c) Stability: This fixed point is asymptotically stable if the system returns back to it after a small perturbation. For the corresponding analysis we calculate the $|\mathcal{V}| \times |\mathcal{V}|$ Jacobian matrix of (2):

$$\mathbf{J} = \left[j_{i,k}\right] = \left[\frac{\partial \dot{x}_i}{\partial x_k}\right] = -\mathbf{L}(\mathcal{G}) \tag{5}$$

where $\mathbf{L}(\mathcal{G})$ is the Laplacian of the network graph

$$\mathbf{L}(\mathcal{G}) = \left[l_{i,k}\right] = \begin{cases} \deg(i) & \text{if } i = k; \\ -\operatorname{adj}(i,k) & \text{otherwise.} \end{cases} \tag{6}$$

The adjacency function $\operatorname{adj}(i,k)$ yields the value 1 if node i is connected to node k, or 0 otherwise. Since the Laplacian of any connected symmetric graph has positive real eigenvalues, the eigenvalues of (5) are negative and the fixed point in (4) is stable for arbitrary network topologies. □

Compared to the convergence proof of *Push-Sum*, our proof is considerably more compact. Convergence proofs are often simpler in a LoMA-scheduled queueing network because we can automatically generate a fluid model from the structure of the protocol's queueing network and apply a classical perturbation analysis.

IV. Execution Model

The chemical model introduced in the previous sections serves as an abstract model to design and analyze LoMA-scheduled queuing networks. In order to build real hard- and/or software that closely follows this abstract model, we have to address engineering concerns and implementation issues, such as how the proclaimed macroscopical LoMA rate can be achieved by a microscopic packet-per-packet scheduling algorithm. In Sect. IV-A, we first discuss the implementation of an efficient mass-action scheduler. Section IV-B then proposes a separation between traffic forwarding engine and "chemical" control layer that can automatically be compiled from the abstract model.[3]

A. An Efficient Mass-Action Scheduler

Let $\mathcal{S}_i$ be the set of all queues in network node i and $\mathcal{R}_i$ the set of all servers (=reactions) consuming packets from those queues. Each network node runs a scheduler that performs the following tasks: Compute the next occurrence time of each server $r \in \mathcal{R}_i$ according to the law of mass action, sort these events into a priority queue, wait until the first event occurs and execute it. The difficulty is to dynamically and efficiently re-schedule the reaction events if packets are added or drained from dependent queues. As first demonstrated in [6], an efficient mass-action scheduler implementation exists, which only requires $\mathcal{O}(\log|\mathcal{R}|)$ time to enqueue or dequeue packets. Here, we give an outline and a possible implementation of this scheduler.

Data structures: Figure 6 depicts the three data containers our implementation maintains. Each queue $s \in \mathcal{S}_i$ keeps track of its fill level and informs the dependent server. Each server $r \in \mathcal{R}_i$ then computes and stores the service rate v_r according to the law of mass action (7).

$$v_r = k_r \prod_{s\in\mathcal{S}_i} x_s^{\alpha_{s,r}} \tag{7}$$

where k_r is the reaction constant, and where $\alpha_{s,r} \in \{0,1\}$ is the stoichiometric coefficient, which denotes whether a packet in queue s is consumed by reaction r.

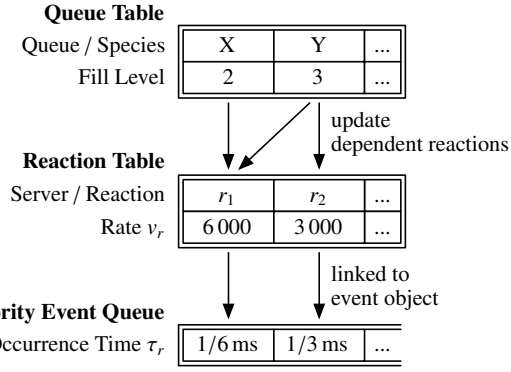


Fig. 6. Reaction Scheduler Implementation: This example corresponds to the reaction network and state depicted in Figure 1 for $k_1 = 10^3/(\text{pkt s})$ and $k_2 = 10^3/\text{s}$.

Generally, the interval at which the servers drains the queues is inversely proportional to this rate: $\tau_r = 1/v_r$. Each server initially registers an event for time τ_r with the local event scheduler, which is implemented as an indexed priority queue. This queue sorts the event, earliest time first. If the queue is implemented as a tree, this can be done in $\mathcal{O}(\log|\mathcal{R}|)$ time [6].

Dynamic scheduling: The scheduler has to reshuffle the events dynamically as packets enter or leave the queues. For example, if a server is forced to wait 1 ms but a packet arrives within this period, the service interval has to be shortened.

Let us illustrate the scheduling process based on the simple queueing network depicted in Fig. 1: Figure 6 shows the state of the scheduler at time $t_{\text{now}} = 0\,\text{s}$, assuming that there are two packets in queue X and three packets in queue Y, and that the reaction constants are $k_1 = 10^3/(\text{pkt s})$ and $k_2 = 10^3/\text{s}$, respectively. Consequently, the first server will serve its queues (before the second) at time $t = \frac{1}{6}\,\text{ms}$, as shown at the top of Fig. 7. The scheduler picks the first event from the priority queue and waits until its occurrence time is reached. It then triggers the corresponding server, which extracts packets from the reactant queue(s) and injects them to the product queue(s). Thereafter, the server recomputes its interval and reschedules its event.

[3]For space reason, we cannot show another execution method we implemented and tested [23], which reshapes the queueing network at run-time with an active networking approach.

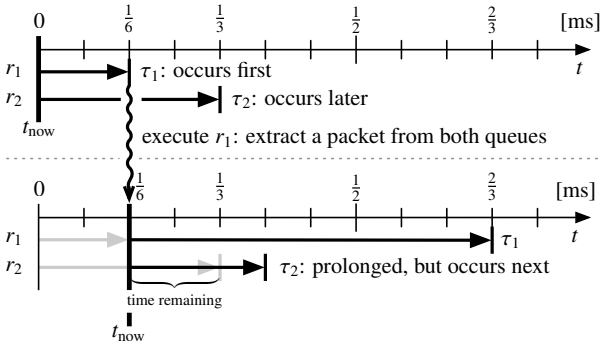


Fig. 7. Dynamic Scheduling: Situation before and after the occurrence of the first reaction. Time intervals of reactions are rescaled when the fill level of dependent queues changes.

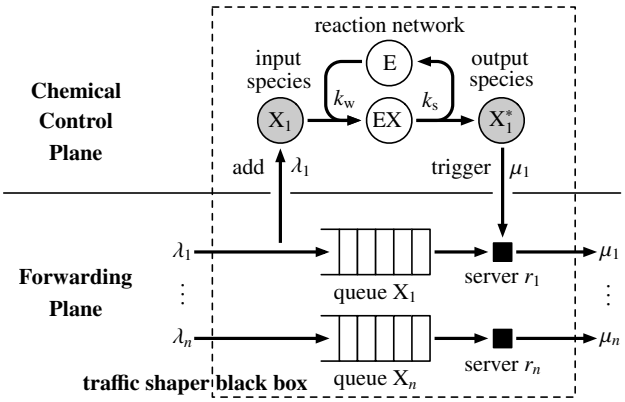


Fig. 8. Chemical flow management architecture: A packet flow is buffered in a classical FIFO queue. Its server is triggered by a chemical reaction network that links input to output.

In our case, reaction r_1 removes a packet from both queues X and Y. Since the occurrence time of reaction r_2 also depends on the fill level of queue Y, it is necessary to reschedule this reaction, too. Two packets are left in queue Y, so the new reaction rate of r_2 is $v_{r_2,new} = 2 \times 10^3\,(\mathrm{pkt/s})$. Because the occurrence time of r_2 has not elapsed yet, the scheduler has to rescale it according to the following equation.

$$\tau_{r,new} = \overbrace{\frac{v_{r,old}}{v_{r,new}}}^{\text{scaling}} \cdot \overbrace{\left(\tau_{r,old} - t_{\text{now}}\right)}^{\text{time remaining}} + t_{\text{now}} \tag{8}$$

The bottom part of Fig. 7 shows how the occurrence time of r_2 is prolonged a bit in turn.

The same figure also illustrates that our scheduling algorithm is not a priority scheduler: the reactions are interleaved according to their weight (=rate), meaning that the reaction with the highest rate is not always scheduled first. At the macroscopic flow level, this scheme leads to the correct expected packet flow rates according to the law of mass action.

Note that unlike for Earliest Deadline First (EDF) or similar timed scheduling algorithms, our approach does not require tagging each packet with a timestamp. This is because the service rate is proportional to the fill level of all dependent queues, which means that the superposition principle applies: we can treat all packets in a queue together and only schedule one event per reaction.

B. A Chemical Flow Control Plane

LoMA-scheduled queueing networks may be used to perform traffic shaping. In such a scenario, the queueing network can be modeled as black-box system with multiple input and output packet flows. The system applies a dynamic filter (in the form of a chemical reaction network) to the "rate signal" of the traversing packet flows and reshapes their traffic patterns.

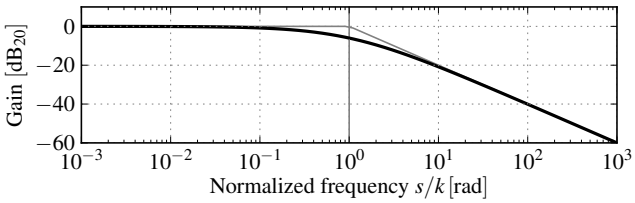
Instead of sending data packets through a complex queueing network in order to shape the packet flow, we envision an execution model that separates packet forwarding from "chemical" flow control as depicted in Fig. 8: An ingress packet flow i ($1 \leq i \leq n$) is sent to a separate FIFO queue that is drained by one server each. The servers have no predefined service rates; instead the rates are determined dynamically by the chemical control plane, where each queue X_i is represented by an input species X_i and its server by an output species X_i^*. Between these species the traffic engineer can span an arbitrary chemical reaction network, which may also be influenced by the activity of the other flows.

Whenever a data packet enters a queue, a molecule is injected to the corresponding input species in the control plane. The reaction network in the chemical control plane simulates the network with a mass-action scheduler and eventually generates an instance of the output species. This immediately triggers the corresponding server in the forwarding plane and causes the next packet to be dequeued.

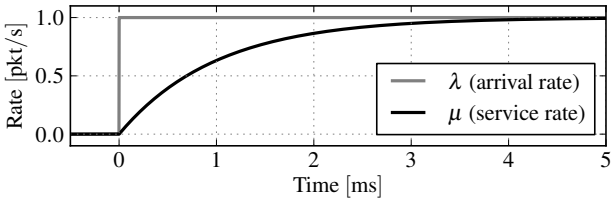
Instead of processing *packet rates*, the chemical control plane may alternatively operate on *byte rates*. In this case, an incoming packet generates a quantity of molecules that corresponds to its length in bytes, whereas the server has to collect an amount of triggers equal to the length of the front packet in the queue before dequeuing it.

Although the abstract molecules in the chemical control plane represent packets (or bytes), they have no shape, meaning that they do not carry payload data. Since all molecule instances of the same species are identical, a typical implementation just has to store an integer value per species to keep track of its multiplicity. It is possible to either compile an abstract chemical network directly into a executable program or to use the reaction network to parameterize a generic reaction network simulator. Such a direct relation between the execution model and the abstract model allows an early and thorough analysis of the behavior already at design time.

Traffic smoothing: In the following, we present two examples how a simple chemical reaction network shapes packet flows. A simple unimolecular reaction between the input and output species as depicted in Fig. 5, acts as a low-pass filter to a packet flow. This can be shown by converting the differential equation in (1) to the frequency domain. The transfer function below has low-pass characteristics with a cut-off frequency at the reaction constant k (see Fig. 9(a)).



(a) Transfer function



(b) Step response

Fig. 9. A LoMA-scheduled queue acts to a packet flow as a low-pass filter: Transfer function and step response of the reaction network in Fig. 5.

$$F(s) = \frac{\mu(s)}{\lambda(s)} = \frac{k}{s+k} \tag{9}$$

This means that bursts with high frequency components are smoothed. The step response of a LoMA-scheduled queue (depicted in Fig. 9(b)) illustrates that the service rate exponentially approximates the arrival rate.

Such a low-pass filter applied at the ingress point of the network leads to less chaotic behavior of traffic patterns in the network. However, this does not come for free, as the filter imposes the packets with a delay of $d = 1/k$ and requires memory to buffer the packets: The mean fill level grows proportionally with the delay and the flow rate: $\hat{x} = \lambda/k$.

Rate limiting: Unlike a traditional M/M/1 queue with a fixed service rate, a LoMA-scheduled queue exhibits an unbounded service rate, which follows the arrival rate. Nevertheless, our next showcase is a reaction *network* that is able to limit the rate of the packet flow.

Rate-limiting reactions are very common in biological systems in the form of enzymatic reactions, such as the one in the chemical control plane of Fig. 8. Intuitively this rate limiter works as follows: Substrate molecules X are generated with the arrival rate λ. Enzyme molecules can be thought of as tokens: they are either in free form (E) or bound as enzyme-substrate complex (EX). The more enzymes are bound, (i.e., the less free enzymes there are), the slower will the transmission rate μ grow for an increasing arrival rate λ. We can also consider E as counting semaphore for which a data packet X has to "wait" before resting in EX for a short time. The semaphore is "signaled" when the packet is eventually sent.

According to Kirchhoff's current rule, the influx and efflux of species EX are equal at equilibrium: $k_\mathrm{w} x_\mathrm{X} x_\mathrm{E} = k_\mathrm{s} x_\mathrm{EX}$. Since the total number of enzyme tokens is constant ($x_\mathrm{E} + x_\mathrm{EX} = e_0 = \text{const.}$) we obtain the *Michaelis-Menten equation* [24], which expresses the transmission rate μ with respect to the fill level x_X of queue X:

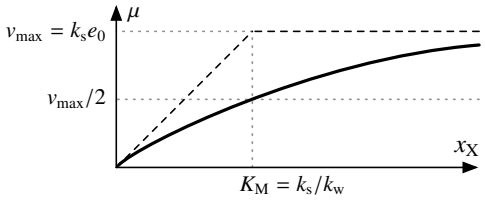


Fig. 10. Saturation curve of the enzymatic reaction network

$$\mu = v_\mathrm{max} \frac{x_\mathrm{X}}{K_\mathrm{M} + x_\mathrm{X}} \tag{10}$$

Figure 10 depicts the resulting hyperbolic saturation curve. The constant $K_\mathrm{M} = k_\mathrm{s}/k_\mathrm{w}$ specifies the fill level of X at which half of the maximal rate $v_\mathrm{max} = k_\mathrm{s} e_0$ is reached. For an empty queue ($x_\mathrm{X} \to 0$) the rate limiter behaves like a unimolecular reaction with rate $v_\mathrm{max}/K_\mathrm{M}$ whereas for a high fill level ($x_\mathrm{X} \to \infty$) the transmission rate converges to the limit v_max. Obviously, the system is only stable if the offered *load* $\rho = \lambda/v_\mathrm{max}$ is lower than 1. In other words, the steady-state fill level grows hyperbolically with the load and reaches infinity for $\rho \to 1$:

$$\hat{x}_\mathrm{X} = K_\mathrm{M} \frac{\rho}{1-\rho} \tag{11}$$

The mean latency of a packet through a chemical reaction network is generally calculated as the sum of the inverse egress reaction rates along a packet's path. In our enzymatic rate limiter, the average steady-state latency $T = T_\mathrm{X} + T_\mathrm{EX}$ of a packet through species X and EX is

$$T = \overbrace{K_\mathrm{M} \frac{1}{v_\mathrm{max} - \lambda}}^{T_\mathrm{X}} + \overbrace{\frac{1}{k_\mathrm{s}}}^{T_\mathrm{EX}} \tag{12}$$

In the limit $k_2 \to \infty$ and $K_M \to 1$, the enzymatic rate limiter approximates the M/M/1 queue ($T = 1/(v_\mathrm{max} - \lambda)$).

V. C3A — A Chemical Congestion Control Algorithm

So far, we studied single LoMA-scheduled queues and simple flow-filtering patterns. In this section, we demonstrate the straightforward combination of queues to "schedule" segments of a transport protocol with the aim of reducing congestion. Congestion control is a novel application of queuing theory, made possible by the law-of-mass-action discipline. To this end, we re-implement the additive increase / multiplicative decrease mechanism of TCP Reno's congestion avoidance mode, originally proposed by Van Jacobson [12]. It is not our aim to cover all features of TCP such as fast start, but to demonstrate how a flow of data packets is throttled by a simple queueing network such that the emitted data stream is fair to TCP streams.

Figure 11 shows the reaction network of our chemical congestion control algorithm C3A. Arriving data packets are placed into queue D. The transmission rate is controlled by the

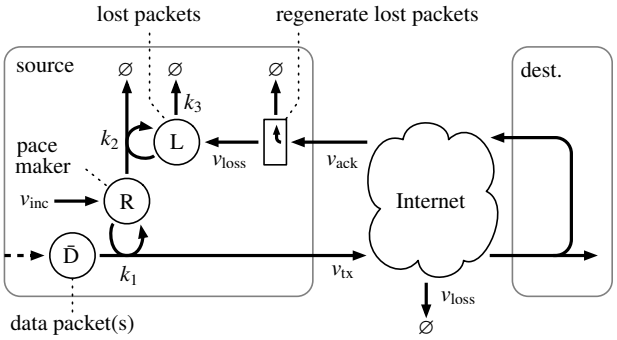


Fig. 11. Reaction network of the chemical congestion control algorithms C3A: The reaction network in (a) is reused in (b) (shaded in grey) and extended by new species and reactions (black) in order to sense and react to round-trip-time (RTT) variations.

quantity of pace maker molecules R: $v_{\text{tx}} = k_1 x_{\text{R}} x_{\text{D}}$, according to the law of mass action. The number of pace maker molecules is continuously increased at rate v_{inc}, which mimics the additive (here: linear) increase mechanism. Before being transmitted the packets are tagged with a continuously increasing sequence number. Based on a gap in the sequence number of received acknowledgments the source node is able to regenerate the lost packets and insert them into queue L. Such a lost packet catalyzes the destruction of pace maker molecules through reaction r_2, which leads to an exponential decay of R-molecules and a corresponding drop of the transmission rate. The duration of the decay however is limited as lost molecules are also decayed by reaction r_3.

A. The Reaction Graph as Design Instrument

A reaction graph (such as the one depicted in Fig. 11) is a very useful tool in the design phase of flow management algorithms. It conveys information in a much more intuitive way that an implementation of the same functionality in an imperative procedural language, such as C. For instance, the linear increase mechanism in our example is clearly visible as inflow of R-molecules. We can also easily recognize the feedback nature of congestion control when looking at the part of the network where the lost packets in L invoke the death of pace makers R. Similar to electronic circuits where currents control the flow of other currents in transistors, here, packet flows control the flow of other packets via bimolecular reactions.

B. TCP-Fairness of C3A

Analytical treatment: According to [4], [5] the transmission rate of TCP-fair packet streams should be proportional to $1/\sqrt{p_{\text{loss}}}$ where p_{loss} denotes the round trip packet loss probability between source and destination. With Kirchhoff's current law (see Sect. III) we can easily proof that our queueing network satisfies this property: At equilibrium, the fill level of the R- and L-queues do not change and their in- and out-flows are equal:

$$\overbrace{v_{\text{inc}}}^{\text{inflow of R}} = \overbrace{k_2 \hat{x}_{\text{R}} \hat{x}_{\text{L}}}^{\text{outflow of R}} \tag{13a}$$

$$\underbrace{\hat{v}_{\text{loss}}}_{\text{inflow of L}} = \underbrace{k_3 \hat{x}_{\text{L}}}_{\text{outflow of L}} \tag{13b}$$

According to the law of mass action, the transmission rate is $\hat{v}_{\text{tx}} = k_1 \hat{x}_{\text{R}} \hat{x}_{\text{D}}$; into this equation, we plug in $\hat{x}_{\text{R}}$ from (13a).

$$\hat{v}_{\text{tx}} = \frac{k_1 v_{\text{inc}} \hat{x}_{\text{D}}}{k_2 \hat{x}_{\text{L}}} \tag{14}$$

Next, we take $\hat{x}_{\text{L}}$ from (13b) and express the packet loss rate v_{loss} with respect to the loss probability: $\hat{v}_{\text{loss}} = \hat{v}_{\text{tx}} p_{\text{loss}}$.

$$\hat{v}_{\text{tx}} = \frac{k_1 k_3 v_{\text{inc}} \hat{x}_{\text{D}}}{k_2 \hat{v}_{\text{tx}} p_{\text{loss}}} \tag{15}$$

By multiplying both sides by $\hat{v}_{\text{tx}}$ and taking the square root we obtain a steady-state transmission rate of

$$\hat{v}_{\text{tx}} = \sqrt{\frac{k_1 k_3 r_{\text{inc}} \hat{x}_{\text{D}}}{k_2}} \cdot \frac{1}{\sqrt{p_{\text{loss}}}} \tag{16}$$

We determined optimal reaction constants by optimizing the behavior of the reaction network with respect to a short response time and a small overshoot. Were therefore linearized the ODEs of the system and determined the frequency response of the resulting LTI system. The obtained constants are $k_1 = 1/(\text{pkt}\,\text{s})$, $k_2 = 10^4/(\text{pkt}\,\text{s})$, $k_3 = 10^5/\text{s}$, and, as rate to increment the pace maker, $v_{\text{inc}} = 10^3\,\text{pkt/s}$.

OMNeT++ simulations: We empirically verified the theoretical dependence between packet loss and resulting transmission rate by OMNeT++ simulations. Figure 12 shows that the simulation results match the predicted transmission rate well for different packet loss probabilities.

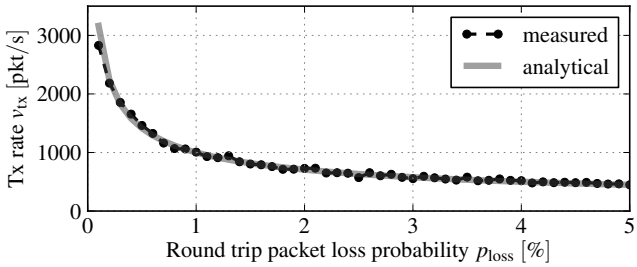


Fig. 12. TCP-Fairness of C3A: The transmission rate drops with a square root relation with respect to the packet loss probability. We determined the values analytically from (16) and complemented them with empirical OMNeT++ simulation results, averaged over a transmission time of 10 s.

Figure 13 depicts the simulation results for two data streams that are sent over the same bandwidth-limited link; the bandwidth is $b = 1\,\text{MB/s}$ and the delay $d = 5\,\text{ms}$. The two plots on the top display the saw-tooth pattern of the transmission rate of both source nodes, which is caused by the linear increase / exponential decrease behavior of C3A.

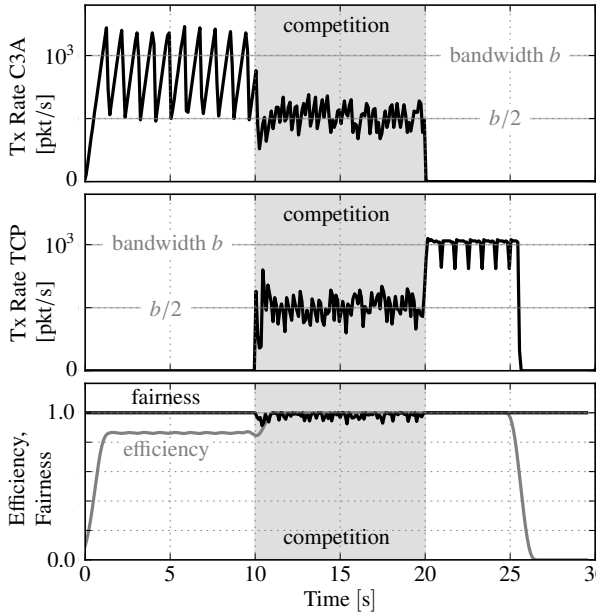


Fig. 13. Chemical congestion controlled packet stream (C3A) in competition to a TCP stream: Simulation in OMNeT++ with its native TCP Reno implementation: MSS $= 10^3$ B, delayed ACKs are disabled, Nagle's algorithm is enabled [26], selective ACKs are disabled [21], the advertised window is at the maximum of 64 kB. We measured the efficiency as the total transmission rate and the fairness by using Jain's fairness index [15].

C. Closing Remark

The aim of our "chemical" approach to congestion control is not to compete with existing advanced TCP implementations. We rather want to demonstrate the advantage of the mass-action-scheduling regime for the design and analysis of packet flow management algorithms.

VI. Discussion

Our account of LoMA-based packet flow management has to be completed in several ways with what we know today and what still has to be explored in future work.

Niche or Realm: We showed several spots in the upper layers of a networking stack where LoMA scheduling can be applied. First, the enzymatic rate limiter is a classical traffic-policing mechanism for the network layer. Second, the C3A algorithm could be turned into a full TCP-fair protocol in the transport layer. The vanilla place to experiment with LoMA scheduling in the Internet is of course the application layer.

For those layers, it might be worth exploring our approach further under the assumptions that widespread work-conserving scheduling of traditional MACs does work nicely enough, especially in an over-provisioned environment. However, to directly benefit from LoMA dynamics, it also seems interesting to investigate "chemical links" on which QoS stacks can be built. We mention content centric networking [13], [14] here, because we feel that this flow-aware approach, which bridges application level objects with forwarding level concerns, would naturally benefit from dynamics control built into its framework.

A possible use case of LoMA scheduling is delay-neutral, scalable forwarding in QoS networks. We pointed out in Tab. II that LoMA forwarding incurs a constant delay. One obvious application is in the field of QoS, where the chemical setting guarantees that the end-to-end delay is constant and the jitter is kept small, independent of the activity of other streams. Assume a sufficiently dimensioned node that keeps per-flow forwarding state, each flow being equipped with its own LoMA queue with low-pass behavior in order to keep the jitter small. If the individual flows are now multiplexed through another LoMA queue into trunks (in order to reduce forwarding state in the core), the low end-to-end jitter remains. That is, the forwarding node multiplexes parallel streams without any crosstalk by increasing the delay by a constant amount, determined solely by the reaction constant of the multiplexing reaction.

Demarcation: Formulated as a hypothesis, we think that *best-effort traffic* is intrinsically incompatible with LoMA scheduling (a statement that we would have to capture and examine formally). Strict packet flow prioritization, for example, is hardly achievable because the pressure generated by the growing fill level of a low-priority or best-effort queue would inevitably lead to its prioritization. This is both a limitation and a useful architectural insight: trying to embed best-effort in our framework is probably not worthwhile, and running LoMA flows over the current Internet will only exploit part of its potential. However, *parallel worlds* can be conceived where resource boundaries outside LoMA scheduling are engineered that assign left-over buffer and bandwidth resources to the best-effort packet class, as well as well-defined transit points to enter and leave the space of LoMA-managed packets.

Finally, one should not credit LoMA scheduling to lead to "better" flow behavior *per se*: *Robustness* and *fairness*, for example, still remain characteristics that have to be programmed (which is feasible in the LoMA context, as we showed with our C3A algorithm). However, we think that provable robustness is easier to obtain with LoMA-scheduled queueing networks than with work-conserving best-effort forwarding, because LoMA couples fill level and service rate, which inherently leads to equilibrium. Furthermore, the low-pass filter effect of LoMA-scheduled queues might be beneficial insofar as it mitigates the on-set impact of traffic bursts of end nodes to core nodes and servers under DoS attacks.

Implementation Issues and Future Work: The queueing model presented in this paper makes four assumptions that simplify its treatment and that we have to discuss here: First, we based our flow-level analysis on packet rates. But since the link capacity is physically given in terms of bytes per second, we usually want to consider the byte rate of packet flows instead. For the LoMA scheduler this is more tricky in variable size packet networks because the service time may change on-the-fly while a link is busy sending the previous packet. We sketched a possible solution to take the packet size into account in Sect. IV-B.

Second, real links impose a delay additional to the waiting time of a queue. In the fluid model, this would lead to a Delay Differential Equation (DDE) system, which is generally hard to

analyze. If we are only interested in the steady-state behavior of a queueing network (e.g. when analyzing the equilibrium fill-level distribution), a link with delay d can be modeled as an additional LoMA-scheduled queue with a reaction constant of $k = 1/d$. Such a virtual queue simulates the delay of the link but keeps the mathematical model free of delay terms.

Third, the message complexity of pure "chemical" protocols will be higher than traditionally. The *Disperser* protocol, for example, uses the packet rate to convey information; more messages are exchanged than in the comparable *Push-Sum* protocol. However, this is only an issue if the packets cross a real link between two queues. For local queueing networks such as the enzymatic rate limiter, packets are not sent remotely but only used to control other packet flows locally. On the other hand, TCP demonstrates that in order to perform rate control, frequent acknowledgment packets is the price to pay for having a reliable feedback signal from the network.

Fourth, we assumed that the queue capacity is infinite. In reality, all queues are bound and some drop behavior has to be specified. For the LoMA-scheduling discipline this means that the maximum service rate automatically has an upper bound, too. This makes sense, as a CPU must be able to execute the scheduled events in time. Thus in LoMA scheduling, memory limits are directly linked to the performance limits of the executing machinery. This allows for controlling one with the other, and makes it possible to dimension the system in advance.

VII. Conclusions

Computer networking is the art of combining logical functionality with dynamic behavior to achieve robust distributed services: On one hand, we have to logically organize distributed computation such that useful side effects like coherent routing state or accurate content copies emerge from the exchange of data packets despite the imponderability of communication links and failing nodes. On the other hand, we have to orchestrate delay by deferring packets; otherwise network dynamics will grow out of control and will make all logical attempts to provide functionality futile. Law-of-Mass-Action scheduling is a low-level fixture on packet delay that introduces useful restrictions over the current work-conserving scheduling scheme: It separates logical forwarding from packet timing and makes time control subject to a programmable environment. When this programming is done by weaving reaction networks, rich and yet analyzable dynamics can be engineered on the flow-level without the need to micro-manage the delays individually. We hope that our work contributes to turning delay management from a "black art" into an engineering discipline.

References

[1] H. I. Abrash. Studies concerning affinity. *Journal of Chemical Education*, 63:1044–1047, 1986. English translation of [33].

[2] A. Eryilmaz, R. Srikant, and J. R. Perkins. Throughput-optimal scheduling for broadcast channels. In *Proc. SPIE*, volume 4531, pages 70–78, 2001.

[3] D. A. Fell. *Understanding the Control of Metabolism*. Portland Press, London, 1997.

[4] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Trans. on Networking*, 7(4):458–472, 1999.

[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer. TCP friendly rate control (TFRC): Protocol specification. RFC 5348 (Proposed Standard), 2008.

[6] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of Physical Chemistry A*, 104(9):1876–1889, 2000.

[7] D. T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications*, 188(1–3):404–425, 1992.

[8] D. T. Gillespie. The chemical Langevin equation. *Journal of Chemical Physics*, 113(1), 2000.

[9] R. Heinrich and S. Schuster. *The Regulation of Cellular Systems*. Springer, 1996.

[10] J. H. S. Hofmeyr. Metabolic control analysis in a nutshell. In T. M. Yi, M. Hucka, M. Morohashi, and H. Kitano, editors, *Proc. 2nd International Conference on Systems Biology*, pages 291–300, Madison, WI, USA, 2001. Omnipress.

[11] B. Ingalls. A frequency domain approach to sensitivity analysis of biochemical networks. *Journal of Physical Chemistry B*, 108(3):1143–1152, 2004.

[12] V. Jacobson. Congestion avoidance and control. In *Proc. Symposium on Communications Architectures and Protocols*, pages 314–329, New York, NY, USA, 1988. ACM.

[13] V. Jacobson. A new way to look at networking, 2006.

[14] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. Briggs, and R. Braynard. Networking named content. In *Proc. 5th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2009)*, pages 1–12, 2009.

[15] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems, 1984.

[16] K. Kamimura, H. Hoshino, and Y. Shishikui. Constant delay queuing for jitter-sensitive IPTV distribution on home network. In *IEEE Global Telecommunications Conference (IEEE GLOBECOM 2008)*, 2008.

[17] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 482–491, 2003.

[18] J.-Y. Le Boudec and P. Thiran. *Network Calculus*, volume 2050 of *Lecture Notes in Computer Science*. Springer, 2004.

[19] H. Le Pocher, V. Leung, and D. Gillies. An application- and management-based approach to ATM scheduling. *Telecommunication Systems*, 12:103–122, 1999.

[20] S. H. Low. A duality model of TCP and queue management algorithms. *IEEE/ACM Transactions on Networking*, 11(4):525–536, 2003.

[21] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options. RFC 2018 (Proposed Standard), 1996.

[22] D. A. McQuarrie. Stochastic approach to chemical kinetics. *Journal of Applied Probability*, 4(3):413–478, 1967.

[23] T. Meyer and C. Tschudin. Chemical networking protocols. In *Proc. Hot Topics in Computer Networks (HotNets-VIII)*. ACM, 2009.

[24] L. Michaelis and M. Menten. Die Kinetik der Invertinwirkung. 49:333–369, 1913.

[25] V. Misra, W.-B. Gong, and D. Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. *SIGCOMM Computer Communication Review*, 30:151–160, 2000.

[26] J. Nagle. Congestion control in IP/TCP internetworks. RFC 896, 1984.

[27] A. S. Perelson and G. F. Oster. Chemical reaction dynamics part II: Reaction networks. 57(1):31–98, 1974.

[28] C. Reder. Metabolic control theory: A structural approach. *Journal of Theoretical Biology*, 135(2):175–201, 1988.

[29] K. Seong, R. Narasimhan, and J. Cioffi. Queue proportional scheduling in gaussian broadcast channels. In *IEEE International Conference on Communications*, volume 4, pages 1647–1652, 2006.

[30] S. Shakkottai and R. Srikant. How good are deterministic fluid models of Internet congestion control? In *Proc. 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, volume 2, pages 497–505, 2002.

[31] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks. *IEEE Trans. on Auto. Ctrl.*, 37(12):1936–1949, 1992.

[32] P. van Mieghem. *Performance Analysis of Communications Networks and Systems*. Cambridge University Press, 2006.

[33] P. Waage and C. M. Guldberg. Studies concerning affinity. *Forhandlinger: Videnskabs - Selskabet i Christiania*, 35, 1864.