The Conservation of Information, Towards an Axiomatized Modular Modeling Approach to Congestion Control

Corentin Briat, Emre Altug Yavuz, Håkan Hjalmarsson, *Fellow, IEEE*, Karl Henrik Johansson, *Fellow, IEEE*, Ulf T. Jönsson, Gunnar Karlsson, *Senior Member, IEEE, Member, ACM*, and Henrik Sandberg

***Abstract*—We derive a modular fluid-flow network congestion control model based on a law of fundamental nature in networks: the conservation of information. Network elements such as queues, users, and transmission channels and network performance indicators like sending/acknowledgment rates and delays are mathematically modeled by applying this law locally. Our contributions are twofold. First, we introduce a modular metamodel that is sufficiently generic to represent any network topology. The proposed model is composed of building blocks that implement mechanisms ignored by the existing ones, which can be recovered from exact reduction or approximation of this new model. Second, we provide a novel classification of previously proposed models in the literature and show that they are often not capable of capturing the transient behavior of the network precisely. Numerical results obtained from packet-level simulations demonstrate the accuracy of the proposed model.**

***Index Terms*—Congestion control modeling, conservation law, fluid-flow model, queuing model, self-clocking.**

I. Introductory Discussions

A. Congestion Control Problem

THE CONGESTION problem [1], [2] is inherent to communication networks where capacity of supporting infrastructure that relays information is small compared to user demand. Congestion is responsible for delay and data loss, which compromise the efficiency of the overall network. Controlling congestion is hence an important problem for which several algorithms have been developed. They mainly rely on the concept of *congestion window* (the number of desired outstanding packets), which is adapted according to a *congestion measure*. According to the type of congestion measure, two classes of congestion control algorithms may be identified [2], [3]. The first and oldest class is *loss-based*, meaning that the congestion measure is the packet-loss information. This class is easy to implement, but leads to a quite rough control since the protocol detects the network congestion only after provoking it. In order to control congestion more smoothly and prevent data loss, *delay-based* algorithms, using for instance the *round-trip time* (RTT) information as the congestion measure, may be considered instead. They are, however, more difficult to implement due to the possible unavailability of certain necessary measures, such as *queuing delays*.

The main difficulty in congestion control lies in the fact that, basically, the hosts ignore almost everything about the network: the routes and their capacity, the numbers of routers (hops), the number of users, etc. Hence, protocol designers face the problem of controlling a very large and complex system with actually very little information.

When designing a protocol, *stability* of the network is certainly the most important constraint. Performance criteria can be additionally considered in order to optimize the network behavior. For instance, we may want to use all available bandwidth (*efficiency*), share it equally between users (*fairness*), and/or be tolerant with respect to unregulated traffic and other protocols (*cross-traffic adaptation*).

In order to observe/predict the network behavior and validate a protocol, simulations and experiments must usually be conducted. NS-2 is a widely accepted open-source event-based simulator dedicated to this purpose. However, as any other simulator, it does not permit to analyze the stability of a network theoretically. Hence, constructing mathematical models for networks may play an important role in network analysis and protocol design since they potentially allow for a theoretical analysis and an equation-based design of new protocols.

B. Models, Approximations and Accuracy

Modeling is now ubiquitous. The key idea is to start from a system and arrive at an abstract representation of it, such as one given in terms of a set of mathematical equations. It is not always necessary that a model represents all characteristics of a system but only a subset of interest: e.g., a molecular-level model can be irrelevant to portray a river. This gave rise to fluid mechanics, which, although being an idealization of the reality, yields very accurate predictions. A similar idealization has been shown to be very useful for the congestion control problem through the consideration of *fluid-flow models* [4].

Manuscript received February 27, 2012; revised August 09, 2013; accepted February 16, 2014; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor V. Misra. Date of publication April 18, 2014; date of current version June 12, 2015. This work was supported by the ACCESS Linnaeus Center, KTH, Stockholm, Sweden.

C. Briat was with the ACCESS Linnaeus Centre, KTH, 10044 Stockholm, Sweden. He is now with the Department of Biosystems Science and Engineering, Swiss Federal Institute of Technology Zürich (ETH-Z), 8091 Basel, Switzerland (e-mail: corentin@briat.info).

E. A. Yavuz was with the ACCESS Linnaeus Centre, KTH, 10044 Stockholm, Sweden. He is now with Ericsson AB, 16483 Stockholm, Sweden.

H. Hjalmarsson, K. H. Johansson, G. Karlsson, and H. Sandberg are with the ACCESS Linnaeus Centre, KTH, 10044 Stockholm, Sweden (e-mail: hakan.hjalmarsson@kth.se; kallej@kth.se; gk@kth.se; hsan@kth.se).

U. T. Jönsson, deceased, was with the ACCESS Linnaeus Centre, KTH, 10044 Stockholm, Sweden.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2014.2308272

C. Metamodels and Network Models

A paragon of network modeling is undoubtedly used in the field of electrical engineering. It is easy to identify the reasons for the success of the theoretical framework:

1) only two universal concepts: current and voltage, governed by simple laws (Kirchhoff's laws);
2) local description of the elements in terms of these variables and additional local concepts (e.g., resistance, etc.);
3) easy transcription of the electrical network into a topologically identical diagram, and vice versa;
4) new models corresponding to new devices may be freely added without compromising the existing ones;
5) model predictions fit very well to reality;
6) systematic way of analysis by hand calculations or simulators.

A very important feature is that the principles of modeling an electrical network is independent of its topology and elements. This is achieved thanks to the structure of the paradigm that we refer from now on as a *metamodel*, which is a model that consists of a set of frames, rules, constraints, submodels, and theories applicable and useful for modeling a predefined class of problems. In the case of electrical networks, the metamodel consists of the concepts of current and voltage, the Kirchhoff's laws, and the local models of electrical elements (resistor, capacitor, transistor, etc.), as well as all the related mathematical tools. Since networks (like communication networks, electrical networks, transportation networks, and even social networks) consist of interconnections of several elements, it turns out that metamodels are then very suitable for describing them since they also consist of interconnection of concepts, rules, and submodels. Hence, metamodels provide, in essence, an elegant scalable and modular way for modeling networks.

D. Motivations and Contributions

The main motivation of this work is to give a clear picture of congestion control modeling problem through derivation of a metamodel having solid mathematical foundations. We introduce a modular metamodel that would lead to an interesting step forward toward a generic way of providing models for communication networks. This metamodel should then satisfy the additional constraints on independence of network topology (scalability) and elements (modularity). It should also provide accurate predictions along with simple graph representation. An underlying difficulty is the presence of several phenomena at different levels: decision to send a packet, transmission of packets on transmission channels, storage of packets in queues and time-varying waiting-time (queuing delays), congestion window size adaptation, etc. Finding a unified way for representing all these critical phenomena is challenging.

The proposed metamodel is based on a single concept of information conservation, from which models for the network constituents (i.e., transmission channels, queues, and users) are obtained. This allows to derive new models for network elements, obtain mathematical proofs for unproved/claimed existing ones, and invalidate some of them. All important variables of the network (sending rates, ACK rates, queue size, etc.) are described by explicit formulas, hence computable. Using the proposed metamodel, describing a given topology is immediate and performed by simply plugging the models together, so as

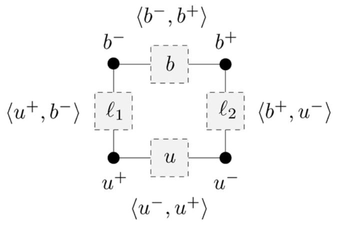
$\langle b^-, b^+\rangle$
b^-
b
b^+
$\langle u^+, b^-\rangle$
ℓ_1
ℓ_2
$\langle b^+, u^-\rangle$
u
u^+
u^-
$\langle u^-, u^+\rangle$

Fig. 1. Example of graph with four edges: one user edge $u = \langle u^-, u^+\rangle$, one buffer edge $b = \langle b^-, b^+\rangle$, and two transmission edge $\ell_1 = \langle u^+, b^-\rangle$, $\ell_2 = \langle b^+, u^-\rangle$.

the actual topology is transcribed into a graph having the different network elements located on the edges, as in electrical engineering. It is also proved that existing sending rate models are either approximations of the proposed sending rate model, or even exact provided that the network topology satisfies certain structural conditions. Simulations and comparisons to existing works tend to suggest the relevance, reliability, and accuracy of the proposed metamodel. A nonexhaustive summary of related works on congestion control modeling is finally made in order to compare congestion control models according to important properties and criteria.

The outline of the paper is as follows. Section II introduces the particular network graph representation considered in the paper. Using continuous-time models, such as fluid-flow models, to describe networks is justified, and concepts of universal clock, local discrete-time network element clock, and clock-coupling are defined in Section III. Section IV presents the conservation law of information, and we derive the transmission channel model in Section V. In Section VI, we develop the model for the first-in–first-out (FIFO) buffer network element. The user model is given in Section VII, and we summarize the obtained results in a compact form in Section VIII. In Section IX, we consider a network with single-buffer/multiple-user topology to implement the proposed model. We validate our model in Section X, and related work is given in Section XI. Section XII concludes the paper.

II. Networks and Graphs

It is convenient to introduce here the particular network graph representation considered in the paper. It is different from the traditional ones [5]–[8] since it places all network elements on graph edges, leaving nodes with the role of connecting points, as in electrical circuits. Four types of nodes are distinguished: the input nodes u_i^-, b_j^- and output nodes u_i^+, b_j^+ for user u_i and buffer b_j, respectively. The superscripts have to be understood as a temporal order of reaction or causality: the data come (-) then leave (+). We will denote any edge E of the graph by $\langle x, y\rangle$, where x and y are the input and output nodes, respectively. Moreover, given any edge E, the input and output nodes are given by $\beta(E)$ and $\varepsilon(E)$, respectively.

According to these definitions, a queue edge is always denoted by $\langle b_i^-, b_i^+\rangle$, a user edge by $\langle u_i^-, u_i^+\rangle$, and a transmission edge by $\langle b_i^+, u_j^-\rangle$, $\langle u_i^+, b_j^-\rangle$ or $\langle b_i^+, b_k^-\rangle$, $i \neq k$. This is illustrated in Fig. 1. We call a circuit C_i the communication path of user u_i, that is the path connecting its output u_i^+ to its input u_i^-, i.e., $C_i = \langle u_i^+, u_i^-\rangle$. Note that in complex networks there exist several possible paths, but only one of them, the one used for communication, is a circuit. In Fig. 1, the only possible circuit is given by $C = \langle u^+, b^-, b^+, u^-\rangle$.

III. Fluid-Flow Idealization

In an asynchronous network like the Internet, each element can be considered to have its own local discrete-time clock $\mathbb{T}_i \subset \mathbb{R}_+$ where $\mathbb{T}_i$ is countable, governing the rhythm of protocol decisions and packets transmission. In congestion control, the clocks beat with the rhythms of acknowledgment reception rates, which are influenced in turn by network congestion; this is referred to as *ACK-clocking*[1]. When several sources send data through the same buffer/path, a flow-coupling takes place leading then to clock-coupling. This clock-coupling arises at a very large scale, and distant sources having their clocks coupled cannot be considered to have independent behaviors. As a consequence, the sending rates and acknowledgment rates are hence intimately interdependent. Modeling this clock-coupling and the underlying phenomena is of incredible complexity since the number of clocks and their interactions grow very quickly with the network complexity, leading then to a very complicated structure for the interrelated local clocks $\mathbb{T}_i$; see for example [9, Eq. (3.7)].

An idea to resolve this complex time-structural problem relies on the definition of a *universal clock* $\mathbb{T}^u$ dictating a common time to the entire network. This leads us to the following fact:

Fact 1: There exists an ideal universal clock $\mathbb{T}^u$ embedding any local clock $\mathbb{T}_i$, i.e., $(\bigcup_i \mathbb{T}_i) \subset \mathbb{T}^u$. $\therefore$

A natural universal clock is given by $\bigcup_i \mathbb{T}_i$ and is a discrete-time clock. It, however, does not simplify too much the modeling problem since it is difficult to write recurrence relations for general network topologies [9]. Deriving a metamodel achieving scalability is then unlikely using such a universal clock. A less natural universal clock $\mathbb{T}^u$ assimilated to a clock running over positive real numbers continuously, i.e., $\mathbb{T}^u \equiv \mathbb{R}_+$, is much more promising. This particular universal clock indeed dramatically simplifies the modeling problem, and this motivates its consideration in this paper. Using such a timescale, a metamodel can be obtained, resulting then in a scalable solution in which the network asynchrony is captured through appropriate expansions and compressions of the time-space. Furthermore, it enables the use of well-established mathematical tools: real function analysis, integration theory, dynamical systems, delay-differential equations, etc. A conclusion is that continuous-time models may be used to describe networks [4], [6], [10]–[12]: These are generally referred to as *fluid-flow models*, emphasizing the connection with continuum mechanics and more specifically with fluid mechanics.

Within this framework, it is possible to provide a proper definition for data flows.

Definition 2: Let us consider a point x in an edge E of the network and denote the number of packets having passed through point x between $t_0 \in \mathbb{T}^u$ and $t \in \mathbb{T}^u$, $t \geq t_0$, by $N_x(t, t_0)$. Then, the flow of data passing through point x is defined as a function $\phi : E \times \mathbb{T}^u \to \mathbb{R}_+$ verifying

$$N_x(t, t_0) = \int_{t_0}^{t} \phi(x, s) ds \tag{1}$$

where the integral is a standard one, e.g., the Lebesgue integral.

Flows are defined in such a wayb rather than being the derivative of the number of packets, since the number of packets is nondifferentiable, i.e., flows may contain dirac pulses, steps, and so on. It is also interesting to note that since the universal clock embeds all the local clocks, it is possible to recover discrete-time asynchronous models such as the one in [9] by setting flows to be trains of dirac pulses on $\bigcup_i \mathbb{T}_i$.

Using the notation defined in Section II, we can build the flow vectors $\phi(x,t)$ using the "col" operator as $\phi(x,t) = \mathrm{col}_{k=1}^{\eta(x)} [\phi_k(x,t)]$, where x is any input and output node of the network elements, i.e., x can be any u_i^-, u_i^+, b_i^-, b_i^+. The quantity $\eta(x)$ denotes the number of flows passing through node x. The concept of flows of data is hence very close to those of current in electrical engineering and flow of a liquid in fluid mechanics.

IV. Conservation Law of Information

The core of the metamodel is the conservation law of information stated in this section. This law allows to improve the characterization of the elements of the network by notably clarifying their input/output relationship, enabling then a modular formalism. This conservation law follows from the remark that the quantity of information[2] is preserved in a communication network: The data can either be in transit, lost, or received. Assuming lossless networks, it is possible to determine the total number of packets in transit in any edge, simply by counting the number of entering packets according to a simple rule. When applied to a specific element, this law allows to characterize the fact that the information is preserved from the input to the output.

Law of Conservation of Information: Given any edge E of a network, then for all $t \in \mathbb{T}^u$, there exists a time $t_0(t) \in \mathbb{T}^u$, $t_0(t) \leq t$ such that

$$\begin{aligned} P_E(t) &:= \int_E \phi(\theta, t) d\theta \\ &= \int_{t_0(t)}^{t} \phi(\beta(E), s) ds \\ &= N_{\beta(E)}(t, t_0(t)). \end{aligned} \tag{2}$$

The integration over E is an abstract integral that has to be understood as a flow spatial integration from $\beta(E)$ to $\varepsilon(E)$, that is, the number of packets $P_E(t)$ in the edge $E = \langle \beta(E), \varepsilon(E) \rangle$ at time t. $\therefore$

The above result stated in quite abstract terms just says that the number of packets in transit in an edge at a certain time t can be determined by counting the number of entering packets (i.e., integrating the input flow) over the interval $[t_0(t), t]$, the lower bound t_0 of the interval depending on the considered element, i.e., transmission channel, queue, or user. A simple application of the law is given in Section V discussing transmission channels models.

The first benefit of this law is to show that we can interchangeably use a spatial or a temporal integral to calculate the quantity of information (number of packets) in an edge. The temporal integral formulation is very convenient to work with since it requires the knowledge of the input flow only, rather than the flow value on the entire edge for the spatial integral formulation. This hence allows to *discretize the space dimension* by only considering flows at the nodes, simplifying then the network representation and the modeling problem.

The second benefit lies in the fact that the temporal integral can be utilized to yield explicit solutions for the output flows of

[1]The term *self-clocking* is also used, but is less explicit.

[2]expressed in bits or packets.

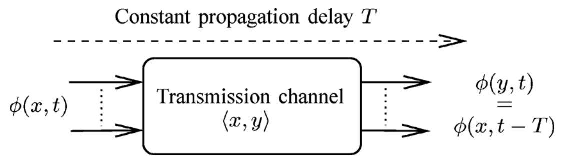

Fig. 2. Transmission channel block.

the different network elements. A very general result is given in the following.

Proposition 3: The input flow $\phi(\beta(E),\cdot)$ and the output flow $\phi(\varepsilon(E),\cdot)$ of edge E verify

$$\phi(\varepsilon(E),t) = t_0(t)'\phi(\beta(E),t_0(t)) \tag{3}$$

where we assume that $t_0(t)$ is absolutely continuous and $t_0(t)'$ is the upper-right Dini derivative of t_0 at t, i.e., $t_0'(t) = \limsup_{h\downarrow 0} h^{-1}(t_0(t+h)-t_0(t))$.

Proof: Since $N_{\beta(E)}(t,t_0(t))$ is the current number of packets on edge E at time t, then differentiation with respect to time provides the balance equation

$$[N_{\beta(E)}(t,t_0(t))]' = \phi(\beta(E),t) - t_0(t)'\phi(\beta(E),t_0(t)).$$

Note also that a second valid balance equation is given by

$$[N_{\beta(E)}(t,t_0(t))]' = \phi(\beta(E),t) - \phi(\varepsilon(E),t).$$

Identifying the right-hand side yields the result. ■

The proposition given above plays a crucial role in the meta-modeling problem since it provides an explicit formula of the output flows. This output flow verifies the conservation of information from the input to the output of the edge E. By integrating the input and output over $[0,\infty)$, the very same value is obtained. This emphasizes that the output flow is defined in such a way that, as desired, it respects the natural property of conservation of information. Proposition 3 is used repeatedly in the paper in order to provide accurate and explicit models for transmission channels, queues and users. Applications are given in Sections V, VI-C, and VII-D.

V. Transmission Channel Model With Constant Propagation Delay

The first element-model is derived in this section, namely the model for lossless transmission channels with constant delay. They exactly behave as transmission lines and a delay-based formulation is provided. The derivation is rather straightforward, but it is a good example of application of Proposition 3.

Result 4: Given a lossless transmission channel corresponding to edge E and having constant propagation delay $T > 0$, the output flow is given by

$$\phi(\varepsilon(E),t) = \phi(\beta(E),t-T). \tag{4}$$

The corresponding module is depicted in Fig. 2.

Proof: According to the conservation law (2), the number of packets in transit $P_E(t)$ in the edge E at time $t \in \mathbb{T}^u$ obeys

$$P_E(t) = \int_{t_0(t)}^{t} \phi(\beta(E),s)ds \tag{5}$$

where $t_0(t) = t-T$ since the propagation delay is constant. A packet sent at time $t-T$ will indeed be, at time t, still in the edge and about to leave. The result follows then from Proposition 3. ■

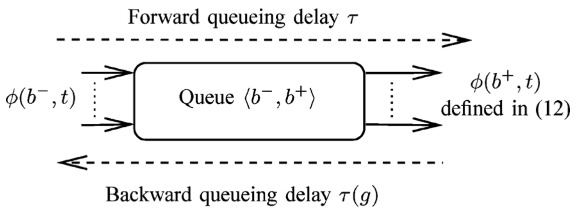
Forward queueing delay τ
$\phi(b^-,t)$
Queue $\langle b^-,b^+\rangle$
$\phi(b^+,t)$
defined in (12)
Backward queueing delay $\tau(g)$

Fig. 3. Queue/buffer block.

We are now in a position to introduce transmission channel operators defining part of the network topology.

Definition 5: The flow vectors $\phi(u^-,t) = \text{col}_i[\phi(u_i^-,t)]$, $\phi(b^-,t) = \text{col}_i[\phi(b_i^-,t)]$, $\phi(u^+,t) = \text{col}_i[\phi(u_i^+,t)]$, and $\phi(b^+,t) = \text{col}_i[\phi(b_i^+,t)]$ are related by transmission channel operators $\mathcal{R}_\bullet$, $\bullet \in \{ub,bu,bb\}$ as

$$\begin{bmatrix}\phi(u^-,t)\\ \phi(b^-,t)\end{bmatrix} = \mathcal{R}\begin{bmatrix}\phi(u^+,t)\\ \phi(b^+,t)\end{bmatrix} + \mathcal{D}\delta(t) \tag{6}$$

where

$$\mathcal{R} = \begin{bmatrix}0 & \mathcal{R}_{ub}\\ \mathcal{R}_{bu} & \mathcal{R}_{bb}\end{bmatrix} \quad \text{and} \quad \mathcal{D} = \begin{bmatrix}0\\ \mathcal{D}_b\end{bmatrix}.$$

The matrices $\mathcal{R}_\bullet$, $\bullet \in \{ub,bu,bb\}$ correspond to routing matrices between output and input nodes. For instance, $\mathcal{R}_{ub}$ maps flows at user output nodes to flows at buffer input nodes. These matrices essentially consist of constant delay operators with delays corresponding to transmission channels. The full-rank input matrix $\mathcal{D}_b$ drives the vector of cross-traffic flows $\delta(t)$ to buffer input nodes.

VI. FIFO Buffer Model

This section is devoted to the very important buffer element that temporarily stores incoming information before processing it. First, the standard fluid model for queues is recalled [2], [3], [11], and a complete delay-map is characterized. In order assign each input flow to its corresponding output flow and solve the output flow separation problem [13]–[18], the conservation law is then applied on the standard queue model, in a similar way as for transmission channels. The output separation problem allows us to focus on the accurate description of queues, which captures both the FIFO behavior and the form of output flows; see Fig. 3. Some extra discussions and interpretations of the results are also provided. Finally, a comparison to an existing model for output flows is carried out and concludes in favor of the proposed one.

A. Queue Model

Routers have queues to store incoming packets temporarily. The following integrator model [11] can be proved to be a limit model of an M/M/1 queue when the packet size and thus the processing time tend to 0 [2], [19].

Definition 6: The queue dynamics of buffer b_i is governed by the model

$$\dot{q}_i(t) = \sum_{j=1}^{\sigma(b_i)} \phi_j(b_i^-,t) - r_i(t) \tag{7}$$

where the aggregate output rate is defined as

$$r_i(t) = \begin{cases} c_i, & \text{if } \mathcal{C}_i(t)\\ \sum\limits_j \phi_j(b_i^-,t), & \text{otherwise.}\end{cases} \tag{8}$$

Above, q_i, c_i, and $\phi_j(b_i^-,t)$ represent the queue size, the maximal output capacity, and the flow of type j at the input, respectively. The condition $\mathcal{C}_i(t)$ is given by

$$\mathcal{C}_i(t) := \left([q_i(t) > 0] \vee \left[\sum_j \phi_j(b_i^-,t) > c_i\right]\right). \quad (9)$$

The corresponding queuing delay can be easily deduced using the relation $\tau_i(t) = q_i(t)/c_i$. $\therefore$

The above model can also be refined to capture additional features such as finite maximal queue length, flow priorities, and multiple output capacities. These extensions are omitted here since they are straightforward. It is important to stress that this model is incomplete and useless in this form. First, the output flow is given in aggregate form. This prevents the modeling of the appropriate routing of each output flow. Second, it does not capture the queue FIFO behavior. Finally, the model does not assign a specific queueing time to each input flow. In Section VI-B, the conservation law (2) is used in order to confer the FIFO property to the model and separate the aggregate output flow into distinct flows.

B. Forward and Backward Queuing Delays

The maps defined in this section are very useful for obtaining a closed formula for the buffer output flows in Section VI-C and for RTT in Section VII-C.

Let us consider first the buffer model (7) with queueing delay $\tau_i(t)$. Assume that the time instants at which packets enter the queue are chosen as reference times. We may then be interested in predicting the packet output time. This leads to the following definition.

Definition 7 (Forward Delay Operator): The forward delay operator $f_i : \mathbb{T}^u \to \mathbb{T}^u$ corresponding to buffer b_i mapping, at a flow level, any input-time t to the output time $f_i(t)$ is defined as

$$f_i(t) := t + \tau_i(t) \quad (10)$$

where $\tau_i(t)$ is the queuing delay of buffer b_i.

It is easy to see that output time can be readily computed from the knowledge of input time and queuing delay value. If, however, we would like to set the reference time to be the output time, we may ask the question whether it is possible or not to retrieve the input time from it. This is equivalent to asking the question of invertibility of the map f_i.

Result 8 ([17]): The map f_i is invertible if and only if the input flow of the corresponding buffer is positive almost everywhere. ▲

Hence, provided that there is a nonzero input flow to the buffer, the input time corresponding to a given output time is well defined and can be obtained using the *backward delay operator*.

Definition 9: The backward delay operator $g_i : \mathbb{T}^u \to \mathbb{T}^u$ corresponding to buffer b_i mapping, at a flow level, any output time t to the input time $g_i(t)$ is defined as $g_i := f_i^{-1}$ under the assumption of Result 8.

We also have the following useful results.

Result 10 ([17]): The functions f_i and $g_i := f_i^{-1}$ obey

$$g_i'(t) = \begin{cases} c_i \left(\sum_{k=1}^{\eta(b_i)} \phi_k(b_i^-, g_i(t))\right)^{-1}, & \text{if } \mathcal{C}_i(g_i(t)) \\ 1, & \text{otherwise} \end{cases}$$

$$f_i'(t) = \begin{cases} c_i^{-1} \sum_{k=1}^{\eta(b_i)} \phi_k(b_i^-, t), & \text{if } \mathcal{C}_i(g_i(t)) \\ 1, & \text{otherwise} \end{cases}$$

$$[\tau_i(g_i(t))]' = \begin{cases} 1 - \frac{c_i}{\sum_{k=1}^{\eta(b_i)} \phi_k(b_i^-, g_i(t))}, & \text{if } \mathcal{C}_i(g_i(t)) \\ 1, & \text{otherwise} \end{cases} \quad (11)$$

where $f'(t)$ stands for the upper-right Dini derivative of $f(t)$, i.e., $f'(t) = \limsup_{h\downarrow 0} h^{-1}(f(t+h) - f(t))$. ▲

The following technical result allows to simplify the conditions involved in hybrid models.

Result 11: The equivalence $\mathcal{C}(g_i(t)) \Leftrightarrow \mathcal{C}(t)$ holds.

Proof:

Proof of $\Rightarrow$: If the buffer is congested at time $g_i(t)$, then the buffer will also be congested at time t since the data entered at time $g_i(t)$ leave at time t.

Proof of $\Leftarrow$: Conversely, if there is any data to leave at time t, they must have entered in the queue in the past, i.e., at time $g_i(t)$. Equivalence is proved. ■

C. FIFO Buffer Output Flow Separation

In this section, we use the results given above and the conservation law in order to improve the buffer modeling by adding the FIFO characteristics and splitting the aggregate output flows into a sum of distinct ones. Without further consideration on the queue type, there exists an infinite number of ways to separate the aggregate output flow directly from the queuing model of Definition 6. When a FIFO queue (i.e., order preserving) is considered, it turns out that the output flow separation problem is easily solvable. The FIFO characterization and output flow separation problems have been fully solved in [17]. In this section, we will simply recall and explain these results and connect them to the conservation law (2).

Result 12 ([17]): Let us consider the queueing model (7), which we assume to represent a FIFO queue. The output flow corresponding to the input flow $\phi_\ell(b_i^-,t)$, $\ell \le \eta(b_i^-)$ is given by

$$\begin{aligned}\phi_\ell(b_i^+,t) &= g_i'(t)\phi_\ell(b_i^-, g_i(t)) \\ &= \begin{cases} \frac{c_i \phi_\ell(b_i^-, g_i(t))}{\sum_j^{\eta(b_i)} \phi_j(b_i^-, g_i(t))}, & \text{if } \mathcal{C}_i(t) \\ \phi_\ell(b_i^-, t), & \text{otherwise.} \end{cases}\end{aligned} \quad (12)$$

Proof: A proof is available in [17]. A more direct one based on Proposition 3 is given here. Noting that for buffer b_i, we have $t_0(t) = g_i(t)$, then using Proposition 3 and the formulas of Result 10, we get (12) with the difference that the condition is $\mathcal{C}(g_i(t))$. However, from Result 11, the condition $\mathcal{C}(g_i(t))$ is equivalent to $\mathcal{C}(t)$, and the result follows. ■

The same model has been also proposed in [13] and [16], but claimed without any proof. We have shown above that this model is an immediate consequence of the information conservation law and gives, for the first time, a theoretical proof for it. This considerably strengthens the trust we may have in this model. A comparison to packet-level simulations in Section VI-D tends to show its exactness.

This model also deserves interpretation. Formula (12) says that output flows consist of scaling and shifting of the input flows. The delay accounts for high flow viscosity and captures the queue FIFO behavior, *at a flow level*, while the nonlinear ratio expresses the flow coupling at the core of the flow and clock-coupling phenomena, see Section III, since each output flow depends on the corresponding input flow and all the other ones as well. A change in a single flow will affect all the output flows. This model also tells that the output flow corresponding to the input flow $\phi_\ell(b^-,t)$ is expressed as a (delayed) ratio of the input flow $\phi_\ell(b^-,t)$ to the total input flow that entered the buffer at the same time. Hence, the output flows are proportional to relative flows modeling the "chance" of having a packet of certain type served at time t. This "chance" is then scaled up by the maximal output capacity to utilize the available capacity.

Operators representing buffers can now be introduced:

Definition 13: The buffer operator $\mathcal{B}_i$ with $\eta(b_i)$ input flows is defined as

$$\mathcal{B}_i : \phi(b_i^-,t) \to \phi(b_i^+,t) \tag{13}$$

where the output flows and the buffer state are governed by (7) and (12). Using these operators, we can build a matrix of operators $\mathcal{B}$ connecting the $\phi(b^-,t)$'s to the $\phi(b^+,t)$'s as

$$\phi(b^+,t) = \mathcal{B}\phi(b^-,t) \tag{14}$$

where $\mathcal{B} = \mathrm{diag}_i\{\mathcal{B}_i\}$.

D. Comparison to Another Model

Two main models for buffer output flows have been reported in the literature on fluid-flow models: the flow-based model [13], [16], [17] described in this paper and the pseudo-queue-based one [14], [15], [18] given by

$$\phi_\ell(b_i^+,t) = \begin{cases} \frac{q_i^\ell(t)c_i}{\sum\limits_k q_i^k(t)}, & \text{if } q_i(t) \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

where $q_i^\ell(t)$ is the number of packets of type ℓ in queue i and c_i is the maximal output capacity of queue i.

Until now, these models have not been confronted to each other. In the following, they will be theoretically and experimentally compared, and it will be shown that the flow-based model is the only model that faithfully characterizes the actual output flows, validating then the proposed conservation-law-based paradigm.

1) Theoretical Argumentation: First, the above model assumes that the "chance" of having a packet of type ℓ at the output at time t is $q_i^\ell(t)/\sum_i q_i^\ell(t)$. Model (15) then makes no difference in picking a packet in the middle, at the end, or at the beginning of the queue since only the number of packets matters. It is thus unable to capture the FIFO characteristic of the queue since swapping packets in the queue does not modify the output flow. In contrast, the proposed model does capture this characteristic through the delay dynamical model and the delayed nonlinear input–output relationship involving flows directly: Relative variations of the input flows are passed to the output flows after some queueing delay. Note, however, that both models coincide at equilibrium.

Second, since output flows in model (15) are computed from the integration of input flows, it turns out that the map from input flows to output flows is a nonlinear low-pass filter with "bandwidth" equal to $1/q(t)$ when $q(t) > 0$. High frequencies in the input flows are hence filtered out, making the existing model inaccurate for high frequency flows (fast transient), especially when the queue size is large. Note that the actual buffer behavior does not have any filtering effect; it just behaves as an ordered tank. On the other hand, the proposed model does not filter out any frequency band due to its feedthrough structure. It, however, has a distortion effect on the input flows due to the nonlinear structure and the dynamically changing delay, i.e., change of frequency and amplitude. As a result, the queue-based model does not satisfy any conservation law since low-pass filters dissipate energy all over the frequency band, resulting in information loss at the model level. This is in total contradiction with the actual queue behavior that just stores information and does not dissipate anything. Note, however, that the flow-based model intrinsically satisfies the conservation since the model is derived from it.

Lastly, the proposed model incorporates naturally the queuing delay in the expression, while for model (15), it is unclear which delay to consider since the order of arrival of data is not tracked.

2) Case Analysis: To compare the models, let us consider two input flows given by

$$\begin{aligned} \phi_1(t) &= (1+\beta)c\left(1+\mathrm{Sq}(\omega t)\right)/2 \\ \phi_2(t) &= (1+\beta)c\left(1-\mathrm{Sq}(\omega t)\right)/2 \end{aligned} \tag{16}$$

where $c > 0$ and $\omega > 0$ are the buffer output capacity and the oscillation of flows. The term $\beta > 0$ is a tunable parameter related to the amplitude of the inputs flows, and the function $\mathrm{Sq}(\omega t) := \mathrm{sign}(\sin(\omega t))$ is a square function of period $T := 2\pi/\omega$. Since the flows are in phase opposition, they lead to an alternation of packet types in the queue, while packet populations remain roughly close to each other at any time. Therefore, the output flows should reflect the actual content of the queues, and the model should be able to keep track of the order of arrival of packets in the queue.

The queue-based model (15) predicts the queues

$$q_i(t) = \theta(t)q_i(0) + \int_0^t \theta(t-s)\phi_i(s)ds, \qquad i = 1,2 \tag{17}$$

with $\theta(t) = \left(\frac{q(0)}{q(0)+\beta ct}\right)^{1/\beta}$ from which it is quite difficult to foresee the shape of the output flows. We can, however, note that the filtering effect of the convolution operator with kernel θ will deform the input flows, making then the output flows not square anymore.

When the proposed flow-based model is considered, it is enough to compute the forward and backward delays and apply the formula for output flows

$$\begin{aligned} \tau(t) &= \beta t + \tau(0), \\ f(t) &= (1+\beta)t + \tau(0) \\ g(t) &= \frac{1}{1+\beta}(t-\tau(0)) \\ \phi_1(b^+,t) &= c(1+\mathrm{Sq}(\omega g(t)))/2 \\ &= \frac{c}{2}\left[1+\mathrm{Sq}\left(\frac{\omega}{1+\beta}(t-\tau(0))\right)\right] \\ \phi_2(b^+,t) &= \frac{c}{2}\left[1-\mathrm{Sq}\left(\frac{\omega}{1+\beta}(t-\tau(0))\right)\right]. \end{aligned} \tag{18}$$

In this case, the predicted output flows have the same shape as the input flows, but with a different frequency and amplitude. The proposed output flow model then exactly captures the content of the queue, that is the alternation of blocks of size $N(T)$

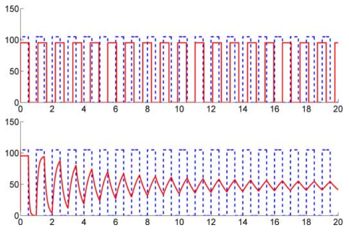

Fig. 4. Comparison of the output flows predicted by *(top)* model (12) and *(bottom)* model (15). Plain: output flow $\phi_1(b^+,t)$. Dashed: input flow $\phi_1(b^-,t)$.

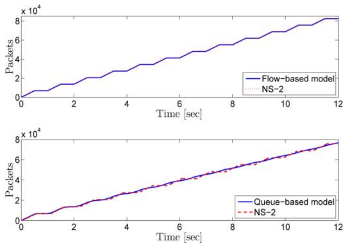

Fig. 5. Comparison to NS-2 simulations of the total number of packets counted at the inputs of the buffers described by *(top)* the proposed flow-based model (7)–(12) and *(bottom)* the queue-based model (7)–(7)–(15).

of type 1 and type 2. It is also immediate to see that the number of packets $N(T) = N_i(T),\ i = 1,2$ received by the buffer over one period T is given by

$$N(T) := \int_0^T \phi_1(b^+,s)ds = \frac{\pi(\beta+1)c}{\omega}. \qquad (19)$$

By virtue of the information conservation law, the same number of packets is retrieved at the output over the period $(\beta + 1)T$, enlarged due to the limiting output capacity c. It is quite convincing that the flow-based model yields a much more coherent picture for this example.

3) Simulation: For simulation, we choose $\omega = 2\pi$ (i.e., $T = 1$ s), $\beta = 1$, $c = 100$ Mb/s, and $\tau(0) = 0$. The output flows obtained from the different models are depicted in Fig. 4, where we observe the behaviors predicted by the above calculations. Notably, the output flows predicted by model (15) tend to slow down (low-pass filtering effect), decrease along time, and seem to both tend to $c/2$, which is basically unrepresentative of the actual content and output of the queue. We can also notice the decrease of the bandwidth for the queue-based model as long as the queue grows in size.

For comparison to NS-2, which deals with packets rather than flows, we integrate (up to an additional constant) the output flows predicted by each model to obtain a number of packets so that the comparison to NS-2 makes sense. The results are depicted in Fig. 5, where we can see that the proposed model yields exactly the same results as NS-2, while the queue-based model is unable to track the stair-like curve returned by NS-2. It is also important to stress that the considered scenario is quite convenient for the queue-based model since the input flows contain mostly constant parts (low frequency parts). A very fluctuating input flow would be very penalizing and would make the low-pass filtering effect of the queue-based model even more apparent.

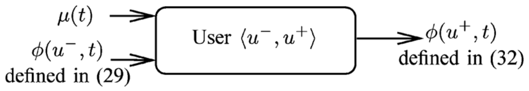

Fig. 6. User block.

VII. Complete User Model – Window Control

The derivation of the user model (see Fig. 6) is, partially, still an open question, and a complete solution, based on the conservation law (2), is proposed in this section. We assume that the implemented congestion control protocol admits a fluid-flow approximation, for instance interpolating the discrete-time trajectories of the real protocol [11], [20]. The conservation law is then applied over the circuit used by a user to obtain the so-called ACK-clocking model [21], [22] relating flight size, user sending rate, and RTT together. Several expressions for RTT are provided according to the considered reference time, similarly as for buffers. These results are finally merged together in order to clarify the connection between the flight size and the user sending rate. The last step concerns the derivation of formulas relating the above variables to user congestion window size.

A. Protocol Model

Here, we assume that the congestion control algorithm can be represented as a continuous-time (hybrid) dynamical system. That is, we have the following fact.

Fact 14: There exist bounded functionals $\mathcal{P}_i$, $\mathcal{W}_i$, and $\mathcal{U}_i$ such that the trajectories $(z_i(t), w_i(t))$ of the following continuous-time model defined over $\mathbb{T}^u$:

$$\begin{aligned}\dot{z}_i(t) &= \mathcal{P}_i(z_i(t),\mu_i(t))\\ w_i(t) &= \mathcal{W}_i(z_i(t),\mu_i(t))\\ \phi_i(u_i^+,t) &= \mathcal{U}_i(w_i(t),\phi_i(u_i^-,t))\end{aligned} \qquad (20)$$

match the trajectories of the asynchronous protocol (defined on $\mathbb{T}_i$) at points in $\mathbb{T}^u \cap \mathbb{T}_i$. Above, z_i, μ_i, $\phi_i(u_i^-,\cdot)$, and $\phi_i(u_i^+,\cdot)$ are the state of the protocol, the congestion measure, the acknowledgment flow rate, and the user sending flow, respectively. The window size, w_i, is considered as the number of outstanding packets to track and supposed to be (weakly) differentiable.

A procedure to solve the above interpolation problem has been first proposed in [11] for TCP and has been adapted to FAST-TCP in [20, Appendix C].

B. ACK-Clocking Model

The ACK-clocking model [21], [22] is a very important consequence of the conservation law. It characterizes the *flight size*[3] $\digamma_i(C_i,t) := P_{C_i}(t)$ of the user u_i at any time $t \in \mathbb{T}^u$ over a circuit $C_i = \langle u_i^+, u_i^-\rangle$. The importance of the ACK-clocking model lies in the semantic; it adds to the model by relating RTT, flow, and flight size together.[4] With this result, it is possible to

[3]The number of outstanding packets.

[4]Note, however, that in [22] the window size is considered instead of the flight size, which is rather different. To make the distinction, the window-based ACK clocking model is denoted by W-ACK, while the flight-size-based one by FS-ACK. Equivalence holds when some conditions, such as "flight size $\geq$ window size," are met. We will come back on this in Section VII-E.

incorporate a number of important user properties and mechanisms into the corresponding model.

Result 15 (FS-ACK-Clocking): The (FS)ACK clocking model is given by

$$\begin{aligned} F_i(C_i, t+\mathrm{RTT}_i\{t\}) &= \int_{C_i} \phi(\theta, t+\mathrm{RTT}_i\{t\})d\theta \\ &= \int_t^{t+\mathrm{RTT}_i\{t\}} \phi_i(u_i^+, s)ds \end{aligned} \qquad (21)$$

where $\mathrm{RTT}_i\{t\}$ is the RTT of a packet sent at time t by user u_i over the circuit C_i. ▲

Proof: Since flight size is a number of packets in a circuit, it can be cast as a spatial integration over the corresponding circuit. Thus, according to the conservation law, it is possible to convert the spatial integration into a temporal one provided that we can determine the integration bounds. To obtain them, we use the notion of RTT and suppose that a data sent by user u_i in the circuit C_i at time t has a round-trip time given by $\mathrm{RTT}_i\{t\}$. This means that the packets sent between t and $t+\mathrm{RTT}_i\{t\}$ are unacknowledged at $t+\mathrm{RTT}_i\{t\}$ and thus still in the circuit. Hence, the corresponding temporal integral has bounds t and $t+\mathrm{RTT}_i\{t\}$. ■

C. Round-Trip-Time Models

The RTT consists of sum of a constant and a time-varying part, namely the propagation delays and the queuing delays, which have been characterized in Sections V and VI, respectively. By combining these results together, it is immediate to obtain an RTT-based forward model on a *forward circuit operator*, which considers the packet input time as a reference.

To properly define it, let us consider a circuit C with N queues, indexed from 1 to N. The indices 0 and $N+1$ are used to denote the input and output nodes of the circuit, respectively. Given a packet input time t, the corresponding packet output time t_C and RTT obey the following formulas based on the forward circuit operator $\mathscr{F}_C$.

Definition 16: The forward circuit operator $\mathscr{F}_C : \mathbb{T}^u \to \mathbb{T}^u$ of circuit C assigned to any packet at input time t, an output time $t_C(t)$ is given by

$$\begin{aligned} t_C(t) &= \mathscr{F}_C(t) \\ \mathscr{F}_C &= \nabla_{N,N+1}^{-1} \circ f_N \circ \nabla_{N-1,N}^{-1} \circ f_{N-1} \circ \\ &\quad \cdots \circ \nabla_{2,3}^{-1} \circ f_2 \circ \nabla_{1,2}^{-1} \circ f_1 \circ \nabla_{0,1}^{-1} \end{aligned} \qquad (22)$$

where $\nabla_{i,j}$ is the constant delay operator with delay corresponding to the propagation delay between i and j and $\circ$ the composition operator. The corresponding RTT expression is then given by

$$\mathrm{RTT}\{t\} = t_C(t) - t = (\mathscr{F}_C - \mathrm{Id})(t) \qquad (23)$$

where Id is the identity operator.

Formula (22) actually represents the alternation between constant delay operators corresponding to transmission channels delay (the ∇_i's) and the queuing delays corresponding to queues (the f_i's). Example 18 illustrates this formula on the topology depicted in Fig. 1. The same formulas, albeit expressed in different ways, have been also obtained in [9, Section 3.3.5] and [22, Eq. (7d)–(7f)].

Although immediate to obtain, these expressions suffer from several drawbacks. First, operator $\mathscr{F}_C$ is clearly noncausal since it requires the knowledge of future information. Second, as pointed out in Section VI-B, the most convenient reference time to use is the output time. In the user modeling problem, it coincides with the reception time of acknowledgments and the moment when the user receives the RTT information. Hence, it seems to be more convenient to consider a *backward circuit operator* based on the backward delay operator of Section VI-B. The existence of such an operator is immediately inferred from the existence of the backward delay operator.

Definition 17: The backward circuit operator $\mathscr{B}_C : \mathbb{T}^u \mathbb{T}^u$ of circuit C assigned to any packet at output time t_C, an input time $t(t_c)$ is given by

$$\begin{aligned} \mathscr{B}_C &:= \mathscr{F}_C^{-1} \\ t(t_C) &= \mathscr{B}_C(t_C) \\ \mathscr{B}_C &= \nabla_{0,1} \circ g_1 \circ \nabla_{1,2} \circ g_2 \circ \nabla_{2,3} \circ \\ &\quad \cdots \circ g_{N-1} \circ \nabla_{N-1,N} \circ g_N \circ \nabla_{N,N+1}. \end{aligned} \qquad (24)$$

Moreover, the corresponding RTT expression is given by

$$\mathrm{RTT}\{t\} = t_C - t(t_C) = (\mathrm{Id} - \mathscr{B}_C)(t_C). \qquad (25)$$

It is important to stress that the RTT formula given above looks noncausal since the RTT of a packet sent at time t is defined in terms of time $t_C > t$. This is, however, not a problem since the RTT information is only available, and then used, by the user at time t_C, when the ACK packet is actually received. What is important is the causality of the operator $\mathscr{B}_C$ in order to ensure computability of the RTT at any time. This emphasizes once again the relevance of considering output times as references. The following example illustrates the above discussions.

Example 18: In the single-user/single-buffer case, the above expressions reduce to

$$\begin{aligned} t_C(t) &= t + T_\mathrm{f} + T_\mathrm{b} + \underbrace{\tau(t+T_\mathrm{f})}_{\text{Future information}} \\ t(t_C) &= \underbrace{g(t_C - T_\mathrm{b})}_{\text{Past information}} - T_\mathrm{f} \\ \mathrm{RTT}\{t\} &= t_C - t \\ &= t_C - g(t_C - T_\mathrm{b}) + T_\mathrm{f} \\ &= T_\mathrm{b} + T_\mathrm{f} + \tau(g(t_C - T_\mathrm{b})) \end{aligned} \qquad (26)$$

where T_f and T_b are the forward and backward propagation delays corresponding to the constant delay operators $\nabla_{0,1}$ and $\nabla_{1,2}$.

Using the backward expression of RTT and the relation $\mathscr{F}_{C_i} \circ \mathscr{B}_{C_i} = \mathscr{B}_{C_i} \circ \mathscr{F}_{C_i} = \mathrm{Id}$, it easy to obtain the following result.

Result 19: The flight size obeys

$$F_i(C_i, \mathscr{F}_{C_i}(t)) = \int_t^{\mathscr{F}_{C_i}(t)} \phi(u_i^+, s)ds \qquad (27)$$

$$F_i(C_i, t) = \int_{\mathscr{B}_{C_i}(t)}^t \phi(u_i^+, s)ds. \qquad (28)$$

▽

Proof: This is an immediate consequence of the conservation law (2) (through the ACK-clocking model) and the above RTT models. ■

Note that in [22], a model is obtained directly from the W-ACK-clocking applied directly to a selected topology. However, the model is not modular itself since hand calculations are needed to make it implementable, which results unfortunately in complexity very sensitive to the topology.

Hence, the objective of obtaining a metamodel is not attained. The reason is that the ACK-clocking model is used in *implicit form*, while our proposed method incorporates solutions of it, yielding an *explicit formulation* achieving the characteristics of a metamodel, i.e., modularity and scalability.

D. ACK-Clocking Dynamics and User Flow Computation

Since the flight-size expression (28) is exactly of the form (2), then Proposition 3 can be immediately applied to derive an explicit expression for the output flow of a given circuit, which is the flow of received acknowledgments.

Result 20: Let us consider a circuit $C_i = \langle u_i^+, u_i^- \rangle$. Then, the ACK-flow that user u_i receives is given by

$$\phi(u_i^-, t) = \mathscr{B}'_{C_i}(t)\phi_i(u_i^+, \mathscr{B}_{C_i}(t)). \tag{29}$$

$\triangledown$

Proof: The key idea is to remark that $\mathcal{F}_i(C_i, t) = N_{u_i^+}(t, \mathscr{B}_{C_i}(t))$. Hence, using Proposition 3 and noting that the ACK-flow corresponds to the flow leaving the circuit $\phi(u_i^-, t)$, the result is obtained. Differentiability of $\mathscr{B}_{C_i}(t)$ is inferred from the differentiability of the backward delay operators. ■

Note that differentiation of (28) yields

$$\phi_i(u_i^+, t) = \mathcal{F}_i'(C_i, t) + \mathscr{B}'_{C_i}(t)\phi_i(u_i^+, \mathscr{B}_{C_i}(t)) \tag{30}$$

meaning that to maintain a constant flight size, i.e., $\mathcal{F}_i'(C_i, t) = 0$, the user has to naturally send data at the same rate it receives ACK packets: This is exactly ACK-clocking, but expressed at a flow level. By flow integration, we can easily recover the "packet-level ACK-clocking." A similar expression is reported in [18], but stated without proof and using the model in (15) to represent the buffer output flows. Once again, the proposed methodology allows to provide strong mathematical foundations for some existing results.

E. User Flow, Flight Size, and Congestion Window Size

We need to clarify the relation between a user congestion window size $w_i(t)$ and its sending rate $\phi(u_i^+, t)$. First, recall that the congestion window size corresponds to the desired flight size, while the flight size is the current number of packets in transit. The window size is then a *reference* to track, while the flight size is the *controlled output*. The *control input* is the user sending rate.

When the window size increases, the user can immediately send a burst of packets to equalize the flight and window sizes. In such a case, we can ideally assimilate them to be equal[5] (and so are their derivatives). The small delay corresponding to the protocol reaction time can be easily incorporated in the constant part of the RTT. The problem is, however, slightly more difficult when the congestion window size becomes smaller than the flight size. In such a case, we cannot withdraw packets from the network, and the only thing we can do is to wait for the packets in the network to be acknowledged until, at some point, the flight size becomes equal to the window size. This basically means that the rate of decrease of the flight size is equal to the rate of received acknowledgments (the rate at which data leave the network). Therefore, while positive slope of the flight size is ideally unconstrained from above, the negative slope is lower bounded.

[5]This is the main assumption in [22] justifying the use of the W-ACK-clocking model.

In [9], a rate limiter is *a posteriori* added to the model in order to constrain the negative slope of the flight size. This solution is, however, difficult to implement in the context of [9] due to the time-varying nature of the slope lower bound and the lack of any ACK-flow model. Note also that in most recent works [18], [22], this problem is automatically excluded by considering that the flight size is always smaller than the window size, and that the window size does not decrease "too much." To the authors' best knowledge, no well-rounded solution has been provided yet for the problem of congestion window size decrease. We provide below an explicit and complete solution to this problem, regardless of the rate of the variation of congestion window size. This is achieved through an augmentation of the user model and the consideration of the flow of ACK packets.

According to the above discussion, the flight size must obey

$$\mathcal{F}_i'(C_i, t) = \begin{cases} \dot{w}_i(t), & \text{if } \mathcal{T}_i(t) \\ -\phi(u_i^-, t), & \text{otherwise} \end{cases} \tag{31}$$

where $\mathcal{T}_i(t)$ is a condition that is true when no lower limit on the rate of variation of the flight size is imposed, and false otherwise.

Result 21: The flight size $\mathcal{F}_i(C_i, t)$ satisfies (31) if the user sending rate is defined as

$$\phi(u_i^+, t) = \begin{cases} \dot{w}_i(t) + \phi(u_i^-, t), & \text{if } \mathcal{T}_i(t) \\ 0, & \text{otherwise} \end{cases} \tag{32}$$

where $\mathcal{T}_i(t) = \left([\pi_i(t) = 0] \wedge [\dot{w}_i(t) + \phi(u_i^-, t) \geq 0]\right)$ and

$$\dot{\pi}_i(t) = \begin{cases} 0, & \text{if } \mathcal{T}_i(t) \\ \dot{w}_i(t) + \phi(u_i^-, t), & \text{otherwise.} \end{cases} \tag{33}$$

Moreover, this model is the simplest one. ▲

Proof: The ACK-buffer π_i, taking nonpositive values, measures the number of ACK packets to retain in order to balance the flight and window sizes. When the virtual buffer has negative state, i.e., $\pi_i(t) < 0$, the arriving ACK-packets have to be retained until the state reaches 0. Once zero is reached, the user can start sending again until the window size decreases too fast, i.e., $\dot{w}_i(t) < -\phi(u_i^-, t)$. Substitution of the user sending rate defined by (32) and (33) in (30) yields the flight-size behavior (31). To see that the model is minimal, it is enough to remark that both conditions in $\mathcal{T}_i(t)$ are necessary. ■

In order to characterize the ACK-retaining mode, the ACK-buffer (33) has to be adjoined to the protocol model (20), resulting in an augmentation of the state of the user model. The protocol behavior depends on the measurements $\mu_i(\varkappa_t)$, which are functions of the overall network state $\varkappa_t$; this state is discussed in more detail in Section VIII.

We are now in a position to define user operators from (32).

Definition 22: The user operator $\mathcal{U}_i(w_i) : \mathbb{R}_+ \to \mathbb{R}_+$ mapping the ACK-flow $\phi(u_i^-, t)$ to the sending flow $\phi(u_i^+, t)$ is given by

$$\phi(u^+, t) = \mathcal{U}(w)\phi(u^-, t) \tag{34}$$

where $\mathcal{U}(w) = \text{diag}_i \{\mathcal{U}_i(w_i)\}$ and $\mathcal{U}_i$ is given in (20).

VIII. General Network Model

Modular and independent models for transmission channels, buffers, and users have been developed in Sections V–VII, respectively. In this section, we summarize the obtained results in a compact form involving dynamical systems and operators, and properties of the model are discussed. Notably, correspondence of the proposed model with existing ones is emphasized/recalled.

A. General Model

The general network model takes the form

$$\begin{aligned}\dot{\varkappa}(t) &= \mathcal{N}\left(\varkappa_t, \phi(u^-,t), \phi(b^-,t)\right)\\ \digamma_i(C_i,t) &= \int_{\mathscr{B}_{C_i}(t)}^{t} \phi(u_i^+,s)ds\end{aligned} \tag{35}$$

with

$$\Phi(t) = \begin{bmatrix} 0 & 0 & \mathcal{U}(w) & 0\\ 0 & 0 & 0 & \mathcal{B}\\ 0 & \mathcal{R}_{ub} & 0 & 0\\ \mathcal{R}_{bu} & \mathcal{R}_{bb} & 0 & 0\end{bmatrix}\Phi(t) + \begin{bmatrix}0\\0\\ \hline 0\\ \mathcal{D}_b\end{bmatrix}\delta(t) \tag{36}$$

where $\Phi(t) = \mathrm{col}(\phi(u^+,t), \phi(b^+,t), \phi(u^-,t), \phi(b^-,t))$ and $\varkappa = \mathrm{col}(\tau, z, \pi)$ are the flows and the state of the network, respectively. The hybrid models for user and queue dynamics are described by the nonlinear (discontinuous) functional $\mathcal{N}$ obtained from (7), (8), (20), and (33). The notation $\varkappa_t$ is here to emphasize that the evolution of the network state depends on past state values [23]. Note that adjoining the flight-size expression is needed to obtain a finite number of equilibrium points. Indeed, since the user flow is computed from the derivative of the flight size, the equilibrium information is lost and can only be recovered from the original expression of the flight size. At equilibrium, we indeed have $\digamma_i^* = w_i^* = \mathrm{RTT}_i^* \phi_i^*$, where RTT_i^* and ϕ_i^* are equilibrium values for RTT and the sending flow of user u_i, respectively.

B. Model Approximations

The proposed framework includes explicit and seemingly exact expressions for every quantity of interest, but this was not historically the case. The sending rate model has always been a missing link in past formulations where *ad hoc* models were considered. It has been recently shown in [9] and [22] that these flow models are actually approximations of the W-ACK-clocking model, which is turn an approximation of the FS-ACK clocking model considered in this paper. We summarize these remarks below for completeness.

1) Ratio Flow Model: By making the approximation $\digamma_i(t) \simeq w_i(t)$ in the FS-ACK-clocking model to get the W-ACK clocking model, we obtain

$$w_i(t) = \int_{\mathscr{B}_{C_i}(t)}^{t} \phi(u_i^+,s)ds \tag{37}$$

and using the right-square rectangle rule, we get the following expression for the sending rates:

$$\phi(u_i^+,t) \approx \frac{w_i(t)}{T_i + \tau(g^i(t))} \tag{38}$$

which is very similar to the ratio flow model for instance considered in [7], [8], and [11].

2) Joint Flow Model: The joint flow model reuses the W-ACK-clocking model and by making a first-order Taylor expansion on the implicit expression

$$w_i(t) - \int_{t}^{\mathscr{F}_{C_i}(t)} \phi(u_i^+,s)ds = 0 \tag{39}$$

we obtain

$$w_i(t) + (T_i + \tau(t+T_i^f))(\dot{w}_i(t) - \phi(u_i^+,t)) \approx 0 \tag{40}$$

or equivalently

$$\phi(u_i^+,t) \approx \frac{w_i(t)}{T_i + \tau(t+T_i^f)} + \dot{w}_i(t) \tag{41}$$

which is exactly the joint flow model considered in [9] and [19]–[21]. Neglecting the derivative term yields the usual ratio model [7], [8], [11].

3) Static Model: The static link model assumes that sending rates are proportional to the derivative of congestion window sizes, making the relation between these sizes and queuing delays static. This model can be obtained by further approximating the ratio model or using linearization and a (0, 0) Padé approximation. In Section IX-B, a more general proof for the static-link model is provided, and it suggests that the static model has a much wider domain of validity, as experimentally emphasized in [24].

IX. Single-Buffer/Multiple-User Topology With Delay-Based Protocols

The purpose of this section is twofold: exemplify the modeling technique on a simple topology and prove that when some conditions on the topology are met, the proposed model reduces to a model involving a static-link model [24]. The proposed model hence allows to clarify the status of the static-link model [24] by providing, for the first time, a mathematical proof for its domain of validity.

To this aim, a single-buffer/multiple-user topology connected by lossless transmission channels is considered. The forward and backward propagation delays of user u_i are denoted by T_i^{f} and T_i^{b}, respectively. We propose to use the following generic model of any delay-based congestion control protocol as the user model:

$$\begin{aligned}\dot{z}_i(t) &= \mathcal{P}_i(z_i(t), \tau(g^i(t)))\\ w_i(t) &= \mathcal{W}_i(z_i(t), \tau(g^i(t)))\end{aligned} \tag{42}$$

where z_i, $w_i(t)$, $T_i = T_i^{\mathrm{f}} + T_i^{\mathrm{b}}$, and $g^i(t) = g(t - T_i^{\mathrm{b}})$ are the state of the protocol, the congestion window size, the propagation delay, and the backward delay operator, respectively. The functions $\mathcal{P}_i$ and $\mathcal{W}_i$ are defined as in (20).

A. Multiple-User/Single-Buffer Topology Model

The general model is given by (35) with (42) and

$$\begin{aligned}\dot{\tau}(t) &= \begin{cases} c^{-1}\sigma(t) + \delta(t) - 1, & \text{if } \mathcal{C}(t)\\ 0, & \text{otherwise}\end{cases}\\
\dot{\pi}_i(t) &= \begin{cases} 0, & \text{if } \mathcal{T}_i(t)\\ \dot{w}_i(t) + \phi(u_i^-,t), & \text{otherwise}\end{cases}\\
\digamma_i(t) &= \int_{g^i(t)-T_i^{\mathrm{f}}}^{t} \phi(u_i^+,\theta)d\theta\\
\phi(u_i^+,t) &= \begin{cases} \dot{w}_i(t) + \phi(u_i^-,t), & \text{if } \mathcal{T}_i(t)\\ 0, & \text{otherwise}\end{cases}\\
\phi(u_i^-,t) &= \begin{cases} \frac{c\phi(u_i^+, g^i(t)-T_i^{\mathrm{f}})}{c\delta(g^i(t)) + \sum_j \phi(u_j^+, g^i(t)-T_j^{\mathrm{f}})}, & \text{if } \mathcal{C}(t)\\ \phi(u_i^+, t - T_i^{\mathrm{b}} - T_i^{\mathrm{f}}), & \text{otherwise}\end{cases}\\
\sigma(t) &= \sum_{i=1}^{N} \phi(u_i^+, t - T_i^{\mathrm{f}})\end{aligned} \tag{43}$$

where $\delta(t)$ denotes the normalized cross-traffic $\delta(t) \in [0,1)$. The topology of this model is completely described by (36) with the operators $\mathcal{B}$, $\mathcal{U} = \text{diag}_{i=1}^{N}\{\mathcal{U}_i\}$, $\mathcal{R}_{ub} = \begin{bmatrix} \nabla_{T_1^\text{b}} & \cdots & \nabla_{T_N^\text{b}} \end{bmatrix}^T$, $\mathcal{R}_{bb} = 0$, and $\mathcal{R}_{bu} = \begin{bmatrix} \nabla_{T_1^\text{f}} & \cdots & \nabla_{T_N^\text{f}} \end{bmatrix}$, where ∇_d is the constant delay operator with delay $d > 0$.

B. Homogeneous Delays and No Cross-Traffic—The Static-Link Model

As stated in Section VIII-B.3, the static-flow model can be obtained using various approximations, which suggests that the static-model is only valid locally. This, however, contradicts the results reported in [24], where it is emphasized that the static model may yield quite precise results over a wide domain. Note also that the static-link model has been invalidated in many scenarios, notably some involving very heterogeneous delays or cross-traffic; see, e.g., [9] and examples of Section X. In the following, we show that *the static model can be exact when some conditions on the network topology are met.*

Result 23: According to the model proposed in (43), the static-link model given by

$$\tau(t) = c^{-1}\sum_i w_i(t - T^\text{f}) - T \tag{44}$$

is exact in the single-link topology whenever:

- the buffers are permanently congested, i.e., $\mathcal{C}(t)$ holds true for all $t \geq 0$;
- the propagation delays are homogeneous, i.e., $T_i^\text{f} = T^\text{f}$, $T_i^\text{b} = T^\text{b}, i = 1,\ldots,N$;
- the cross-traffic is absent, i.e., $\delta \equiv 0$;
- the users are not in ACK-retaining mode, i.e., $\mathcal{T}_i(t)$ holds true for all $i = 1,\ldots,N$.

Proof: Assuming in (43) that the propagation delays are homogeneous, i.e., $T_i^\text{f} = T^\text{f}$, $T_i^\text{b} = T^\text{b}$, $i = 1,\ldots,N$, that there is no cross-traffic, i.e., $\delta \equiv 0$, that the buffer is always congested (i.e., $\mathcal{C}(t)$ holds true for all t), and all users are active (i.e., the $\mathcal{T}_i(t)$'s are all true), we obtain, after the substitution of sending flows in queue dynamics, that $\dot{\tau}(t) = c^{-1}\sum_i \dot{w}_i(t - T^\text{f})$. After integration, we get the model (43), where we have assumed $\tau(0) = 0$ and $w_i(0) = 0, i = 1,\ldots,N$. The additional constant term $-T$ is determined using the fact that $\sum_i w_i^*(T + \tau^*) = c$ at equilibrium. ■

Compared to the justification of this model in [22], the proof developed above is much more insightful since no model approximation is made, only assumptions on the network topology. This shows that the static-link model has an application domain that is much wider than the ratio-link and the joint-link models, when the conditions on the topology are met. It is indeed valid in the nonlinear setting, and it does not result from any approximation, just assumptions on the topology.

This also shows that the proposed metamodel is able to provide theoretical justifications of a simpler model. This supplies a way for deriving proofs for validity domain of models.

Note also that the above result might be generalizable to the case of chained buffers and multiple users. The exactness of the static model over more complex topologies is an open question.

C. Homogeneous Delays and Cross-Traffic

When cross-traffic is added to the problem, the overall picture changes. The cross-traffic acts as a bandwidth limiter both in the networking and control terminology. Indeed, a nonzero $\delta(t)$ reduces the maximal output capacity c, creating then sort of "varying output capacity" $c(1 - \delta(t))$, which reduces the bandwidth perceived by users.

Result 24: In the congested mode, the queue model writes

$$\begin{aligned} \dot{\tau}(t) &= c^{-1}\sum_i \dot{w}_i(t - T^\text{f}) + \delta(t) - \varphi(t) \\ \varphi(t) &= \frac{c\delta(g_{bf}(t))}{c\delta(g_{bf}(t)) + \sum\limits_j \phi_j(u_j^+, g_{bf}(t) - T^\text{f})} \end{aligned} \tag{45}$$

where $g_{bf}(t) = g(t - T^\text{b} - T^\text{f})$ and $\varphi(t)$ is the output flow at time t corresponding to the cross-traffic. The term $\varphi(t)$ is responsible for the bandwidth reduction.

Proof: Since the buffer is always congested and the users are not in ACK-retaining mode, we have

$$\phi_i(u_i^+, t) = \dot{w}_i(t) + \phi_i(u_i^-, t) \tag{46}$$

and

$$\phi_i(u_i^-, t) = \frac{c\phi_i(u_i^+, g_b(t) - T^\text{f})}{c\delta(g_b(t)) + \sum\limits_j \phi_j(u_j^+, g_b(t) - T^\text{f})}. \tag{47}$$

Substituting the above expressions in the queue model and noting that $\sum_j \phi_j(u_j^-, t - T^\text{f}) + \varphi(t) = c$ yields the result. ■

X. Model Validation

We consider the scenarios given in [9] and [22] to validate the proposed model. The results obtained via NS-2 have been slightly shifted in time so that the congestion window variation times match. Unlike [22], the results are not shifted in amplitude, and this causes small discrepancies. If, however, the NS-2 results were shifted vertically so that they match initial equilibrium values, then the curves would match almost perfectly. The simulations are performed on a nonoptimized MATLAB/Simulink implementation of the model. Comparisons to NS-2 in terms of accuracy and execution times are made even though a fair comparison should have been made with a C/C++ implementation of the proposed model.

A. Single Buffer/Multiple Users

In this section, we consider the interconnection of two users through a single resource, as depicted in Fig. 7. The bottleneck has capacity $c = 100$ Mb/s, and the packet size including headers is 1590 B. The following scenarios from [9] and [22] are considered.

- Scenario 1: The congestion window sizes are initially $w_1^0 = 50$ and $w_2^0 = 550$ packets, at 3 s w_1 is increased to 150 packets. The propagation delays are $T_1 = 3.2$ ms and $T_2 = 117$ ms for users 1 and 2, respectively; see Fig. 8.
- Scenario 2: The congestion window sizes are initially $w_1^0 = 210$ and $w_2^0 = 750$ packets, at 5 s w_1 is increased to 300 packets. The propagation delays are $T_1 = 10$ ms and $T_2 = 90$ ms for users 1 and 2, respectively; see Fig. 9.

We can see that our results fit well with the ones obtained by packet level (NS-2) simulations. Yet, this is not the case for the ratio-link, static-link, and joint-link models when the transient state is considered. The obtained results by the proposed model are identical to the results given in [9] and [22] and obtained using the W-ACK-clocking model. This is expected since the W-ACK clocking is an approximation of the FS-ACK-clocking model defended in this paper. Note also that the approximation condition related to the window size increase is satisfied here.

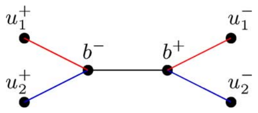

Fig. 7. Topology of scenarios 1 and 2.

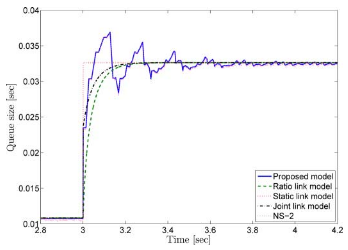

Fig. 8. Scenario 1: queue size.

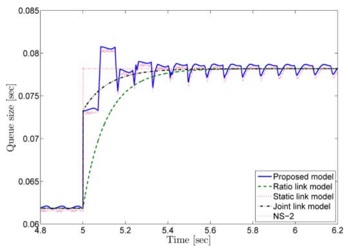

Fig. 9. Scenario 2: queue size.

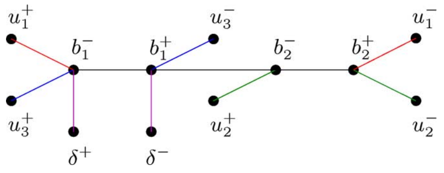

Fig. 10. Topology of scenarios 3–6 (δ represents cross-traffic I/O nodes).

Considering the scenario 1, the proposed model simulates in 91 s, whereas NS-2 in 26 s, with a 0.1-ms time-step.

B. Multiple Buffers/Multiple Users

Here, we consider the case of two buffers interconnected in series (see Fig. 10) with capacities $c_1 = 72$ Mb/s and $c_2 = 180$ Mb/s. The packet size including headers is 1448 B. The link propagation delays are 20 ms for link 1 and 40 ms for link 2. The total round-trip propagation delays are $T_1 = 120$ ms, $T_2 = 80$ ms, and $T_3 = 40$ ms for sources 1–3, respectively. Initially, the congestion window sizes are $w_1^0 = 1600$ packets, $w_2^0 = 1200$ packets, and $w_3^0 = 5$ packets. The following scenarios from [9] and [22] are considered.

- Scenario 3: No cross-traffic, and the congestion window w_1 is increased by 200 packets at 10 s; see Fig. 11.
- Scenario 4: No cross-traffic, and the congestion window w_2 is increased by 200 packets at 10 s; see Fig. 12.

The proposed model is again able to capture the network behavior well, and it retrieves the previous results reported in [9] and [22]. This is again due to the equivalence between the ACK-clocking models in this case. Considering the scenario 3, the proposed model simulates in 129 s, whereas NS-2 in 87 s, with a 0.1-ms time-step.

We introduce now a constant cross-traffic $x_{c1} = c_1/2$ on the first link. Initially,[6] we set $w_1^0 = 1200$, $w_2^0 = 1600$, $w_3^0 = 5$, and we consider the following scenarios.

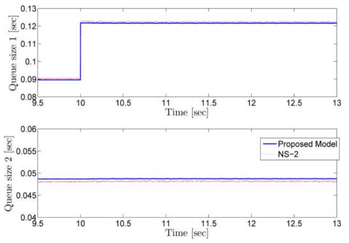

Fig. 11. Scenario 3: *(top)* queue 1 and *(bottom)* queue 2.

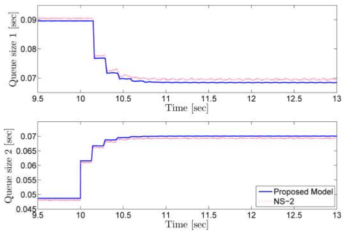

Fig. 12. Scenario 4: *(top)* queue 1 and *(bottom)* queue 2.

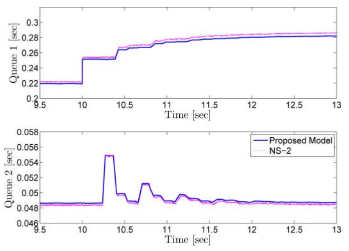

Fig. 13. Scenario 5: *(top)* queue 1 and *(bottom)* queue 2.

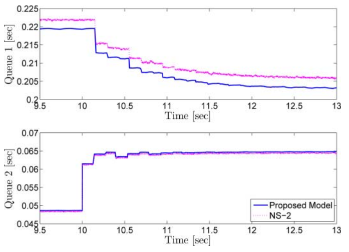

Fig. 14. Scenario 6: *(top)* queue 1 and *(bottom)* queue 2.

- Scenario 5: The congestion window w_1 is increased by 200 packets at 10 s; see Fig. 13.
- Scenario 6: The congestion window w_2 is increased by 200 packets at 10 s; see Fig. 14.

The obtained results are identical to the results obtained by the NS-2 simulations. Notice the reaction time between the moment at which the congestion window size is increased and the moment at which the second queue sees the flow variation. This illustrates that the model captures well the communication path and the order of elements (spatial and temporal topology). These characteristics are not directly visible in the simulation results given in [22] since the curve steps seem to have been aligned on the same temporal cursor.

Despite results equivalence, the proposed metamodel enjoys interesting properties such as modularity and scalability, which the model reported in [9] and [22] lacks. This is an important improvement over previous models that were not able to cumulate accuracy, scalability, modularity, and other interesting properties. This will be discussed in more detail in Section XI.

[6]This scenario is actually identical to the one in [22, Section III.B.3]; the initial values for congestion window sizes given in [22] are incorrect.

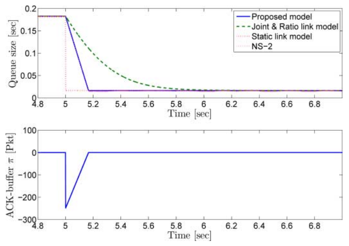

Fig. 15. Scenario 7: *(top)* queue size and *(bottom)* ACK buffer.

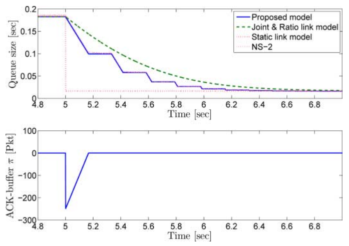

Fig. 16. Scenario 8: *(top)* queue size and *(bottom)* ACK buffer.

C. Decreasing the Congestion Window Size

The models proposed in [18] and [22] do not capture sudden decreases in the congestion window size that would cause the buffer to empty or become smaller than the actual flight size, that is, smaller than the number of packets in flight. The proposed model does capture these phenomena since: 1) the FS-ACK-clocking model derived from the conservation law involves the flight size rather than the congestion window size, unlike in [22]; and 2) the user model implements an ACK-buffer to count the number of packets to remove from the network before starting to send again. Note that the derivation of the user model including the ACK-buffer has been made possible due to the availability of an explicit expression for the flow of acknowledgments (29). This makes it computable through an explicit solution for the queuing delay and the buffer output flows [17], [25]. In [9], the decreasing of congestion window size is handled by adding a rate limiter to constrain the (negative) slope of the queue size. This rate limiter is, however, rather difficult to characterize due to the time-varying nature of the lower bound on the slope that depends on the received rate of acknowledgment and the network state, the former being unfortunately unavailable in the framework of the thesis [9].

Let us consider the single-user/single-buffer case where the total propagation delay is $T = 150$ ms, the packet size including headers is 1040 B, and the initial value of the congestion window size is $w^0 = 500$. At $t = 5$ s, the congestion window size is halved. We consider the following scenarios.

- Scenario 7: $c = 12.5$ Mb/s and no cross-traffic; see Fig. 15.
- Scenario 8: $c = 25$ Mb/s and half capacity used by cross-traffic; see Fig. 16.

We can see that we obtain exactly the same results as NS-2 simulations (and the rate-limiter model reported in [9]). As desired, the ACK-buffer measures (counts) the number of packets to remove before starting to send again. Considering the scenario 8, the proposed model simulates in 157 s, whereas NS-2 in 19 s, with a 0.1-ms time-step.

XI. Related Work

Despite being unorthodox, it was to the authors' point of view to discuss the related works after presenting the proposed metamodel. Many congestion models have been proposed in the literature with different formalisms: time domains, protocols, network topologies, and other hypotheses. It is thus quite difficult to compare them directly and give a thorough discussion. Yet, we have identified some important characteristics that are summarized in Tables I–III. Table I collects global characteristics of a network model such as the *time-domain* property to indicate whether the model is in discrete (*DT*) or continuous (*CT*) time, or the *modularity* property to indicate whether it is modular or not. With *delay*, we indicate that the model considers it either as constant (*Cst*), time-varying (*TV*), state-dependent (*SD*), or asynchronous (*Async*) discrete-time system whose asynchrony (incorporating delays) depends on the state of the system. *Spatial* and *temporal* topologies denote whether the model shares the same spatial and time topologies with the actual network, respectively. *Generic* indicates the ability of the model to describe the network for different protocols, whereas *rate model* denotes from which formula the user sending rate is computed. The *ratio* flow model [7], [8], [11], [14], [15] is given by quotient of the congestion window size and the RTT. The *static* flow model [24] assumes that the flow is proportional to the derivative of the congestion window size, and the *joint* flow model [9], [19]–[21] assumes that the flow is given by the sum of the ratio-flow model and the congestion window size derivative.

Tables II and III examine the important subparts of the model, namely the user and buffer models, respectively. It is also important to mention that none of them has the structure of a metamodel having the precision of the considered one. In Table II, *homogeneous* and *heterogeneous* delays denote how well a model captures the case of homogeneous or heterogeneous propagation delays, whereas *cross* and *bursty* traffic denote how well a model captures the presence of cross-traffic and bursty traffic [26], respectively. *Exact* indicates that the transient-state behavior of a model matches well to the one observed in packet-level simulations. *Too fast* and *too slow* mean that a model converges to steady state faster or slower, respectively, compared to what is observed in packet-level simulations. *Faster* states a relativity faster convergence, yet not as much as observed when it is *too fast*. In Table III, *output flows* indicates whether it is defined as *aggregate*, meaning all input flows are mixed, or *split*, where the buffer acts as a square multiple-input–multiple-output (MIMO) system mapping a given input flow to a given output flow. *FIFO characterization* denotes whether the model captures the actual content of the queue and the FIFO behavior. With *queuing delay model*, we indicate whether its effect on input flows is clearly defined. *Frequency filtering effect* denotes whether the model of buffer has a filtering effect on the input flows to the output flows.

A. Table I

[1]In this model, while the delay is denoted as a function of time, it is implicitly defined from the state, without further investigation. An exact delay characterization as a function of the state has been obtained in [17]. [2]The state-dependent delay is implicitly characterized by the ACK-clocking formula and the RTT expression. [3]The modularity of the model could have

TABLE I
MAIN CHARACTERISTICS OF SOME EXISTING MODELS IN THE LITERATURE

Network Models	[27]	[7]	[11]	[28]	[24]	[9] (DAE)	[29]	[15]	[30]	[22]	This paper
Characteristics											
Time-domain	DT	CT	CT	CT	CT	DT	CT	Hybrid	DT	CT	Hybrid
Delay	Cst	Cst	TV/SD[1]	Cst	Cst	Async.	Cst	Cst	Async.	SD[2].	SD
Modular	Almost[3]	Yes	No	Yes	Yes	No	Yes	Almost[4]	No	Almost[5]	Yes
Spatial topology	Yes	No	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Temporal topology	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Generic	Yes	No	No	No	No	No	No	No	Yes	Yes	Yes
Rate model	No[6]	Ratio	Ratio	Ratio	Static	(FS)ACK[7]	Joint	Ratio	No[6]	(W)ACK[8]	(FS)ACK[9]

TABLE II
COMPARISON OF DIFFERENT USER FLOW MODELS WITH RESPECT TO THE ACCURACY OF ESTIMATED QUEUE SIZES

	User sending flow models	Static	Ratio	Joint	(W)ACK	(FS)ACK+ ACK-Buffer[10]
Congestion window	**Characteristics**					
Increase	Homogeneous delays	Exact/Too fast[11]	Too slow[14]	Faster[12]	Exact	Exact
	Heterogeneous delays	Too fast	Too slow[14]	Faster[12]	Exact	Exact
	Cross-traffic	Too fast[13]	Too slow[14]	Faster[12]	Exact	Exact
	Bursty-traffic	Exact/Too fast[15]	Too slow[14]	Faster[16]	Exact	Exact
Decrease	Homogeneous delays	Too fast[17]	Too slow[14]	Too slow[14]	Too fast[17]	Exact
	Heterogeneous delays	Too fast[17]	Too slow[14]	Too slow[14]	Too fast[17]	Exact
	Cross-traffic	Too fast[17]	Too slow[14]	Too slow[14]	Too fast[17]	Exact

TABLE III
COMPARISONS BETWEEN DIFFERENT BUFFER DESCRIPTIONS

Buffer descriptions	Aggregate model (7)-(8)	Pseudo-queue model [15], [18]	Flow model [13], [16], [17]
Characteristics			
Output flows	Aggregate	Split	Split
FIFO characterization	Lost[18]	No[19]	Yes[20]
Queuing delay model	Lost[18]	No[21]	Yes [22]
Frequency filtering effect	All-pass/undef.[23]	All-pass/Low-pass[24]	All-pass [25]

been developed, but was not. [4]Modular but not able to represent any topology, essentially due to lack of any output flow model for buffers. [5]This modeling technique can be applied to any type of topology, however hand calculations, which result in poor scalability, are necessary. [6]The notion of flow is not defined in a discrete-time model. [7]The discrete-time model is derived from a continuous-time model using the (FS)ACK-clocking model. [8]The flow is implicitly defined as the solution of the (W)ACK-clocking model. [9]The flow is computed by solving the (FS)ACK-clocking model explicitly.

B. *Table II*

[10]This model is the one introduced in [25] and also considered in the current paper. [11]In the single-buffer case, the model is exact. It is unclear whether this is also true for the multiple buffer case. In most cases, the model is too fast. [12]Accurate modeling of discontinuity/high slope of the queue when burst of data arrives at the queue input. [13]The model is usually too fast. It might, however, be possible to refine it to account for constant cross-traffic [19]. [14]Unable to characterize the high slope of the queue due to the low-pass filtering effect of the resulting queue model; see Section VI-D. [15]Exact in the case of homogeneous delays and single-buffer topology since the relation is static and it has an infinite bandwidth; see Section IX-B. Too fast otherwise. [16]Larger bandwidth than the ratio model, but still limited since the bandwidth reduces as the queue size grows. [17]These models may withdraw packets from the network by injecting negative flows.

C. *Table III*

[18]The information is lost since the output flows are aggregate. [19]The output flows are insensitive to content swapping. [20]The output flows are sensitive to any swap of information in the queue. [21]The exact queuing delay is actually difficult to assign to any output flow due to the structure of the model for the output flows involving queues. [22]Undefined due to the aggregate formalism. [23]All-pass when the sum of the input-flows does not exceed the maximal output capacity, otherwise the filtering effect is not really defined due to the aggregate formalism. [24]All-pass when the sum of the input-flows does not exceed the maximal output capacity, otherwise acts as a low-pass filter with bandwidth inversely proportional to the queue size. [25]This is due to the direct-feedthrough structure of the model. The delay and nonlinear formula for the output flows only have a compressing/expanding effect on time and amplitude.

XII. Conclusion

This paper presents a modular fluid-flow network congestion control model to analyze communication networks with arbitrary topology. This modular metamodel is introduced by mathematically modeling network elements such as queues, users, and transmission channels and network performance indicators such as sending/acknowledgment rates and delays. It is composed of building blocks that implement local mechanisms, some of which are being ignored/unmodeled by some existing models in the literature. It is shown that the proposed model allows to recover existing models and brings a formal proof for their validity/invalidity as well as for their domain of validity. We present a novel classification of the previously proposed models in the literature, and we show that the existing models are often not capable of capturing the transient behavior of the network precisely. Numerical results obtained from packet-level simulations demonstrate the accuracy of the proposed model. We plan to extend our work with modeling of data loss, such as packet drops, and the timeout mechanism at the user level.

Acknowledgment

This work is dedicated to the memory of the authors' colleague Prof. U. T. Jönsson, who unfortunately passed away before he could see it completed.

References

[1] V. Jacobson, “Congestion avoidance and control,” *Comput. Commun. Rev.*, vol. 18, no. 4, pp. 314–329, 1988.
[2] R. Srikant*, The Mathematics of Internet Congestion Control*. Boston, MA, USA: Birkhäuser, 2004.
[3] S. Low, F. Paganini, and J. C. Doyle, “Internet congestion control,” *IEEE Control Syst.*, vol. 22, no. 1, pp. 28–43, Feb. 2002.
[4] D. Mitra, “Stochastic theory of a fluid model of producers and consumers coupled by a buffer,” *Adv. Appl. Probab.*, vol. 20, no. 3, pp. 646–676, 1988.
[5] R. Johari and D. Tan, “End-to-end congestion control for the Internet: Delay and stability,” Statistical Laboratory, University of Cambridge, Cambridge, U.K., Tech. Rep., 2000.
[6] C. Hollot, V. Misra, D. Towsley, and W. B. Gong, “A control theoretic analysis of RED,” in *Proc. 20th IEEE INFOCOM*, Tel-Aviv, Israel, 2001, pp. 1510–1519.
[7] G. Vinnicombe, “On the stability of networks operating TCP-like congestion control,” in *Proc. 15th IFAC World Congress*, Barcelona, Spain, 2002, pp. 217–222.
[8] F. Pagaganini, J. C. Doyle, and S. Low, “A control theoretical look at Internet congestion control,” in *Multidisciplinary Research in Control*, L. Giarré and B. Bamieh, Eds. Berlin, Germany: Springer, 2003, vol. 289, Lecture Notes in Control and Information Sciences, pp. 17–31.
[9] K. Jacobsson, “Dynamic modeling of Internet congestion control,” Ph.D. dissertation, KTH School of Electrical Engineering, Stockholm, Sweden, 2008.
[10] D. V. Lindley, “The theory of queues with a single server,” *Math. Proc. Cambridge Philos. Soc.*, vol. 48, pp. 277–289, 1952.
[11] V. Misra, W. Gong, and D. Towsley, “Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED,” in *Proc. ACM SIGCOMM*, Stockholm, Sweden, 2000, pp. 151–160.
[12] G. Vinnicombe, “On the stability of end-to-end congestion control for the Internet,” University of Cambridge, Cambridge, U.K., Tech. rep. CUED/F-INFENG/TR 398, 2000.
[13] C. Ohta and F. Ishizaki, “Output processes of shaper and switch with self-similar traffic in ATM networks,” *IEICE Trans. Commun.*, vol. E81-B, no. 10, pp. 1936–1940, 1998.
[14] J. Hespanha, S. Bohacek, K. Obraczka, and J. Lee, “Hybrid modeling of TCP congestion control,” in *Hybrid Systems: Computation and Control*, M. Di Benedetto and A. Sangiovanni-Vincentelli, Eds. Berlin, Germany: Springer, 2001, vol. 2034, Lecture Notes in Computer Science, pp. 291–304.
[15] D. Färnqvist, K. Strandemar, K. H. Johansson, and J. P. Hespanha, “Hybrid modeling of communication networks using modelica,” in *Proc. RVK*, 2002.
[16] Y. Liu, F. L. Presti, V. Misra, D. F. Towsley, and Y. Gu, “Scalable fluid models and simulations for large-scale ip networks,” *Trans. Model. Comput. Simul.*, vol. 14, no. 3, pp. 305–324, 2004.
[17] C. Briat, H. Hjalmarsson, K. H. Johansson, G. Karlsson, U. T. Jönsson, and H. Sandberg, “Nonlinear state-dependent delay modeling and stability analysis of Internet congestion control,” in *Proc. 49th IEEE Conf. Decision Control*, Atlanta, GA, USA, 2010, pp. 1484–1491.
[18] Y. Zhang, Y. Xiang, S. Liu, and D. Loguinov, “Queuing dynamics and single-link stability of delay-based window congestion control,” *Comput. Netw.*, vol. 54, pp. 1543–1553, 2010.
[19] N. Möller, “Window-based congestion control,” Doctoral dissertation, KTH, Stockholm, Sweden, 2008.
[20] K. Jacobsson, L. Andrew, A. Tang, S. Low, and H. Hjalmarsson, “An improved link model for window flow control and its application to FAST TCP,” *IEEE Trans. Autom. Control*, vol. 54, no. 3, pp. 551–564, Mar. 2009.
[21] K. Jacobsson, L. Andrew, A. Tang, K. H. Johansson, H. Hjalmarsson, and S. Low, “ACK-clock dynamics: Modeling the interaction between ACK-clock and network,” in *Proc. 27th IEEE INFOCOM*, Phoenix, AZ, USA, 2008, pp. 181–185.
[22] A. Tang, L. L. H. Andrew, K. Jacobsson, K. H. Johansson, H. Hjalmarsson, and S. H. Low, “Queue dynamics with window flow control,” *IEEE/ACM Trans. Netw.*, vol. 18, no. 5, pp. 1422–1435, Oct. 2010.
[23] J. K. Hale and S. M. V. Lunel*, Introduction to Functional Differential Equations*. New York, NY, USA: Springer-Verlag, 1991.
[24] J. Wang, D. X. Wei, and S. H. Low, “Modelling and stability of FAST TCP,” in *Proc. 28th IEEE INFOCOM*, 2005, pp. 938–948.
[25] C. Briat, H. Hjalmarsson, K. H. Johansson, G. Karlsson, U. T. Jönsson, H. Sandberg, and E. A. Yavuz, “An axiomatic fluid-flow model for congestion control analysis,” in *Proc. 50th IEEE Conf. Decision Control*, Orlando, FL, USA, 2011, pp. 3122–3129.
[26] H. Jiang and C. Dovrolis, “Why is the Internet traffic bursty in short time scales?,” in *Proc. ACM SIGMETRICS*, 2005, pp. 241–252.
[27] R. Johari and D. K. Tan, “End-to-end congestion control for the Internet—Delays and stability,” *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 818–832, Dec. 2001.
[28] F. Paganini, Z. Wang, J. C. Doyle, and S. Low, “Congestion control for high performance, stability, and fairness in general networks,” *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 43–56, Feb. 2005.
[29] A. Tang, K. Jacobsson, L. L. H. Andrew, and S. H. Low, “An accurate link model and its application to stability analysis of FAST TCP,” in *Proc. 26th IEEE INFOCOM*, Anchorage, AK, USA, 2007, pp. 161–169.
[30] R. Shorten, F. Wirth, and D. Leith, “A positive systems model of TCP-like congestion control: Asymptotic results,” *IEEE/ACM Trans. Netw.*, vol. 14, no. 3, pp. 616–629, Jun. 2006.

Corentin Briat, photograph and biography not available at the time of publication.

Emre Altug Yavuz, photograph and biography not available at the time of publication.

Håkan Hjalmarsson, (M’98–SM’11–F’13) photograph and biography not available at the time of publication.

Karl Henrik Johansson, (S’92–M’98–SM’08–F’13) photograph and biography not available at the time of publication.

Ulf T. Jönsson, photograph and biography not available at the time of publication.

Gunnar Karlsson, (S’85–M’89–SM’99) photograph and biography not available at the time of publication.

Henrik Sandberg, photograph and biography not available at the time of publication.