Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

FLUID-FLOW AND PACKET-LEVEL MODELS OF DATA NETWORKS UNIFIED UNDER A MODULAR/HIERARCHICAL FRAMEWORK: SPEEDUPS AND SIMPLICITY, COMBINED

Matías Bonaventura
Rodrigo Castro

Departamento de Computación
FCEyN, UBA and ICC, CONICET
Ciudad Universitaria, Pabellón 1
Buenos Aires, C1428EGA, ARGENTINA

ABSTRACT

As network technologies undergo an exponential growth in terms of bandwidth and topology complexity, the gap is worsened between the performance of network simulation techniques and real network scenarios. Fluid-flow models for network dynamics are a well know option for reducing simulation overhead while offering useful averaged approximations of network metrics. Yet, the methods and tools established in the packet-level simulation community are alien to those used in continuous system modeling by means of differential equations. This hinders the synergy between specialists in both techniques. In this work, we present a novel modeling methodology and simulation tool to unify the experience of designing network simulation models both with fluid-level and packet-level techniques under a single modular and hierarchical formal framework. We verified the efficacy of our approach both in terms of simulation speedups and modeling simplicity for canonical network simulation scenarios.

1 INTRODUCTION AND MOTIVATION

Packet-level simulation has always represented an important tool to help designing and evaluating new network topologies and protocols. Yet, it is well known that there exist issues in simulation performance scalability when the complexity of the network grows (either due to topology complexity, throughput intensity, or a combination of both). Such issues often impose a limitation in the quality and/or time-to-delivery of the answers that can be obtained via simulation. Network technologies have witnessed an exponential growth in terms of the bandwidth (Ethernet Alliance 2015) and topology size (from massive clusters to the Internet itself). This situation increases the gap between the performance capabilities of current network simulation techniques and real-world networks.

The network modeling and simulation (M&S) community has dealt with simulation performance in several different ways (Fernandes 2017), such as simulation parallelization, coarse- vs. fine-grained models, fluid-flow vs. packet-level abstractions, and hybrid (fluid and packet) simulation approaches (Ngangue Ndih and Cherkaoui 2015), to name a few. Each strategy imposes its particular limitations and bring about new problems. For instance, topology splitting and model distribution for parallel simulation can offer gains only in cases where there is light inter-subnetworks traffic, meanwhile clock synchronization techniques can also present technical difficulties (Riley et al. 1999).

In this work, we propose novel methods for efficient modeling and simulation for fluid-flow network approximations. A higher abstraction level represents averaged (fluid) packet data rates, instead of resorting to a packet-by-packet approach. The behavior of the network is represented by analytical models that can be much faster to solve numerically and yield results with acceptable accuracy when compared to fine-grained packet-level simulations. There inevitably exists a trade of accuracy for speed: in fluid approximations only the first order moment (mean value) is preserved for probabilistic sequences of discrete events (i.e. of packet

arrivals and departures from network nodes). Several fluid-flow approximations have been proposed and refined incrementally since the early days of packet network modeling (Kleinrock 1969). We are interested in a more general kind of fluid models, those that can be represented by sets of Ordinary Differential Equations (ODEs). Such continuous dynamic systems can be captured by the DESS (Differential Equation System Specification) formalism within Systems Theory (Zeigler et al. 2018). They have been successfully applied to study complex end-to-end dynamics in TCP data flows, initiated in Padhye et al. (1998) and Liu et al. (2003) continued in this work.

We identify, though, several limitations in the existing methods and tools for ODE-based fluid-flow network simulation. At the heart of the fluid approximation approach is the need of verifying results incrementally and iteratively against packet-level simulations. Unfortunately, existing methods and tools for network simulation are of a very different nature than those required to solve ODEs. In network M&S the natural approach is to compose topologies modularly, through the interconnection (links, queues) of network nodes (hosts, routers, etc.) each one embedding particular behavior (discrete event algorithms) to deal with streams of packets. In ODE numerical solving, different tools (e.g. Matlab, Octave, SciPy, etc.) provide algorithms that require the specification of equations in forms that are alien to network descriptions. The modeler is then left with the task of implementing a packet-level model, inspect the network topology and discrete behavior of its nodes, come up with a set of approximating ODEs, encode them in a separate tool (or develop a custom ODE solver), simulate both systems separately, and compare results.

We claim that this approach is heavyweight, error prone, and hinders true synergy between specialists at the discrete and continuous domains of network M&S. The network modeler needs to be well acquainted with ODEs, numerical solving methods and their correct implementation. These are well studied topics but are generally not part of the knowledge of network experts and designers. Also developing and maintaining customized code is time-consuming and error-prone for ODE experts. In this paper, we present a modular and scalable integrated approach to combine the modeling and numerical solving of fluid network models along with their packet-level counterparts under a unified and consistent mathematical description and practical tool. Modularity provides the modeler with the ability to graphically interconnect self contained models of network elements that can embed either a packet-level algorithm or its fluid-flow approximation, depending on the task at hand. In the case of fluid models, the overall set of ODEs gets automatically defined and ready to solve under a discrete event-based framework. Basic network nodes can then be reused to create arbitrary, possibly complex topologies, without the need to manually redefine a new set of ODEs for each new simulation scenario. We shall rely on the Discrete EVent Systems Specification (DEVS) (Zeigler et al. 2018) as the underlying M&S formalism, capable of representing discrete event, discrete time and continuous dynamics combined in a mathematically sound way. DEVS models are block-based units of self-contained behavior, that can be interconnected through input/output ports to create modular and hierarchical topologies of blocks, matching very closely the topology-oriented paradigm of real data networks. Within the DEVS realm, ODEs are solved using Quantized-State Systems (QSS) methods (Kofman and Junco 2001).

This paper is organized as follows. Section 2 presents the formal frameworks, methods and tools. Section 3 introduces the proposed fluid equations and modular blocks to represent fluid network topologies. Section 4 uses three case studies to asses the modeling advantages of the proposed approach, and verifies the accuracy trade off and performance gains. Section 5 presents conclusions and future lines of research.

2 BACKGROUND AND RELATED WORK

2.1 Packet-Level Network Simulation

Various commercial and academic simulators model in great detail several aspects of data networks (Wehrle et al. 2010). Modeled network entities rely on different algorithms to advance simulation in a discrete-event fashion and exchange messages between models. In general, they take a *packet-by-packet* approach where messages are sent from one model to another whenever a real data packet would be sent thought the different

network layers. Models implement control logic and protocols very closely to the algorithms implemented in real software and hardware, yielding results comparable to real data networks. Few packet-level network simulators are supported by formal simulation methods that guarantee correctitude.

In this work, we use PowerDEVS (Bergero and Kofman 2011), a discrete event simulator that implements the DEVS mathematical formalism (Zeigler et al. 2018). PowerDEVS offers a data network library providing different blocks (atomic and coupled DEVS models) and has been extensively used for packet-level simulation (Castro and Kofman 2015). PowerDEVS was extended recently (Laurito et al. 2017) to generate automatically large topologies based on network descriptions (e.g. those provided by SDN controllers) and used to simulate high-speed networks at CERN (Bonaventura et al. 2016).

2.2 Fluid-Flow Network Simulation

Packet-level simulation provides fine-grained details at the cost of high execution times, making them unsuitable for large high-speed networks. Higher abstraction level models trade speedups for coarser-grained accuracy: several *fluid-flow models* have been proposed to describe *averaged* network dynamics through a set of ODEs. The performance of numerically solving fluid models grows only linearly with the number of nodes and is independent of link speeds.

Misra, Gong, and Towsley proposed a set of ODEs (Misra et al. 2000), which we refer to as the MGT reference model, to describe the behavior of TCP over a network of routers implementing Random Early Detection (RED). MGT was later extended with NewReno and SACK versions of TCP (Liu et al. 2003), short-lived TCP sessions (Marsan et al. 2005), explicit congestion notification (Kunniyur and Srikant 2003), and parallelization techniques using GPUs (Liu et al. 2014).

Even when fluid-flow models are a feasible solution to simulate accurately and efficiently large data networks, they are still far from being adopted by network experts. The particular set of ODEs that represent a network is defined by the topology and the characteristics of each node. These equations should be written in specific formats for numerical packages or, for more flexibility and performance, classical ODE solving methods are coded. The relationship between a given set of ODEs and a network topology might not be clear, forcing network experts to understand and learn to specify differential equations. Moreover, for experts in ODEs and numerical methods the effort to develop and test new network models is twofold. On the one hand, to solve ODEs they need to code and maintain software to translate topologies into equations, possibly implement ODE-solvers for particular challenges as Delay Differential Equations (DDE), and in some cases custom simulations cycles are developed (Liu et al. 2003). On the other hand, to verify new models they still need to develop packet-level models in a completely different software.

2.3 DEVS and QSS for Network Simulation

DEVS is capable of representing any type of discrete system and approximating continuous systems with controlled accuracy, providing a suitable formal framework for representing both packet-level and fluid-flow data models. DEVS atomic and coupled models encapsulate behavior and allow systems to be built in a modular and hierarchical way. PowerDEVS also implements a family of QSS methods to solve ODEs whose behavior can be exactly represented by a DEVS model (Kofman and Junco 2001). In contrast to classical methods that discretize time, QSS discretizes the state variables, resulting in a discrete-event approximation of a continuous system (instead of the classic discrete-time approximation). In particular, for solving network fluid models it is relevant the development of QSS Delay Differential Equations (DDEs) (Castro et al. 2011) which allows dynamically changing delay (such as the one experienced by network traffic) to be applied by reusing an already implemented DEVS atomic model. A salient property for network fluid models is the asynchronous nature of QSS. With classical discrete time methods equations describing uncongested links or inactive hosts take as much computing time to solve as congested links or active hosts. While special preprocessing techniques were suggested to remove inactive nodes (Liu

et al. 2003), alternating busy periods cannot be handled. Instead, QSS will balance dynamically computing efforts in network nodes only when needed.

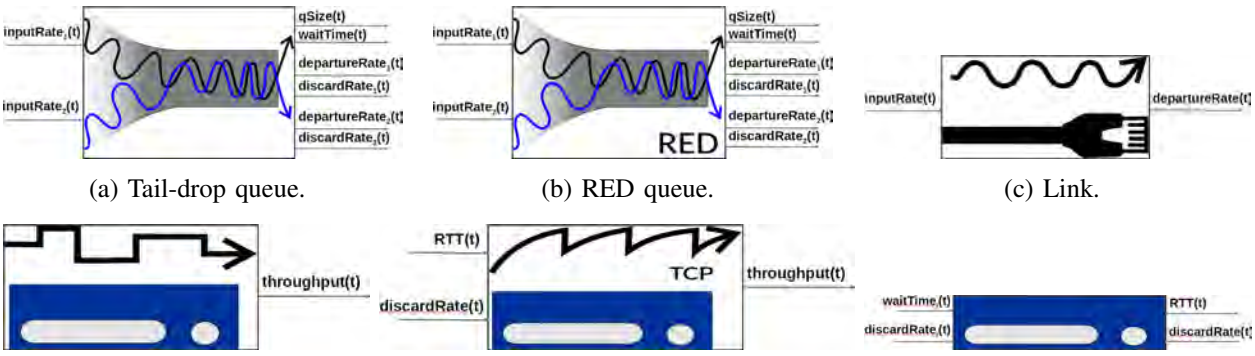
In this work, a subset of the MGT fluid-flow equations is simulated within a unified mathematical formalism and tool (DEVS/QSS and PowerDEVS respectively) used also for the packet-level simulation. The focus is on the M&S techniques, and not on the accuracy of the MGT model (discussed in the above mentioned references). Yet, few modifications are proposed for MGT to better represent discontinuities (easily handled by QSS) and experimental results are shown to validate our approach.

In Castro and Kofman (2015), an early version of the MGT model was implemented using QSS, where a single flow class is considered and equations describe the dynamic of a single queue. Here, we extend this work allowing for complete topologies to be modeled for multiple flows, and providing graphical modular blocks to encapsulate the dynamics of different network elements. We leverage the modularity of QSS/DEVS to encapsulate the ODE numerical solving complexities, providing new fluid-flow network models that resemble closely the packet-level models.

3 MODULAR APPROACH FOR FLUID-FLOW NETWORK MODELING

This section introduces a new library of fluid-flow models, shown in Figure 1, that provides the building blocks to compose fluid network topologies. Each block contains the implementation of a subset of the ODEs governing the dynamic behavior of the complete system. Similarly to the packet-level approach, where DEVS models encapsulate the behavior of network entities, our fluid-flow approach uses DEVS models to encapsulate subsets of ODEs (representing queues, links, hosts, etc.). The final full set of ODEs describing the fluid network gets automatically expressed by the interconnection of pre-defined building blocks (in the same way packet-level topologies get fully defined by interconnecting packet-oriented blocks).

The proposed approach decouples three domains of knowledge required to produce fluid simulations: a) description of the network (network modeling), b) implementation of ODEs (ODE modeling) and c) ODE numerical solvers (ODE simulation). Each domain requires specific knowledge and should be ideally performed by different experts. In traditional approaches these aspects are tightly coupled requiring network experts to acquire knowledge in continuous dynamic systems, while ODE experts need to code topologies and solvers. Conversely, with the technology presented in this work, network experts can graphically specify fluid network topologies without knowledge on ODEs, using blocks from a pre-defined fluid library. Continuous dynamic equations are defined using block-oriented PowerDEVS diagrams, which can be tested independently and easily interchanged with other equivalent network elements. ODE experts can focus on better expressing network dynamics, without concerning on other issues such as topological information, simulation techniques or numerical solvers. The ODEs numerical solving is handled by the QSS simulator engine which can switch between available solvers without changing the equation's block diagrams.



(a) Tail-drop queue. (b) RED queue. (c) Link.

(d) Basic host. (e) TCP sender. (f) TCP receiver.

Figure 1: Library of graphical blocks for fluid-flow network entities encapsulating ODE dynamics.

3.1 Fluid-Flow Equations for a Tail-Drop Queue

The queue block implements a FIFO (First In First Out) tail-drop queue, providing an external interface via its input/output ports and a set of parameters as described in Table 1. Figure 1a shows the queue block implemented in PowerDEVS, with 2 input flows. The outgoing signals are those governed by the set of ODEs (1)-(3) explained below. Buffer size will grow/shrink according to the difference between the input and output rates. As in MGT, the evolution of the queue size uses a differential version of Lindley equation:

$$\frac{dqSize(t)}{dt} = \sum_i inputRate_i(t) - C \qquad (Q_{min} \leq qSize(t) \leq Q_{max}). \tag{1}$$

We highlight the boundary conditions $Q_{min} \leq qSize(t) \leq Q_{max}$ which impose sharp discontinuities on the evolution of the differential equation. In MGT no upper bound is considered so tail-drop behavior can not be property captured, and the lower bound is guaranteed by a special $-1_q(t)$ function which is solved numerically using an iterative backward calculation (Mao and Petzold 2002). A novel QSS *bounded* integrator was developed, shown in Figure 2a, where the detection of discontinuities is simple and efficient due to the inherent discrete-event nature of QSS methods (Bergero and Kofman 2011).

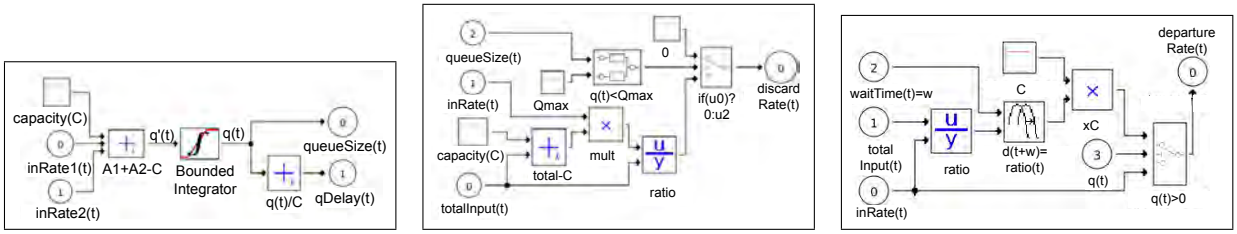
Table 1: Block interface for the RED and tail-drop queue models.

	Name	Signal/Parameter	Description
Input Ports	**Input Rate**	$inputRate_i(t)$	N ports with the input rate of each incoming flow denoted by the subindex $i \in \{1,...,N\}$.
Output Ports	**Queue Size**	$qSize(t)$	Evolution of the queue size as described by (1)
	Wait Time	$waitTime(t)$	Time a packet arriving at time t remains enqueued waiting to be served. Calculated as $qSize(t)/C$
	Departure Rate	$departureRate_i(t)$	N outports with the departure rate for each i$-th$ incoming flow. Calculated as in (3)
	Discard Rate	$discardRate_i(t)$	N outports with the discard rate for each i$-th$ incoming flow. Calculated as in (2)
Configuration Parameters	**Buffer Size Limits**	Q_{max}, Q_{min}	Upper and lower limits for $qSize(t)$. $Q_{min} < Q_{max}$ where typically $Q_{min} = 0, Q_{max} \in \mathbb{R} > 0$
	Capacity	C	Service capacity of the outgoing link of the queue
	Drop Prob. Function	$t_{min}, t_{max}, p_{max}$ (only RED)	In RED queues, definition for the drop probability function in as defined in (5)
	Queue Size Estimate	$\alpha \in [0,1]$, $\delta \in \mathbb{R}$ (only RED)	In RED queues, weight and sampling rate for the exponentially weighted moving average (4). δ can be statically set to $1/C$ or dynamical to $1/inputRate(t)$

Equation (2) models packets dropped for each i-th input flow. Packet discards occur only when the buffer is full ($qSize = Q_{max}$) and its rate equals the difference between the service capacity C and the sum of all input rates. The total discard rate is shared among all flows proportionally to their current input rate:

$$discardRate_i(t) = \begin{cases} 0 & qSize(t) < Q_{max} \\ \frac{inputRate_i(t)}{\sum_j inputRate_j(t)} (\sum_j inputRate_j(t) - C) & qSize(t) = Q_{max} \end{cases}. \tag{2}$$

Equation (3) models packets served by a router for each input flow. When the buffer is empty there is no queuing wait time so the departure rate will equal the arrival rate. When the buffer is not empty, the router sends at full capacity sharing the bandwidth among incoming flows proportional to their input rates:



(a) Queue Size Equation (1). (b) Discards Equation (2). (c) Departure Equation (3).

Figure 2: Block-oriented representation of queue equations as implemented in PowerDEVS.

$$departureRate_i(t + waitTime(t)) = \begin{cases} inputRate_i(t) & qSize(t) = 0 \\ \frac{inputRate_i(t)}{\sum_j inputRate_j(t)} C & qSize(t) > 0 \end{cases} . \tag{3}$$

It is important to note the delay nature of equation (3) and the aspects for its numerical approximation. Incoming packets at time t will be served forward in time after a dynamically changing queuing $waitTime(t)$, as depicted in the left hand-side of the equation. This transforms the equation into a DDE, imposing particular requirements to the numerical solvers. In other studies this delay is not considered (Misra et al. 2000). In our work, DDEs are solved straightforwardly with a modular QSS-DDE block (Castro et al. 2011) (see Figure 2c).

3.2 Modular Addition of AQM Schemes to the Tail-Drop Queues

Modern routers implement AQM schemes to control the buffer size by dropping packets before tail-drops occur. Different AQM schemes have been modeled as fluid equations, e.g. RED and PI controller (Castro and Kofman 2015; Liu et al. 2003). The RED queue block shown in Figure 1b provides the same input/output ports as the tail-drop queue (refer to Table 1). RED behavior is defined in Equations (4)-(8) and modeled as a separated block, reusing parts of the tail-drop queue behavior, allowing the modeler to switch easily between different AQM policies and enabling topologies with heterogeneous queues (not possible in MGT).

$$\frac{d\widehat{qSize}(t)}{dt} = \frac{log_e(1-\alpha)}{\delta}\widehat{qSize}(t) - \frac{log_e(1-\alpha)}{\delta} qSize(t), \tag{4}$$

$$discardProbRED(t) = \begin{cases} 0 & 0 \le \widehat{qSize}(t) \le t_{min} \\ \frac{\widehat{qSize}(t) - t_{min}}{t_{max} - t_{min}} p_{max} & t_{min} \le \widehat{qSize}(t) \le t_{min} \\ 1 & t_{max} \le \widehat{qSize}(t) \end{cases} , \tag{5}$$

$$RED.qSize(t) = queue.qSize(t), \tag{6}$$

$$queue.inputRate_i(t) = inputRate_i(t) * (1 - RED.discardProbRED(t), \tag{7}$$

$$queue.discardRate_i(t) = queue.discardRate_i(t) + inputRate_i(t) * RED.discardProbRED(t). \tag{8}$$

The discard probability is calculated based on 2 functions as specified in RFC2309. The differential equation (4) models the exponentially weighted moving average $\widehat{qSize}$. Equation (5) models the classic RED discard probability function $discardProbRED$. Packets discarded by RED do not contribute to the buffer size (6) and increase the total discard rate (8). In MGT, RED discards are not taken into account in the buffer size calculation. Liu et al. (2014) considers either tail-drops or RED discards, but not both.

3.3 Fluid-Flow Basic Hosts

Hosts are data flow sources that provide input to the queues. In basic sources, shown in Figure 1d, the *throughput* does not depend on external signals. Basic sources might be useful in early stages of network design when traffic pattern specifications are vague. Hosts' traffic can be represented with simple continuous functions, such as step-wise constant rates (to represent UDP hosts), oscillatory or trapezoidal (for regular data rates), or other customly shaped functions of time. Later, they can be easily replaced by more complex blocks. Some examples are given in Section 4. This type of traffic is not considered in MGT.

3.4 Modular Construction of Fluid-Flow Topologies

The link block, Figure 1c, represents the physical cable that interconnect hosts and queues, and imposes bandwidth and propagation delays. Bandwidth delay is modeled by the queue capacity and the host send rate. Propagation delay is modeled in (9) adding a fixed *propDelay* delay to the input signal. Host, link and queue blocks are connected to build network topologies as specified in the following equations:

$$link.departureRate(t) = link.inputRate(t - propDelay), \tag{9}$$

$$link_1.inputRate(t) = host.throughput(t), \tag{10}$$

$$queue_j.inputRate_i(t) = link_j.outputRate(t), \tag{11}$$

$$link_j.inputRate(t) = queue_{j-1}.departureRate_i(t). \tag{12}$$

where $L = (link_1, ..., link_n)$ and $Q = (queue_1, ..., queue_{n-1})$ are, respectively, the ordered sets of links and queues which the flow traverses along the network. $1 \leq j \leq n$ is the hop index along the path and $i \in \mathbb{N}_{>0}$ is the index at which the flow enters each of the queues $queue_j \in Q$. The host *thoughput* output is connected to the first link which then connects to the first queue in the flow path. Then, the *departureRate* output of each queue is connected with the input of the next queue with an intermediary link. Graphical topology examples showing the interconnection of blocks are depicted in Figure 3.

3.5 Fluid Equations for Hosts with Congestion Control

We consider hosts with throughput controlled by the TCP Reno protocol where dynamics of the data sending rate depend on the network state. The TCP host block interface is described in Table 2 and shown in Figure 1e. We model a bulk transfer TCP scenario with unlimited data to send based on the MGT equations:

$$throughput(t) = W(t) * N/RTT(t), \tag{13}$$

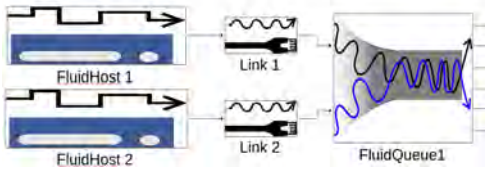
$$\frac{dW(t)}{dt} = \frac{1}{RTT(t)} - \frac{W(t)}{2} discardRate(t - RTT(t)) \qquad (1 \leq W(t) \leq W_{max}).$$

The throughput is calculated based on the TCP window size W, the round trip time RTT, and the total number of TCP sessions N. The TCP window size models additive increase in the first term, increasing by one every round trip time, and multiplicative decrease in the second term, halving the window according to the discard rate. TCP senses packet losses after approximately one RTT, so the discard rate is considered with a dynamic $RTT(t)$ delay transforming the equation into a DDE with the same considerations as discussed earlier for Equation (3). Also, the sharp discontinuities introduced by the maximum and minimum size of the window take the same considerations as discussed earlier for Equation (1). TCP hosts are connected with the rest of the network as follows (in addition to Equations (9)-(12)):

$$link_1.inputRate(t) = TcpHost.throughput(t), \tag{14}$$

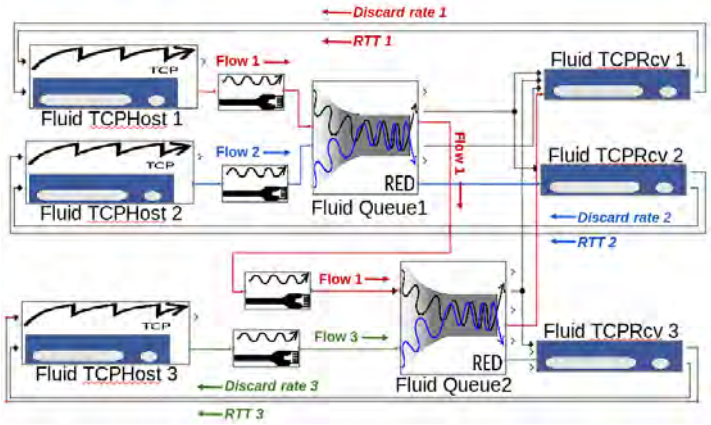
$$TcpHost.RTT(t) = \sum_{link \in L} link.propDelay + \sum_{queue \in Q} queue.waitTime(t), \tag{15}$$

$$TcpHost.discardRate(t) = \sum_{queue \in Q} queue.discardRate_i(t), \tag{16}$$



(a) Fluid topology for Experiment 1.

(b) Packet-level topology for Experiment 1.



(c) Fluid topology for Experiment 3.

Figure 3: Topologies for Experiments in Section 4.

where L, Q and i are interpreted as in (9)-(12), which in this case includes the return path delay for ACKs, and discards of ACKs can be considered negligible. As the basic hosts, the TCP *thoughput* output is connected to a link. The RTT and $discardRate$ input signals are, respectively, the accumulation of delay and discards that the flow experiences when traversing each of the queues and links in its path. These equations are encapsulated in the TcpReceiver block shown in Figure 1f to improve topology visualization.

Table 2: Block interface for the TCP host.

	Name	Signal/parameter	Description
Input Ports	**Round Trip Time**	$RTT(t)$	Delay experienced by the flow to traverse the network and back. as defined by (15)
	Discard Rate	$discardRate(t)$	Total packet drops rate experienced by the flow as defined by (16)
Output Ports	**Throughput**	$throughput(t)$	Send rate as defined by (13)
Configuration Parameters	**Maximum Window**	W_{max}	Maximum allowed TCP Window Size, usually 65535 bits
	#Sessions	N	Number of TCP sessions within the host

4 MODELING AND SIMULATION CASE STUDIES

Three experiments of increasing complexity are proposed to evaluate the new approach in terms of modeling benefits. We verify that simulations yield results qualitatively comparable to the literature on the MGT model. Fluid-flow models are compared against their counterpart packet-level models, both types implemented in PowerDEVS (models available upon request). Finally, a performance analysis is provided.

4.1 Modeling of Case Study Topologies

The first experiment uses 2 basic hosts connected to a single bottleneck tail-drop queue. The topology for the fluid-flow and packet-level models are shown in Figures 3a and 3b, respectively. The second experiment demonstrates the TCP block in a single RED queue scenario, replacing basic hosts with TCP hosts, the tail-drop queue with a RED queue, and TCP receivers and return links are added. The third experiment demonstrates TCP hosts with several sessions traversing interconnected queues. Figure 3c shows the fluid-flow topology composed by 3 TCP hosts and 2 RED queues.

The packet-level topology is analogous to the fluid topology (not shown). In all cases, topologies for both the fluid and the packet-level models look very similar. The network modeler drags and drop (or code) fluid- or packet-based blocks, and interconnects them to build full topologies. This is done within the same tool and requires no prior knowledge about the internal implementations, thus decreasing the learning curve. Conversely, in most existing fluid approaches, topology gets implicitly defined by the set of ODEs, hiding away their graphical view. Also, pre-defined blocks provide different versions for network elements (RED and tail-drop queues, basic and TCP hosts), which can be easily interchanged fostering lightweight experimentation with different alternatives.

4.2 Experiment 1: Network Queue with Basic Hosts

The first topology demonstrates the basic UDP hosts and the tail-drop queue block (Equations 1-3). The 2 UDP inputs rates follow a linear and sinusoid trajectories and the queue is set up with a 600Kb buffer, a link speed of 100Kb, and 0 propagation delay. The packet-level model uses stochastic packet sizes (not considered in the fluid model) following a normal probabilistic distribution with $mean = 1000Kb$ and $var = 500Kb$. Experiments 2 and 3 are also configured with the same stochastic packet size.

Figure 4 compares the simulation results of the fluid and packet-level models. The fluid model approximation shows excellent concordance with the packet-by-packet simulation. The fluid model follows the averaged behavior and, as expected, does not capture the packet-size variance. The fluid tail-drop queue provides an excellent approximation to the packet-level queue even in the presence of input noise.

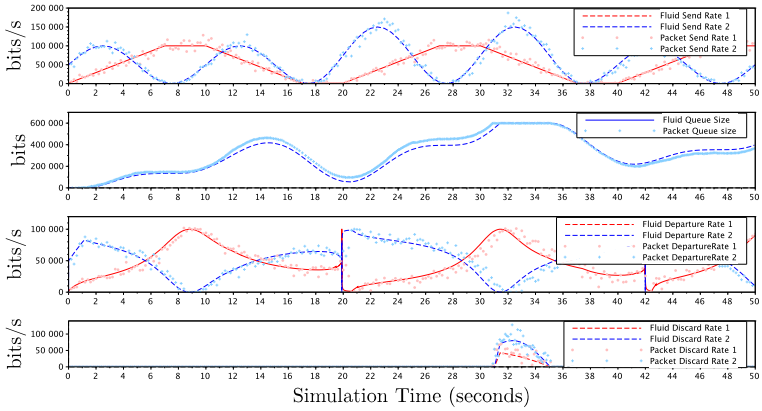


Figure 4: Packet-level and fluid-flow simulations: tail-drop queue and UDP hosts.

4.3 Experiment 2: Single TCP Sessions

The second topology is configured with 5Mbps links, 10ms propagation delay, and the following parameters for the RED queue: 400Kb buffer, $t_{min} = 300Kb$, $t_{max} = 400Kb$, $p_{max} = 0.1$, and $\alpha = 1E-3$. The first host begins with 2 TCP sessions ($N = 2$), then after 15s the second hosts starts another TCP session ($N = 1$), and finally after 30s the first host closes all connections. The continuous model is set to start at 1.3s to skip the initialization phase which is not modeled.

Figure 5 compares qualitatively the following metrics: TCP window size (in packet-level: average over all sessions in a host), hosts' throughputs, buffer sizes (effective and RED estimate), and departure for each flow. As reported for the MGT model, the fluid-flow approximation is meant to capture the averaged operation of the system. The plot shows that the ODE system produces the same behavioral profile as the packet-level system. In both models, fluid- and packet-level, it is observed that when the number of sessions change TCP adapts to share fairly the bottleneck link. As expected, the packet-level simulation shows a bigger variance, but the fluid approximation follows similar dynamics resembling the averaged

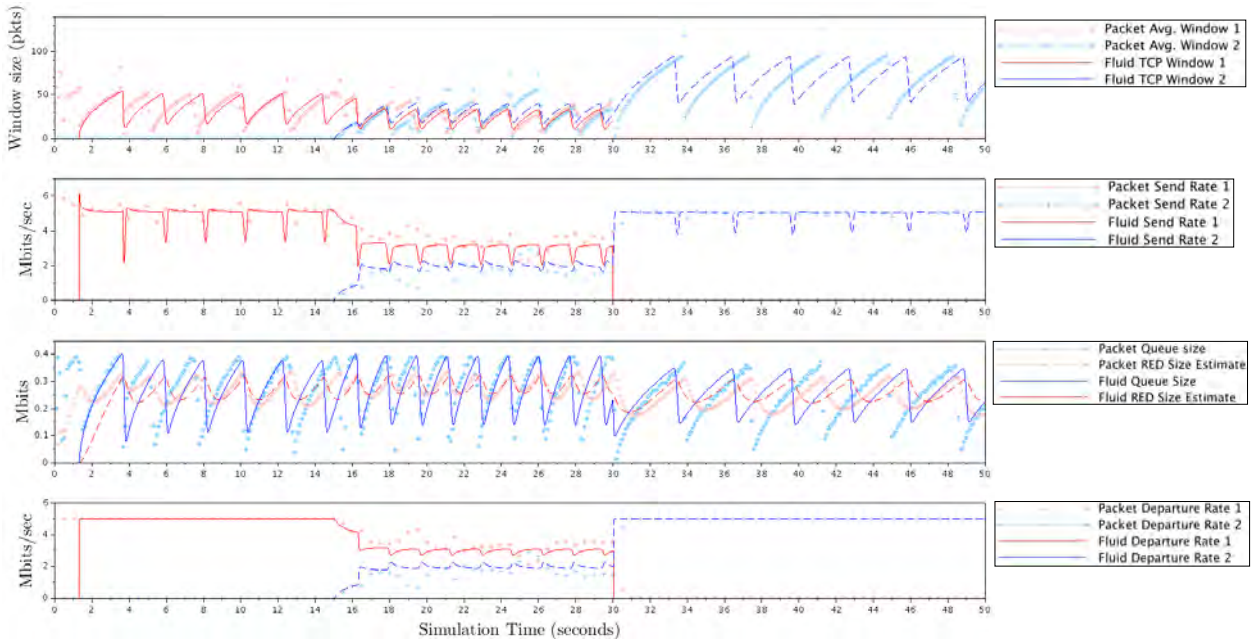


Figure 5: Comparison of packet-level and fluid simulations for Experiment 2.

behavior. There is a small phase shift between the two results probably due to fast-retransmit/fast-recovery (FRFR) not being modeled in the fluid TCP block. The fluid RED queue follows a similar path as the packet-level queue both for effective and estimated buffer sizes (again with a phase shift).

The results verify that the dynamics of the new fluid TCP and RED queue blocks yield results comparable to those reported for the MGT model. The fluid equations approximate the mean values of network metrics such as latency, buffer sizes, link utilization, etc. and follow similar dynamics as the packet-level metrics.

4.4 Experiment 3: Multiple TCP Sessions and Interconnected Queues

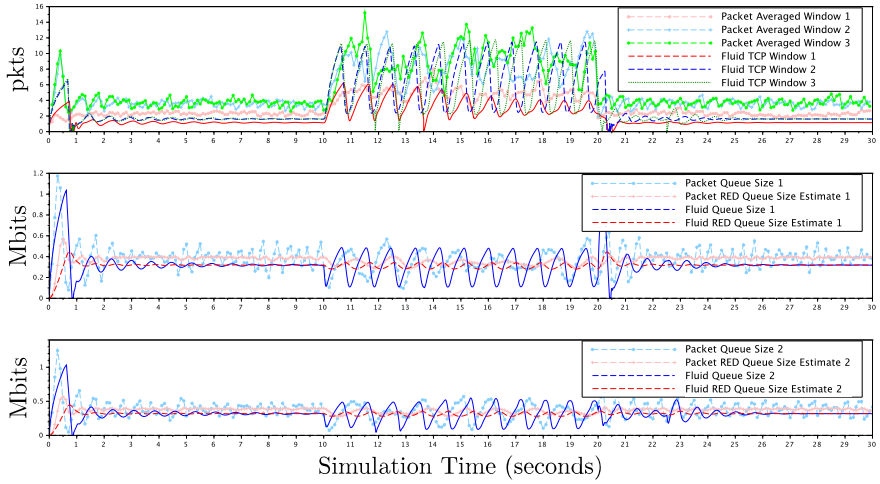
For the third topology, queues and links are configured as in the previous example. Traffic from the first host traverses both queues, while traffic from host 2 and 3 only traverses queue 1 and 2 respectively.

Simulation starts with 40 TCP sessions per host (120 in total), after 10s sessions change to 10 per host, and after 20s all 40 sessions per host restart. Figure 6a compares TCP window and buffer sizes. Again, the fluid-flow model follows the expected averaged behavior and exhibits dynamics similar to the packet-level model which shows steeper peaks. With 120 TCP sessions, the averaged TCP window and buffer sizes stabilize around the RED t_{min},t_{max} values. When TCP sessions drop to 30, RED is unable to control the buffer size and the TCP window size oscillates. Fluid equations capture both transient and stable phases.

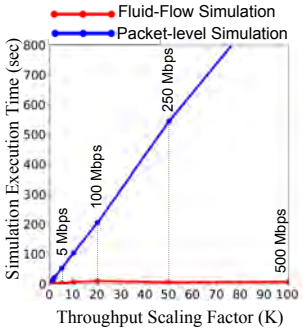
This experiment shows the qualitative accuracy of TCP equations for a larger number of connections. When using simple data patterns (such as UDP) the fluid queue yields very satisfactory results. When modeling more complex flows (such as TCP), as reported for the MGT model, the fluid model yield acceptable approximations. Fluid TCP captures relevant dynamics in send patterns and key features such as fair link sharing. Depending on the case study, the sacrificed accuracy is a price to pay for significant execution speedups. As the modular approach allows to easily replace and test different implementations of a same element, a new TCP host that models FRFR might yield better approximations.

4.5 Experiment 4: Simulation Performance Scalability Analysis

The goal is to verify that our modular approach (using QSS solvers for the fluid case) exhibits similar scalability results and performance advantages as reported by the MGT model (using discrete-time solvers).



(a) Comparison of packet-level and fluid simulations metrics.



(b) Simulation Execution Time.

Figure 6: Simulations results for Experiments 3 and 4: 2 RED queues and 3 TCP hosts with 120 sessions.

The model in Section 4.4 is scaled up by a factor of K, increasing proportionally the link bandwidth, buffer capacity, RED thresholds and number of TCP sessions. We set $\alpha = 1/K$ and use 3rd order QSS with accuracy parameters according to link speeds. Single threaded simulations are run for increasing values of K using PowerDEVS2.2 (Intel Corei7 3.40GHz, 8GB RAM). Figure 6b shows that simulation times for packet-level models scale linearly with link bandwidth (more packets required to saturate each link) while remaining almost flat for fluid models. We verified speedups of up to 200x for 1Gbps links.

5 CONCLUSIONS AND FUTURE WORK

We presented a novel modeling methodology and simulation tool to unify the experience of designing network simulation models both with fluid- and packet-level techniques. Under the DEVS M&S framework both types of models boil down to a common discrete event simulation, rely on a same mathematical framework, modeling methodology, and practical tool, thus reducing the learning curve and simplifying model description. Experimentation with canonical network scenarios corroborated that our models and tool indeed provide flat simulation times for fluid-based abstractions, compared to linearly increasing simulation times in the packet-based models. Meanwhile, the strategy of developing libraries of reusable, block-oriented and self-contained models proved successful: the visual design of network topologies can now be implemented almost indistinguishably from the representation of the network, be it fluid- or packet-based.

Next steps include: test larger topologies/higher-speed networks (like those modeled for the TDAQ network at CERN), evolve towards hybrid network simulation (interacting fluid- and packet-level models), and generate fluid models automatically (based on network descriptions available at SDN Controllers).

REFERENCES

Ethernet Alliance 2015. "Ethernet Roadmap". www.ethernetalliance.org/roadmap. Accessed: April 4, 2018.

Bergero, F., and E. Kofman. 2011. "PowerDEVS: a Tool for Hybrid System Modeling and Real-Time Simulation". *Simulation* 87(1-2):113–132.

Bonaventura, M., D. Foguelman, and R. Castro. 2016. "Discrete Event Modeling and Simulation-Driven Engineering for the ATLAS Data Acquisition Network". *Computing in Science & Engineering* 18(3):70–83.

Castro, R., and E. Kofman. 2015. "An Integrative Approach for Hybrid Modeling, Simulation and Control of Data Networks Based on the DEVS Formalism". In *Modeling and Simulation of Computer Networks and Systems: Methodologies and Applications*, Chapter 18. Morgan Kaufmann.

Castro, R., E. Kofman, and F. E. Cellier. 2011. "Quantization-Based Integration Methods for Delay-Differential Equations". *Simulation Modelling Practice and Theory* 19(1):314–336.

Fernandes, S. 2017. *Performance Evaluation for Network Services, Systems and Protocols*. Springer.

Kleinrock, L. 1969. "Models for Computer Networks". In *Proceedings of the IEEE International Conference on Communications*, 21/9–21/16. Boulder, Colorado.

Kofman, E., and S. Junco. 2001. "Quantized-State Systems: a DEVS Approach for Continuous System Simulation". *Transactions of The Society for Modeling and Simulation International* 18(3):123–132.

Kunniyur, S., and R. Srikant. 2003. "End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks". *IEEE/ACM Transactions on Networking (TON)* 11(5):689–702.

Laurito, A., M. Bonaventura, M. E. Pozo Astigarraga, and R. Castro. 2017. "TopoGen: A network Topology Generation Architecture with Application to Automating Simulations of Software Defined Networks". In *Proceedings of the 2017 Winter Simulation Conference*, edited by W. K. V. Chan et al., 1049–1060. Piscataway, New Jersey: IEEE.

Liu, J., Y. Liu, Z. Du, and T. Li. 2014. "GPU-Assisted Hybrid Network Traffic Model". In *Proceedings of the 2nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 63–74. ACM.

Liu, Y., F. Lo Presti, V. Misra, D. Towsley, and Y. Gu. 2003. "Fluid Models and Solutions for Large-Scale IP Networks". In *ACM SIGMETRICS Performance Evaluation Review*, Volume 31, 91–101. ACM.

Mao, G., and L. R. Petzold. 2002. "Efficient Integration Over Discontinuities for Differential-Algebraic Systems". *Computers & Mathematics with Applications* 43(1-2):65–79.

Marsan, M. A., M. Garetto, P. Giaccone, E. Leonardi, E. Schiattarella, and A. Tarello. 2005. "Using Partial Differential Equations to Model TCP Mice and Elephants in Large IP Networks". *IEEE/ACM Transactions on Networking* 13(6):1289–1301.

Misra, V., W.-B. Gong, and D. Towsley. 2000. "Fluid-Based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED". In *ACM SIGCOMM Computer Communication Review*, Volume 30, 151–160. ACM.

Ngangue Ndih, E. D., and S. Cherkaoui. 2015. "Simulation Methods, Techniques and Tools of Computer Systems and Networks". In *Modeling and Simulation of Computer Networks and Systems: Methodologies and Applications*, edited by O. M. et al. Morgan Kaufmann.

Padhye, J., V. Firoiu, D. Towsley, and J. Kurose. 1998. "Modeling TCP Throughput: A Simple Model and its Empirical Validation". *ACM SIGCOMM Computer Communication Review* 28(4):303–314.

Riley, G. F., R. M. Fujimoto, and M. H. Ammar. 1999. "A Generic Framework for Parallelization of Network Simulations". In *Proceedings of the 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 128–135. IEEE.

Wehrle, K., M. Günes, and J. Gross. 2010. *Modeling and Tools for Network Simulation*. Springer.

Zeigler, B. P., A. Muzy, and E. Kofman. 2018. *Theory of Modeling and Simulation 3rd Edition: Discrete Event and Iterative System Computational Foundations*. Elsevier.

AUTHOR BIOGRAPHIES

MATÍAS BONAVENTURA is a MASc in Computer Science, PhD student at the Universidad de Buenos Aires and Project Associate with the ATLAS TDAQ group at CERN. His research interests are hybrid modeling and simulation of networked systems. His email address is mbonaventura@dc.uba.ar.

RODRIGO CASTRO is Professor at the Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, head of the Simulation Lab, and Researcher at CONICET. His research interests include simulation and control of hybrid systems. His email address is rcastro@dc.uba.ar.