An Algorithmic Framework for Discrete-Time Flow-Level Simulation of Data Networks

Lasse Jansen [1,2] , Ivan Gojmerac [1] , Michael Menth [2] ,
Peter Reichl [1], and Phuoc Tran-Gia [2]

[1] Telecommunications Research Center Vienna (ftw.)
Donau-City-Str. 1, 1220 Vienna, Austria

[2] University of Würzburg, Institute of Computer Science
Am Hubland, 97074 Würzburg, Germany

{jansen,gojmerac,reichl}@ftw.at,
{menth,trangia}@informatik.uni-wuerzburg.de

Abstract. In this paper, we present a comprehensive algorithmic framework for discrete-time flow-level simulation of data networks. We first provide a simple algorithm based upon iterative equations useful for the simulation of networks with static traffic demands, and we show how to determine packet loss and throughput rates using a simple example network. We then extend these basic equations to a simulation method capable of handling queue and link delays in dynamic traffic scenarios and compare results from flow-level simulation to those obtained by packet-level simulation. Finally, we illustrate the tradeoff between computational complexity and simulation accuracy which is controlled by the duration of a single iteration interval Δ.

1 Introduction

Simulation has traditionally been an important tool for performance evaluation of data networks, mostly in the form of packet-level simulation by employing discrete-event simulation techniques [1]. Every packet arrival and departure at each link is modeled as a separate event. Although packet-level simulation still represents the most widely used approach, the simulation of today's networks with very high packet rates is often not feasible, as too many simulation events must be generated even for small intervals of simulated time.

However, in many cases the overhead of packet-level simulations is not necessary at all in order to achieve a realistic estimation of network statistics like throughput rates, queue sizes, or loss probabilities. In those cases, an efficient alternative to packet-level simulation is the simulation of networks at the level of individual flows, for which there exists a multitude of different techniques, commonly summarized under the terms *fluid simulation* or *flow-level simulation*.

In this paper, we concentrate upon *discrete-time flow-level simulation*. Traffic is not modeled in terms of discrete packets but rather in terms of a continuous amount of data. The data is shifted in fixed intervals Δ on predefined routes through the network. However, to our best knowledge, literature in this field of research lacks a discrete, easy-to-implement formulation of discrete-time flow-level simulation which is able to

model end-to-end connections. Addressing this issue, in this paper, we provide such a formulation, which additionally allows the network to be simulated at different levels of detail. We develop the fundamental flow-level simulation techniques step by step, first presenting a simple algorithm for throughput calculation, and then extending this algorithm to capture network dynamics like link and queueing delay.

The paper is structured as follows: Section 2 gives an overview of related work. In Section 3 we describe the basic equations for calculating the time-dependent aggregate throughput rates and the loss probabilities on the links, and we present a method for the calculation of the stationary network state in the presence of static traffic demands. Subsequently, in Section 4 we extend these basic equations to scenarios with dynamic traffic patterns by introducing queue and link delay modeling. Section 5 provides a comparison of flow-level and packet-level simulation results and demonstrates the influence of different durations of iteration interval Δ. Finally, Section 6 concludes the paper with summarizing remarks.

2 Related Work

This section provides a brief description of previous work and relevant applications of flow-level simulation. There are two main variants of flow-level simulation. The foundation for the *continuous-time* variant was given in [2] and [3], and has since been further developed and widely applied by other authors [4–7]. The basic principle of this approach is to model flow rates and rate changes without considering discrete data packets. Each flow is assigned a certain transmission rate, and rate reductions due to bottleneck links are tracked as events in the event chain of the simulator. Although widely used, under particular circumstances this approach has been shown to suffer from the so called *ripple effect* which can cause severe performance degradations concerning computation time. It occurs in networks with circularly overlapping end-to-end connections and overloaded links, causing rate change events to reproduce themselves in a circular manner [8, 9].

An alternative to the continuous-time variant is *time-stepped fluid simulation* which is the foundation of the simulation method presented in this paper. Here, the data is modeled in terms of quantities of a continuous amount of data which are shifted through the network at a fixed time step. This model was first proposed in [10], but the routing model applied therein is hardly applicable to realistic networks. In particular, a traffic demand matrix cannot be represented, as end-to-end connections have not been modeled at all.

Another approach which defines the evolution of network data streams in terms of differential equations is used in [11]. End-to-end connections are modeled as traffic aggregates which enable the simulation of realistic network and routing scenarios. The equations are numerically solved using the Runge-Kutta algorithm. In this paper, we use the basic idea from [10], and additionally enable end-to-end connections like in [11]. Furthermore, we provide a clean formulation of discrete-time flow-level simulation which is well suited for practical implementation.

The possibility of dynamic traffic rate adjustments and delay modeling in the presented simulation approach allows for multipath routing simulations in which traffic aggregates between individual pairs of nodes are carried via multiple parallel paths.

Implementing the simulation approach described in this paper, we have already evaluated the Adaptive Multi-Path algorithm (AMP) [12, 13] using our flow-level simulation environment (cf. [14, 15]).

Furthermore, the throughput and delay approximation capabilities of this approach enable more complex elastic traffic models, like e.g. TCP, to be integrated into such a flow-level simulation framework, as the sending rate of individual sources can be dynamically adjusted based upon the information about flow round-trip time (RTT) and loss probability on the path, employing differential equation models of TCP as presented in [11, 16, 17].

3 Basic Iterative Algorithm

In this section we first clarify the notation, after which we introduce the iterative algorithm that calculates throughput rates and loss probabilities. Finally, we show the application of this algorithm for the calculation of stationary throughput in a network with static traffic demands.

3.1 Definitions

The simulation process is based upon shifting data through the network in fixed time intervals Δ. In our model, a data rate λ denotes the total amount of data in a single iteration interval, normalized by Δ as the time between two successive iteration steps. In other words, a data rate of λ in interval Δ corresponds to an amount of data, $\lambda \cdot \Delta$, which may represent e.g. bits, bytes, or equally sized packets.

In the following we list some definitions for reference. Expressions indexed by $[n]$ are time-dependent and calculated in each iteration.

L	Set of all links.
A	Set of all aggregates.
$L^a \subseteq L$	Set of links crossed by aggregate $a \in A$.
$A_l \subseteq A$	Set of aggregates crossing link $l \in L$.
$first(a) \in L^a$	First link on the route of aggregate $a \in A$.
$next(a, l) \in L^a$	Successor link of $l \in L^a$ on the route of aggregate $a \in A_l$.
$sink(a) \in L^a$	Virtual link representing the sink of aggregate $a \in A$ and succeeding the last link of a.
c_l	Link capacity of link $l \in L$, i.e. data that can be served during a single iteration interval Δ.
$p_l[n]$	Loss probability at link $l \in L$.
$\lambda_l^a[n]$	Arrival rate of aggregate $a \in A$ at link $l \in L^a$.
$\lambda^a[n]$	Sending rate of aggregate $a \in A$, equal to $\lambda_{first(a)}^a[n]$.
$\lambda_l[n]$	Sum of all arrival rates at link $l \in L$.
$\theta^a[n]$	Throughput of aggregate $a \in A$.
Δ	Iteration duration interval, i.e. the time between two iteration steps.

3.2 Iterative Equations for Discrete-Time Flow-Level Simulation

We assume the network to be empty at initialization time ($n = 0$). Each traffic aggregate $a \in A$ has a specific route consisting of a sequence of links $l \in L^a$ along which the arrival rates $\lambda_l^a[n]$ of this aggregate are shifted in each iteration step. The overall arrival rate at link l in iteration step n is then:

$$\lambda_l[n] = \sum_{a \in A_l} \lambda_l^a[n]. \tag{1}$$

Based on this sum and the capacity c_l of this link, its loss probability $p_l[n]$ can be calculated for the corresponding iteration step n. The capacity c_l of a link l is assumed to be proportionally partitioned among the competing aggregates $a \in A_l$.

$$p_l[n] = \max\left\{1 - \frac{c_l}{\lambda_l[n]}, \quad 0\right\}. \tag{2}$$

The arrival rate $\lambda_l^a[n]$ of aggregate a and the loss probability at link l in iteration step n determine the arrival rate $\lambda_{next(a,l)}^a[n+1]$ of a at its next link at the next iteration step $n + 1$.

$$\lambda_{next(a,l)}^a[n+1] = \lambda_l^a[n] \cdot \big(1 - p_l[n]\big). \tag{3}$$

The throughput of aggregate a is given by

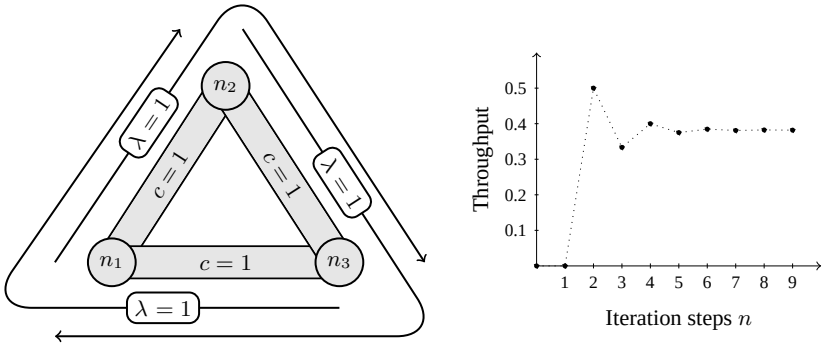
$$\theta^a[n] = \lambda_{sink(a)}^a[n]. \tag{4}$$

3.3 Calculation of Stationary Throughput for Static Traffic Demands

We now use the equations from the previous section to calculate the stationary throughput rates in settings with static traffic demands. By first calculating the throughput rates of a small example network analytically, we show that this can be a non-trivial task and propose an algorithm based upon the iterative equations as a simple alternative.

Figure 1a shows a triangle network with all link capacities set to $c = 1$ and three overlapping aggregates with static, non-adaptive sending rates $\lambda = 1$, as well. Intuition might lead to the belief that each aggregate should achieve a throughput of $\theta = 0.5$, as this corresponds to the most reasonable bandwidth distribution with respect to both fairness and throughput. However, the analytical calculation of the throughput yields significantly different results. Due to the symmetrical nature of the example, we assume that the loss probability p is the same on all links. The throughput, which equals the arrival rate at the second link of each aggregate, can then be expressed by $\theta = \lambda \cdot (1 - p)^2$. The arrival rate of each aggregate at its respective first link equals $\lambda \cdot (1 - p)$. As each link is crossed by two aggregates, i.e. one at its initial hop and the other at its second hop, the following equation must be solved: $c = \lambda(1 - p) + \lambda(1 - p)^2$. With $\lambda = 1$ and $c = 1$, and presuming that the loss probability is greater than 0, we can calculate the throughput resulting in $\theta \approx 0.38$.

While in this simple example we are able to calculate the throughput quite easily, by solving just one quadratic equation, we would have to solve much more complex



a. Triangle network **b.** Throughput evolution

Fig. 1: Throughput calculation for an triangle network.

sets of nonlinear equations in the case of larger networks. When instead applying the proposed iterative equations from the last section to the example network, we observe in Figure 1b that the stationary throughput of all aggregates converges at the same point as the analytical solution, i.e. at the value of about 0.38, after only a few iterations of the algorithm.

The exact solution for throughput rates in arbitrary networks is achieved after a number of iterations which corresponds to the hop count of the longest possible path composable of overlapping aggregates in the network. As in this example we can construct a path of infinite length due to circularly overlapping aggregates, an infinite number of iterations is required to obtain an exact solution. Therefore, in order to achieve reasonably precise results, we iteratively calculate the throughput of all aggregates until the differences between two consecutive arrival rates of all aggregates at each link drop below a predefined small value of ε. A formal description is provided in Algorithm 1 which may serve as an efficient and conceptually simple alternative to analytical throughput calculation.

4 Capturing Network Dynamics

In this section, we extend the flow-level simulation to dynamic scenarios, in which link and queue delay is modeled and the sending rates of aggregates can change in each iteration step independently of the link delay.

4.1 Definitions

In addition to the definitions from Section 3 we list the notations needed for the calculation of delay and queue size.

$d_l \cdot \Delta$	Link propagation delay of link l, $d_l \in \mathbb{N}$.
$q_l[n]$	Queue length at link l, initialized with $q_l[0] = 0$.
q_l^{max}	Maximum queue length at link l.
$\delta_l^a[n]$	Present delay of the data of aggregate a arriving at link l, initialized

Algorithm 1 Throughput calculation for static traffic demands.

$n \Leftarrow 0$
initialize $\lambda_l^a[n]$ with 0 $\quad \forall a \in A, \forall l \in L^a \setminus \{first_a\}$
repeat
  $stop \Leftarrow$ **true**
  for all $l \in L$ **do**
    calculate $\lambda_l[n]$ according to Equation (1)
    calculate $p_l[n]$ according to Equation (2)
    for all $a \in A_l$ **do**
      if $sink(a) \neq l$ **then**
        calculate $\lambda^a_{next(a,l)}[n+1]$ according to Equation (3)
        if $\left|\lambda^a_{next(a,l)}[n+1] - \lambda^a_{next(a,l)}[n]\right| > \varepsilon$ **then**
          $stop \Leftarrow$ **false**
        end if
      end if
    end for
  end for
  $n \Leftarrow n+1$
until $stop = true$
for all $a \in A_l$ **do**
  $\theta^a \Leftarrow \lambda^a_{sink(a)}[n]$
end for

with $\delta^a_{first(a)}[0] = 0$.

$\delta^a[n]$ End-to-end delay of the data of aggregate a arriving at its destination in iteration interval n.

4.2 Modeling Link Propagation Delay

The propagation delay $(d_l \cdot \Delta)$ is expressed by an integer multiple d_l of the iteration interval Δ. Due to this delay, the arrival rates $\lambda_l^a[n]$ at link l at iteration step n are propagated to their respective next link only at iteration step $n + d_l$:

$$\lambda^a_{next(a,l)}[n + d_l] = \lambda_l^a[n] \cdot \big(1 - p_l[n]\big). \tag{5}$$

In concrete implementations of our algorithm, the formulation of this equation implies that for each aggregate a, at every link l, a number of d_l values must be stored for future arrival at the respective next link. This can be viewed as dividing each link in d_l slots, and then shifting the data amount $(\lambda_l^a[n] \cdot \Delta)$ one slot further on its route in each iteration, as illustrated in Figure 2. This method can be implemented quite efficiently by using an array of fixed size d_l for each aggregate a at each link $l \in L^a$ and using (n modulo d_l) as index operator, as e.g. done in [18].

Each iteration maps the arrival rates $\lambda_l^a[n]$ for each link l and aggregate a to arrival rates $\lambda^a_{next(a,l)}[n + d_l]$ for the successor link $next(a, l)$ of the respective aggregate in the future. Thus, the traffic faces a delay of d_l iteration intervals until then. We take advantage of this fact for tracking the end-to-end delay, by using the following data structure. In addition to each $\lambda_l^a[n]$ we define a $\delta_l^a[n]$ and add the following operation to each iteration step to capture the delay up to this point.

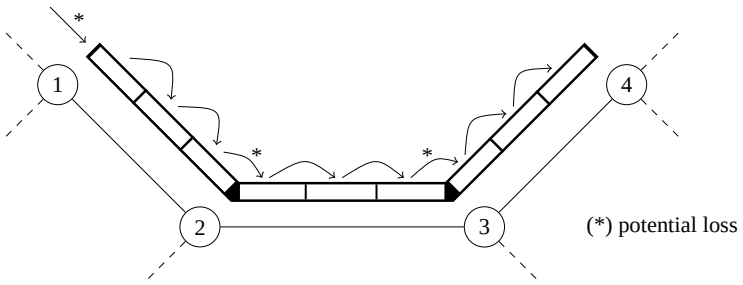


Fig. 2: Network with 4 nodes and a traffic aggregate that crosses 3 links, each with a link delay of $d_l = 3$.

$$\delta^a_{next(a,l)}[n + d_l] = \delta^a_l[n] + d_l. \tag{6}$$

The end-to-end delay is then given by

$$\delta^a[n] = \delta^a_{sink(a)}[n]. \tag{7}$$

For now, this does not yield insightful results because all end-to-end delays are constant. But we will use this data structure as a basis for keeping track of queueing delay in the next section.

4.3 Modeling Queuing Delay

So far, we have considered networks without queues, meaning that traffic exceeding the link bandwidth, i.e. $\lambda_l[n] > c_l$, is dropped. When using queues, the excess traffic that fits in the queue is buffered and only the carryover is dropped.

During one iteration interval, $(\lambda_l \cdot \Delta)$ new data arrives while $(c_l \cdot \Delta)$ data can be transmitted by link l. As the queue size can be at most q_l^{max}, the queue size for the next iteration step $(n + 1)$ can be calculated by:

$$q_l\,[n+1] = \min\left\{q_l^{max}, \quad \max\left\{q_l[n] - c_l \cdot \Delta + \lambda_l\,[n] \cdot \Delta, \quad 0\right\}\right\}. \tag{8}$$

At most $(q_l^{max} + c_l \cdot \Delta - q_l[n])$ can be buffered in the queue, while the exceeding traffic is dropped. Therefore, we can calculate the loss probability by:

$$p_l[n] = \max\left\{1 - \frac{q_l^{max} + c_l \cdot \Delta - q_l[n]}{\lambda_l[n] \cdot \Delta}, \quad 0\right\}. \tag{9}$$

The flow-level simulation works so far by calculating the arrival rate at a link l based on the current arrival rate at its predecessor for d_l iteration steps in the future. While the link delay d_l is constant, a queue introduces additional delay which can vary in each iteration step. We therefore substitute Equation (5) with the following equations.

Within iteration step n, $(\lambda_l[n] \cdot \Delta)$ data arrives which can be sent at the earliest in $i = \left\lfloor \frac{q_l[n]}{c_l \cdot \Delta} \right\rfloor$ iteration steps. Thus, i is the minimum queuing delay for the traffic of all

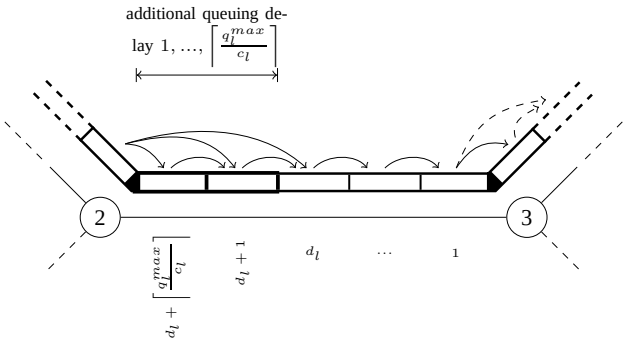


Fig. 3: Array data structure for traffic aggregates with queueing delays.

aggregates arrived during interval n. The maximum queueing delay for the arriving data corresponds to $k = \left\lceil \frac{q_l[n+1]}{c_l \cdot \Delta} \right\rceil$. We now calculate how the arriving data is distributed between the delays from i to k.

In order to compute the amount of data which can be dequeued in i iteration steps, we need to consider that it is possible that there is already data stored for this iteration step, which does not use the full link capacity. Therefore, we calculate the remaining capacity for this iteration step by $((i+1) \cdot c_l \cdot \Delta - q_l[n])$. This free capacity is shared proportionally by all aggregates competing for the link. As a consequence, the amount of data that will arrive at interval $(n + d_l + i)$ at the next link, $\lambda^a_{next(a,l)}[n + d_l + i] \cdot \Delta$, is increased. We flag $\lambda^a_{next(a,l)}[n + d_l + i]$ before the increase with a '$-$', and after the increase with a '$+$'.

$$\lambda^a_{next(a,l)}[n + d_l + i]^+ = \lambda^a_{next(a,l)}[n + d_l + i]^- + \min \left\{ \frac{(i+1) \cdot c_l \cdot \Delta - q_l[n]}{\Delta} \cdot \frac{\lambda^a_l[n]}{\lambda_l[n]}, \quad \lambda^a_l[n] \cdot (1 - p_l[n]) \right\}. \tag{10}$$

The arrival rates $\lambda^a_{next(a,l)}[n + d_l + j]$ with queueing delay j, $i < j < k$, make proportional use of the full link capacity c_l.

$$\lambda^a_{next(a,l)}[n + d_l + j] = c_l \cdot \frac{\lambda^a_l[n]}{\lambda_l[n]}, \qquad \forall j,\ i < j < k. \tag{11}$$

If $k > i$, the arrival rate $\lambda^a_{next(a,l)}[n + d_l + k]$ with queueing delay k corresponds to the proportional fraction of the latest buffered data in the queue $(q_l[n+1] - (k-1) \cdot c_l \cdot \Delta)$.

$$\lambda^a_{next(a,l)}[n + d_l + k] = \frac{q_l[n+1] - (k-1) \cdot c_l \cdot \Delta}{\Delta} \cdot \frac{\lambda^a_l[n]}{\lambda_l[n]}. \tag{12}$$

We observe that in the case of queues we calculate the arrival rates for several future iteration steps at once if $k > i$, which is illustrated in Figure 3. The data structure for

$\lambda_l^a[n]$ in a concrete implementation can essentially stay as in Equation (5) with the exception that the size of the respective modulo array must now be $d_l + \left\lceil \frac{q_l^{max}}{c_l} \right\rceil$.

To keep track of the end-to-end delay of aggregates we use the same data structure as in Equation (6) but add the additional queuing delay. For j with $i < j \leq k$ we can simply write

$$\delta^a_{next(a,l)}[n + d_l + j] = \delta^a_l[n] + d_l + j, \qquad \forall j,\ i < j \leq k. \tag{13}$$

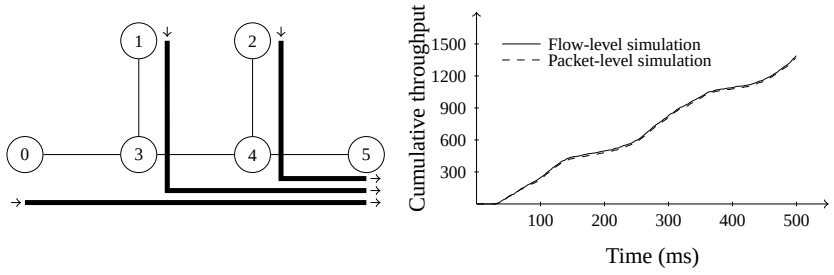
However, for the minimum delay i, it is possible that there is already data stored for arrival at interval $(n + d_l + i)$. Therefore, we have to calculate the weighted average of the delay of the data rate that was stored earlier ($\lambda^a_{next(a,l)}[n + d_l + i]^-$) and the delay of the data arrived in the last interval ($\lambda^a_{next(a,l)}[n + d_l + i]^+ - \lambda^a_{next(a,l)}[n + d_l + i]^-$). Again, we flag $\delta^a_{next(a,l)}$ before modifying with a '$-$' and after modifying with a '$+$'.

$$\begin{aligned} \delta^a_{next(a,l)}[n + d_l + i]^+ = \delta^a_{next(a,l)}[n + d_l + i]^- \cdot \left(\frac{\lambda^a_{next(a,l)}[n + d_l + i]^-}{\lambda^a_{next(a,l)}[n + d_l + i]^+} \right) \\ +\ (\delta^a_l[n] + d_l + i) \cdot \left(1 - \frac{\lambda^a_{next(a,l)}[n + d_l + i]^-}{\lambda^a_{next(a,l)}[n + d_l + i]^+} \right). \end{aligned} \tag{14}$$

5 Analysis of Simulation Accuracy

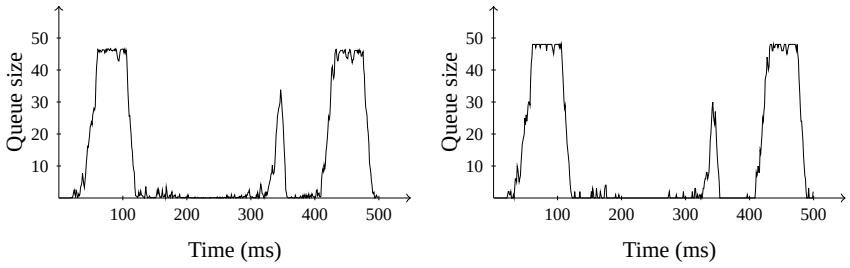
In this section we demonstrate the application of the proposed flow-level simulation model using a small example network in order to provide the reader with an insight about the character of results which may be expected.

The network in Figure 4a consists of six nodes and three traffic aggregates. The traffic is *Poisson*, meaning that the packet interarrival times are exponentially distributed. In addition, in order to generate a dynamic network-wide traffic matrix, the mean rate of each traffic source varies in an oscillatory manner with different frequencies.



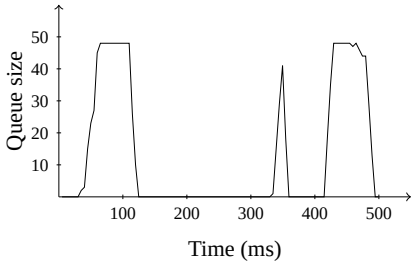
a. Example network. **b.** Cumulative throughput.

Fig. 4: Simple example network with cumulative throughput for the aggregate from 0 to 5.

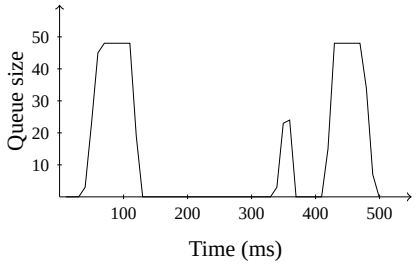


a. Packet simulation.

b. Flow-level simulation with Δ of 1 ms.



c. Flow-level simulation with Δ of 5 ms.



d. Flow-level simulation with Δ of 10 ms.

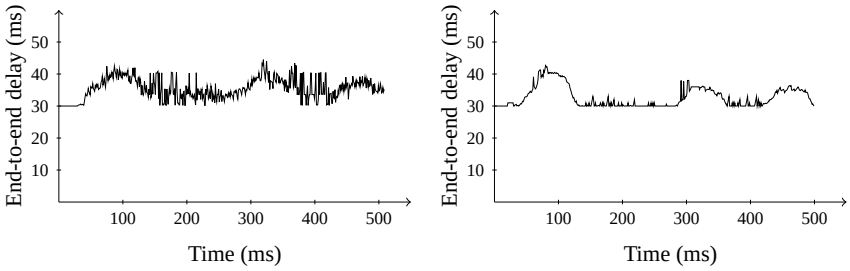
Fig. 5: Queue sizes for packet-level simulation and flow-level simulation with different iteration intervals.

The presented simulation setting results in temporary overload at both the link from 3 to 4 and from 4 to 5, allowing us to observe the throughput of aggregates which experience queueing and loss at more than one link. We performed the packet-level simulations in the *ns-2* network simulator [19] and the the flow-level simulations in our own simulator.

We concentrate on the aggregate from 0 to 5. In Figure 4b the cumulative throughput (the total amount of data arrived until iteration interval n) is shown for this aggregate for both packet-level simulation and flow-level simulation (with Δ of 1 ms), demonstrating that the flow-level simulation results closely resemble those from packet-level simulation.

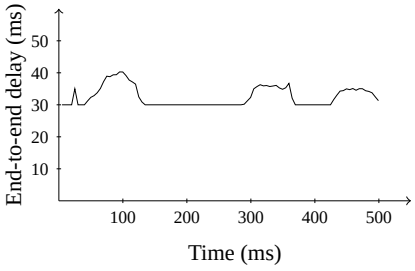
Next we demonstrate the effects of varying the iteration interval Δ in flow-level simulation. In Figure 5 the size of the queue at the link from node 4 to 5 is shown, using the same traffic scenario as before. For comparison, in Figure 5a the queue size over time is shown for packet-level simulation. For flow-level simulation with $\Delta = 1$ ms the queue size over time is very similar (Figure 5b). Increasing Δ to 5 ms (Figure 5c) and 10 ms (Figure 5d), we observe that the short-time variations disappear, but the basic pattern persists. The computational complexity of the flow-level simulation is inversely proprotional to the iteration interval, meaning that simulation with a Δ of 10 ms is 10 times as fast as simulation with a Δ of 1 ms.

In Figure 6 the total (i.e. end-to-end) packet delay is shown for the aggregate from 0 to 5 for packet simulation and flow-level simulation. In packet-level simulation, the

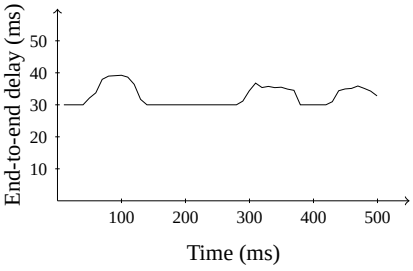


a. End-to-end delay in packet simulation.

b. End-to-end delay in flow-level simulation with Δ of 1 ms.



c. End-to-end delay in flow-level simulation with Δ of 5 ms.



d. End-to-end delay in flow-level simulation with Δ of 10 ms.

Fig. 6: End-to-end delay statistics for the aggregate from 0 to 5.

end-to-end delay was determined by calculating the average delay for all packets that arrived at their destination within an interval of 1 ms. In flow-level simulation we apply Equations (13) and (14). We observe that the calculated end-to-end delays for packet and flow-level simulation are quite similar when using a Δ of 1 ms, with the exception that the short-time variations are higher in packet-level simulation (Figures 6a–6b). In Figures 6c–6d we notice that these variations disappear completely, but the simulation still captures the basic end-to-end delay characteristics quite well.

6 Conclusion

In this paper we have provided a comprehensive formulation of discrete-time flow-level simulation at different levels of detail. We have first introduced a basic algorithm capable of calculating the loss probabilities and throughput rates of end-to-end traffic aggregates in a setting with static traffic demands, and have demonstrated this algorithm to be a conceptually very simple and efficient alternative to complex analytical throughput calculation.

We have then extended the algorithm to handle time-dependent behavior of networks by introducing link and queueing delay and a technique for measurement of end-to-end delay. We have compared results from flow-level simulation examples to results from packet-level simulation, and have shown that the they are very similar, especially

if a sufficiently small iteration interval Δ is chosen for flow-level simulation. A smaller interval leads to more accurate results while a larger interval speeds up the simulation. However, we have demonstrated that even for larger intervals the basic network behavior is captured. Future work may include the analysis of realistic ISP networks, both in terms of network size and traffic patterns observed in today's Internet.

Acknowledgements

This work has been performed partially in the framework of the Austrian Kplus Competence Center Program.

References

1. A. M. LAW AND W. D. KELTON, *Simulation Modeling and Analysis*, McGraw-Hill, 3^{rd} ed., 2000.
2. D. MITRA, D. ANICK, AND M. M. SONDHI, *Stochastic Theory of a Data Handling System with Multiple Sources*, Bell Systems Technical Journal, 61 (1982), pp. 1871–1894.
3. D. MITRA, *Stochastic Theory of a Fluid Model of Producers and Consumers Coupled by a Buffer*, Advances in Applied Probability, 20 (1988), pp. 646–676.
4. R. C. F. TUCKER, *Accurate Method for Analysis of a Packet-Speech Multiplexer with Limited Delay*, in IEEE Transactions on Communications, vol. 36, 1988, pp. 479–483.
5. J. M. PITTS, *Cell-Rate Modelling for Accelerated Simulation of ATM at the Burst Level*, in Communications, IEE Proceedings, 1995.
6. G. KESIDIS, A. SINGH, D. CHEUNG, AND W. KWOK, *Feasibility of Fluid Event-Driven Simulation for ATM Networks*, in IEEE Globecom, London, UK, Nov. 1996.
7. M. BAHR AND S. BUTENWEG, *On Rate-Based Simulation of Communication Networks*, in Design, Analysis, and Simulation of Distributed Systems (DASD), Apr. 2003.
8. B. LIU, D. R. FIGUEIREDO, Y. GUO, J. F. KUROSE, AND D. F. TOWSLEY, *A Study of Networks Simulation Efficiency: Fluid Simulation vs. Packet-level Simulation*, in IEEE Infocom, 2001, pp. 1244–1253.
9. B. LIU, Y. GUO, J. KUROSE, D. TOWSLEY, AND W. GONG, *Fluid Simulation of Large Scale Networks: Issues and Tradeoffs*, Tech. Rep. UM-CS-1999-038, 1999.
10. A. YAN AND W.-B. GONG, *Time-Driven Fluid Simulation for High-Speed Networks*, IEEE Transactions on Information Theory, 45 (1999), pp. 1588–1599.
11. Y. LIU, F. L. PRESTI, V. MISRA, D. TOWSLEY, AND Y. GU, *Fluid Models and Solutions for Large-Scale IP Networks*, in ACM SIGMETRICS, 2003, pp. 91–101.
12. I. GOJMERAC, T. ZIEGLER, F. RICCIATO, AND P. REICHL, *Adaptive Multipath Routing for Dynamic Traffic Engineering*, in Proc. IEEE Globecom 2003, San Francisco, USA, 2003, pp. 3058–3062.
13. I. GOJMERAC, T. ZIEGLER, AND P. REICHL, *Adaptive Multipath Routing Based on Local Distribution of Link Load Information*, in International Workshop on Quality of future Internet Services (QofIS), 2003, pp. 122–131.
14. I. GOJMERAC, L. JANSEN, T. ZIEGLER, AND P. REICHL, *Feasibility Aspects of AMP Performance Evaluation in a Fluid Simulation Environment*, in MMBnet Workshop, Hamburg, Germany, 2005.
15. I. GOJMERAC, L. JANSEN, T. ZIEGLER, AND P. REICHL, *A Simulation Study of Microscopic AMP Behavior*, in Polish-German Teletraffic Symposium (PGTS), 2006.
16. V. MISRA, W.-B. GONG, AND D. F. TOWSLEY, *Fluid-Based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED*, in ACM SIGCOMM, 2000, pp. 151–160.
17. M. A. MARSAN, M. GARETTO, P. GIACCONE, E. LEONARDI, E. SCHIATTARELLA, AND A. TARELLO, *Using Partial Differential Equations to Model TCP Mice and Elephants in Large IP Networks*, in IEEE Infocom, 2004.
18. FLUID FLOW MODEL IN NS, *Source code available at ftp://gaia.cs.umass.edu/pub/ffm_in_ns.tar.gz*, 2003.
19. THE NETWORK SIMULATOR (VERSION 2), *Source code and documentation available at http://www.isi.edu/nsnam/ns/*, 1995.