Simulating Stochastic Processes with OMNeT++

Jan Kriege, Peter Buchholz
Department of Computer Science
TU Dortmund
{jan.kriege,peter.buchholz}@tu-dortmund.de

ABSTRACT

Modeling of correlated traffic processes is a challenge in simulation modeling but is necessary to obtain realistic simulation models of many computer or communication networks. We present a new simple module denoted as `ArrivalProcess` to be included as a traffic source in OMNeT++ simulation models. The new traffic source supports different stochastic process models which are defined using a XML format. The process descriptions can be automatically generated from trace data using the software tool ProFiDo which is also briefly presented. By means of examples it is shown that the use of stochastic processes rather than simple distributions results in much more accurate results for systems with correlated arrival streams.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Stochastic processes

General Terms

Performance

Keywords

Markovian Arrival Processes, ARMA Processes, ARTA Processes, Input Modeling

1. INTRODUCTION

The accurate modeling of the traffic is essential to obtain sufficiently accurate models of computer or communication networks. It is known for some time that traffic processes in modern networks include complex dependencies such that the simple modeling with Poisson processes or even with some more complex interarrival time distribution is not sufficient [21, 27]. Usually, arrival streams in networks have a high variability and interarrival times are positively correlated. The neglection of this correlation often results in a dramatic underestimation of resource requirements.

OMNeT++ 2011, March 21, Barcelona, Spain

DOI 10.4108/icst.simutools.2011.245512

Although the necessity to model traffic sources adequately is accepted in principle, it is not really supported in simulation. In the standard simulation literature [19] the fitting of distributions is introduced in breadth whereas the modeling of correlated arrival streams using stochastic processes is only very briefly touched. The same situation can be observed in simulation software. Standard tools to model input streams from trace data like Expert Fit [20] or the Arena Input Analyzer [17] support the modeling of distributions without taking correlations into account. Even more, simulation tools support the specification of a large number of different distributions but models for correlated traffic usually have to be hand-coded.

OMNeT++ [13] as open source simulation framework is nowadays widely used for the simulation of communication protocols and networks. It contains support for modeling complex networks by providing models for a large number of protocols. However, for traffic generation, OMNeT++, like other simulation tools, only supports the usage of distributions that have to be fitted using some external software tool like ExpertFit [20]. We are not aware of any support to model correlated input streams. Nevertheless, there are a few papers which consider the modeling of more complex traffic streams according to specific applications. In [15] HTTP traffic generation according to general distributions with some additional parameters to define activity periods during a day is described. VoIP traffic generation according to real sound files is considered in [6]. Both approaches do not provide a general approach for modeling correlated input traffic.

The general use of correlated arrival streams is prohibited by missing tool support to generate arrival process specifications from measured data and by the missing support to represent arrival processes in simulation tools. In this paper we describe a step towards an integration of stochastic processes as traffic sources for correlated input streams in OMNeT++. We present a tool to fit the parameters of different types of stochastic processes according to the values in a trace and we show how the resulting process models can be integrated in OMNeT++ by defining an Arrival Process Module. By means of two examples it is shown that the use of stochastic processes rather than distributions results in a much better representation of the real behavior of a network.

This paper is structured as follows. In Sect. 2 several stochastic process types currently supported by our Arrival Process Module for OMNeT++ are briefly introduced. The ProFiDo framework which is used to generate the OMNeT++ Module from measured traces is outlined in Sect. 3. The

OMNeT++ Arrival Process Module is presented in detail in Sect. 4 and in Sect. 5 we give some application examples for the module. The paper ends with the conclusions in Sect. 6.

2. THEORETICAL BACKGROUND

This section presents different types of stochastic processes that have been proposed in the past for modeling correlated input streams, it summarizes fitting approaches for these processes and it describes the requirements that are necessary to use the processes in simulation.

2.1 Phase-Type Distributions and Markovian Arrival Processes

The first class of processes we consider generate arrivals by the transitions of a finite Markov chain. We begin with the introduction of the basic class of distributions based on this model, namely Phase-type distributions. A Phase-type (PH) Distribution [25] of order n can be described by an absorbing Markov Chain with n transient and one absorbing state. It is defined by a non-singular $n \times n$ matrix $\boldsymbol{D}_0$ with $\boldsymbol{D}_0(i,j) \geq 0$ for $i \neq j$, $\boldsymbol{D}_0(i,i) \leq -\sum_{j=1,j\neq i}^{n} \boldsymbol{D}_0(i,j)$ and a row vector $\boldsymbol{\pi}$ with $\boldsymbol{\pi}(i) \geq 0$ and $\boldsymbol{\pi}\mathbf{1} = 1$ where $\mathbf{1}$ is the unit column vector of length n. Markovian Arrival Processes (MAPs) [24] are a generalization of PH distributions that can model autocorrelations. A MAP of order n is defined by two $n \times n$ matrices $(\boldsymbol{D}_0, \boldsymbol{D}_1)$ where $\boldsymbol{D}_0$ is as defined for a PH distribution and $\boldsymbol{D}_1 \geq 0$. Then Matrix $\boldsymbol{Q} = \boldsymbol{D}_0 + \boldsymbol{D}_1$ with $\boldsymbol{Q}\mathbf{1} = \mathbf{0}$ is the generator matrix of an irreducible Markov process. Matrix $\boldsymbol{D}_0$ contains the rates of internal transitions without an arrival and matrix $\boldsymbol{D}_1$ contains the rates of transitions generating an arrival. Every PH distribution $(\boldsymbol{D}_0, \boldsymbol{\pi})$ can be interpreted as a MAP by setting $\boldsymbol{D}_1 = -\boldsymbol{D}_0\mathbf{1}\boldsymbol{\pi}$.

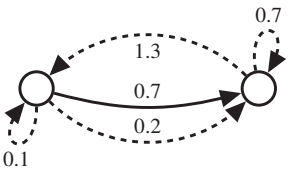

Figure 1: Example for a MAP of order 2

A simple example for a MAP of order 2 is shown in Fig. 1. Solid edges denote transition from $\boldsymbol{D}_0$, dashed edges transitions from $\boldsymbol{D}_1$. Whenever the process performs a dashed transition, an arrival is generated. It has been shown [14] that MAPs are capable of modeling a large number of different behaviors and that they allow an accurate modeling of most processes appearing in real networks. However, the finding of the right parameters, which is denoted as parameter fitting, is a nonlinear optimization problem which requires some effort. Nevertheless, much progress has been made in finding methods for parameter fitting of PH distributions and MAPs such that nowadays a large number of approaches are available in the literature. In general one can distinguish Expectation Maximization methods, that try to maximize the likelihood, and approaches that fit a PH distribution or a MAP according to different quantities like empirical moments, joint moments or lag-k coefficients of autocorrelation that have been estimated from a trace. An overview of different methods can be found in [14]. Some more recent approaches are summarized in [18]. However, to use MAPs in practice, software tools have to be available that incorporate different fitting methods and allow an easy and efficient generation of a MAP from measured data.

PH Distributions and MAPs have originally been developed as input processes in queuing systems that are solved analytically [22]. However, they may as well be used in simulation models. Sampling from MAPs can be done by simulating the underlying Markov chain and by generating an arrival whenever a transition from $\boldsymbol{D}_1$ occurs. This requires storing the two matrices $(\boldsymbol{D}_0, \boldsymbol{D}_1)$ and the current state and drawing random numbers from a uniform distribution to determine the next state and from an exponential distribution to determine the transition times.

2.2 Autoregressive Moving Average Processes

Autoregressive Moving Average (ARMA) Processes [7] are used for a long time in different areas of statistics but have been rarely used as input processes in simulation models. They describe a time series with dependent successive values Z_t. A simple subclass is the Autoregressive process of order p, denoted as $AR(p)$, which is defined as

$$Z_t = \alpha_1 Z_{t-1} + \alpha_2 Z_{t-2} + ... + \alpha_p Z_{t-p} + \epsilon_t \qquad (1)$$

where the innovation ϵ_t has a Normal distribution with zero mean and variance σ_ϵ^2 and the coefficients α_i are used to weight the previous observations Z_{t-i}. If one weights previous innovations instead of previous observations this results in a time series

$$Z_t = \epsilon_t + \beta_1 \epsilon_{t-1} + \beta_2 \epsilon_{t-2} + ... + \beta_q \epsilon_{t-q}. \qquad (2)$$

which is called a Moving Average process of order q and denoted $MA(q)$. A combination of the two previous description finally yields an $ARMA(p,q)$ process, which is defined as

$$Z_t = \alpha_1 Z_{t-1} + ... + \alpha_p Z_{t-p} + \epsilon_t + \beta_1 \epsilon_{t-1} + ... + \beta_q \epsilon_{t-q}. \qquad (3)$$

For estimating the parameters of AR, MA and ARMA processes methods like maximum likelihood estimation [16] or least squares regression approaches [23] are known for some time and have been implemented in statistical software packages.

For sampling from ARMA processes the vectors with the AR and MA coefficients α_i and β_j have to be stored. Additionally, the previous p elements of the time series and the previous q innovations have to be saved. Moreover, ARMA processes require the possibility to draw random numbers from a Normal distribution to determine the next innovation. Since ARMA processes always assume a Normal distribution for the generated observations Z_t a sufficiently accurate modeling of observed traffic process with a clearly non normal marginal distribution is problematic. Furthermore an ARMA process may generate negative values which cannot be used in simulation as e.g. interarrival times where only positive values are allowed [1]. This issue can be addressed by transforming the Z_t, e.g. [31] discusses several linear and non-linear transformations and its properties.

2.3 Autoregressive To Anything Processes

The major shortcoming of ARMA models is their dependence on the Normal distribution. Since innovations are

drawn from a Normal distribution only marginal distributions that are simple transformations of the Normal distribution can be modeled. However, it is known that real traffic in networks often cannot be adequately described by Normal distributions. To overcome this shortcoming Autoregressive To Anything (ARTA) Processes have been introduced in [5, 9] for simulation input modeling. ARTA processes combine an $AR(p)$ base process with an arbitrary marginal distribution F_Y and thus can model correlated input processes with a wide variety of shapes for the distribution. They are defined as a sequence

$$Y_t = F_Y^{-1}[\Phi(Z_t)], t = 1, 2, ... \quad (4)$$

where F_Y is the marginal distribution, Φ is the standard Normal cumulative distribution function and $\{Z_t; t = 1, 2, ...\}$ is a stationary Gaussian $AR(p)$ process as given in Eq. 1. The $AR(p)$ base process can be constructed in a way such that the $\{Z_t\}$ have a standard Normal distribution and then the probability-integral transformation $U_t = \Phi(Z_t)$ ensures that U_t is uniformly distributed on $(0, 1)$. Application of $Y_t = F_Y^{-1}[U_t]$ yields a time series $\{Y_t, t = 1, 2, ...\}$ with the desired marginal distribution F_Y. This approach works for any distribution F_Y for which F_Y^{-1} can be computed, either by a closed-form expression or by numerical methods. [9, 10] established a relation between the autocorrelation structures of the ARTA process and the $AR(p)$ base process and fitting methods for ARTA models are given in [5, 9].

The simulation of an ARTA process can be performed in two steps. First, the next random variate Z_t from the underlying base process is determined as described in Sect. 2.2 and in the second step the transformation from Eq. 4 is applied to compute Y_t. Hence, the simulation requires methods for simulating an $AR(p)$ process and for computing the inverse cumulative distribution function of all supported distributions F_Y.

3. PROFIDO

ProFiDo [2, 3] is a flexible Java-based toolkit for fitting and analyzing stochastic processes. It provides a graphical user interface for a consistent use of commandline-oriented tools that implement several of the fitting methods as briefly introduced in Sect. 2. These tools can be arranged into a workflow to realize different steps of data preprocessing, parameter fitting and analysis of stochastic processes. For ProFiDo we have developed an XML interchange format for the description of stochastic processes that ensures the interoperability of different fitting tools in a workflow. An example for the XML interchange format is shown in Listing 1 that contains the XML representation of the MAP from Fig. 1. The interested reader is referred to [2] for a general overview of the framework including the fitting tools currently supported by ProFiDo and a description of the XML interchange format.

Listing 1: MAP description in ProFiDo's XML format

```
<map>
  <states>2</states>
  <d0>
    -1.0   0.7
     0.0  -2.0
  </d0>
  <d1>
    0.1  0.2
    1.3  0.7
  </d1>
</map>
```

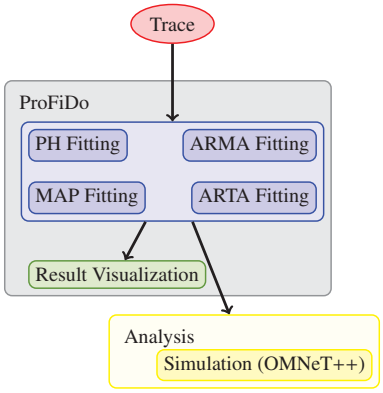

Figure 2: ProFiDo Framework

Fig. 2 outlines the components of the toolkit. ProFiDo takes a trace, that contains observations obtained from a real system, as input and can fit one or several different stochastic processes according to the specified workflow. Additionally, ProFiDo offers the possibility to visualize various properties like density and distribution function or autocorrelation coefficients of the fitted processes and the measured trace.

Once a stochastic process with the desired properties has been fitted, it can be easily integrated into (simulation) models. For this purpose, we have linked a module for OMNeT++ to the ProFiDo framework that loads stochastic processes from ProFiDo's XML interchange format and generates correlated events by simulating these processes. In the following sections we will introduce the architecture of this module and give some application examples.

4. ARRIVAL PROCESS MODULE

OMNeT++ models consist of modules that can be arranged in a hierarchical way and that can communicate via gates using message passing. One can distinguish between simple modules, that are the basic building blocks of a model, and compound modules. Simple modules are written in C++ accompanied by a NED-description that defines the interface of the module consisting of gates and parameters which pass configuration data to the simple module. Compound modules are a grouping of simple modules and other compound modules introducing a hierarchy.

The Arrival Process Module for OMNeT++ presented in this work is a simple module that can generate random numbers from the stochastic processes introduced in Sect. 2, i.e. MAPs, ARTA processes and ARMA processes. The model description is parsed from a file in the ProFiDo XML interchange format. In this way, the process description derived by ProFiDo with some fitting algorithm can be used in an OMNeT++ simulation model without any additional programming effort.

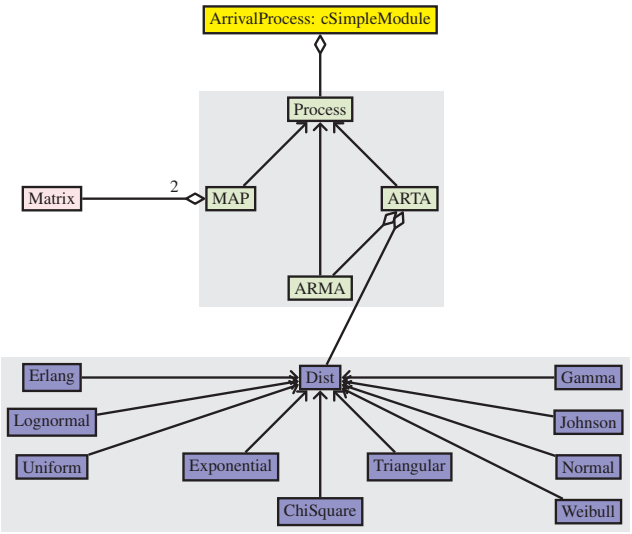

Figure 3: Class Hierarchy of the Arrival Process Module

Fig. 3 shows the class hierarchy of the module. The class `ArrivalProcess` is the main class of the module that contains its C++ implementation. It inherits from `cSimpleModule` as all simple modules in OMNeT++ do. The corresponding NED-description is shown in Listing 2.

Listing 2: NED description of the Arrival Process Module

```
simple ArrivalProcess
{
 parameters:
  xml model;
  string transform = default("");
  @display("i=block/source");
 gates:
  output out;
}
```

The module has one XML type parameter that is used to pass the model description and one string parameter that can be used to specify a function for a transformation of the time series as it will be explained later. The only gate of the module is an output gate. Whenever an arrival according to the stochastic process is generated, a message is passed through this gate to another module that is connected to this gate with one of its input gates and processes the incoming arrival. The implementation of the class `ArrivalProcess` is actually quite simple, since all the code necessary for simulating the processes is contained in the class `Process` and its subclasses shown in Fig. 3. `ArrivalProcess` provides methods for loading the process description from an XML file and initializes a `Process` according to the contents of the XML description. Additionally, it has to provide a method `handleMessage()` for dealing with message events. `Process` is an abstract class that all the classes for the actual stochastic processes like MAPs or ARTA processes inherit from. All inheriting classes have to implement a method called `getNextRandomVariate()` that generates the next random number from that process. Hence, `handleMessage()` from the class `ArrivalProcess` schedules the next arrival according to `getNextRandomVariate()`. When the arrival is due, a message is sent to the outgoing gate and the next arrival is scheduled according to the result of `getNextRandomVariate()` by initiating a self-message that arrives at the generation time of the next message.

In the following paragraphs we will briefly introduce the subclasses that inherit from the class `Process` and that implement the random number generation for the different process types.

4.1 Simulation of MAPs

The class `MAP` is used for simulating Markovian Arrival Processes. Of course, it can also be used for the simulation of PH distributions as well, if they are transformed into MAPs. The class `Matrix` is used as an utility class to store the two matrices $\boldsymbol{D}_0$ and $\boldsymbol{D}_1$. A MAP is initialized by drawing the initial state from the distribution defined by $\boldsymbol{\pi}$ which is given as the unique solution of $\boldsymbol{\pi}(-\boldsymbol{D}_0^{-1}\boldsymbol{D}_1) = \boldsymbol{\pi}$ normalized to 1 and contains the stationary distribution just after an arrival has occurred. The class simulates the underlying Markov Chain by drawing an exponentially distributed random number with rate $|\boldsymbol{D}_0(i,i)|$ corresponding to the current state i to determine the next transition time and an uniformly distributed random number to compute the next state according to the probability distribution defined by $\boldsymbol{D}_0(i,j)/|\boldsymbol{D}_0(i,i)|$ and $\boldsymbol{D}_1(i,j)/|\boldsymbol{D}_0(i,i)|$. This is repeated until a transition from $\boldsymbol{D}_1$ occurs. In this case the sum of the transition times is returned and the current state is stored as starting point for the generation of the next random sample. For drawing the required random numbers with exponential and uniform distribution the random number generators of OMNeT++ are used.

4.2 Simulation of ARMA Processes

The simulation of $ARMA(p,q)$ processes is realized by the class `ARMA`. The class can serve for generating arrival events directly or as a base process for an ARTA model. In the former case it is recommended to transform the generated observations (see below) to avoid the problems with negative times that cannot be processed in simulation models as mentioned in Sect. 2.2. The class manages four lists for the p AR coefficients, the q MA coefficients, the last p observations and the last q innovations. The first two lists are fixed after initialization while the last two have to be updated after each generation of an arrival event. For initialization the p previous observations and the q previous innovations have to be determined. For the innovations this is straightforward, because they are iid random numbers, and consequently the $\epsilon_{t-1}, \ldots, \epsilon_{t-q}$ can be drawn from a Normal distribution with zero mean and variance σ_ϵ^2. The p previous observations are correlated and have to be initialized by drawing from a multivariate Normal distribution, which is done using a procedure as described in [19]. Let

$$\boldsymbol{\Sigma} = \begin{bmatrix} r_0 & r_1 & \cdots & r_{p-1} \\ r_1 & r_0 & \cdots & r_{p-2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p-1} & r_{p-2} & \cdots & r_0 \end{bmatrix}$$

be the covariance matrix with the autocovariance values $(r_0, r_1, \cdots, r_{p-1})$ of the ARMA process and let $\boldsymbol{\mu} = (\mu, \cdots, \mu)$ be a vector that contains p times the mean value μ of the ARMA process. We draw p independent random numbers $\boldsymbol{X} = (X_0, X_1, \cdots, X_{p-1})^T$ from a standard Normal distribution and compute the lower triangular matrix $\boldsymbol{C}$ with $\boldsymbol{\Sigma} = \boldsymbol{C}\boldsymbol{C}^T$. Then, we can generate the multivariate Normal random numbers that are used to initialize the $Z_{t-i}, i = 1...p$ by setting $\boldsymbol{Z} = \boldsymbol{\mu} + \boldsymbol{C}\boldsymbol{X}$. See [8] for a method to compute the autocovariance values of an ARMA process and [11] for a simple method to perform the Cholesky decomposition to obtain matrix $\boldsymbol{C}$. For simulation the class draws a normally distributed random number as new innovation and computes the weighted sum according to Eq. 3. Finally, the oldest observation and the oldest innovation are replaced by the newly computed observation and the new innovation, respectively.

4.3 Simulation of ARTA Processes

The class `ARTA` provides methods for simulating ARTA models. The `ARTA` class has two objects, one of type `ARMA` and one of type `Dist` that are required to perform the simulation of the ARTA process. The `ARMA` class is used to simulate the base process and generates correlated random number with Normal distribution. This random number is then transformed twice; first into a uniformly distributed random number and then using the inverse cdf to yield the final ARTA random number (cf. Eq. 4). The class `Dist` is an abstract superclass with the virtual method `inverse_cdf (double x)` that all the distributions from Fig. 3 have to implement to compute the inverse cdf for the last transformation step. The implementation of the ARTA class requires several numerical methods for which fast approaches exist in the literature. For the transformation from normally to uniformly distributed random numbers the computation of the Normal cdf is necessary which is done by using the approach from [12]. All other required numerical methods are related to the computation of the inverse cumulative distribution function. The inverse cdf of the Normal distribution is computed using the algorithm from [30]. The same holds for the inverse cdf of the Lognormal and Johnson distribution, which are both derived from the Normal distribution. The algorithm from [4] is used to compute the inverse cdf of Gamma, Erlang and χ^2 distributions, that are all related. For the Uniform, Exponential, Weibull and Triangular distribution closed-form expressions exist for the inverse cdf [19].

4.4 Post-Processing of the Time Series

As already mentioned before, it might be desirable or even necessary to transform the generated interarrival times in a post-processing step. This is for example the case when the input process has been fitted according to a trace that uses a different time scale than the rest of the model. For these models one could adjust the complete process to the desired time scale or transform the generated values of the model in a post-processing step. Another reason for a transformation of the generated interarrival times are processes that might output invalid values. In Sect. 2.2 linear and non-linear transformations for $ARMA(p, q)$ have been mentioned for such cases. The OMNeT++ Arrival Process Module accounts for these requirements by providing the possibility to enter a post-processing function using OMNeT++'s NED language expressions. The function is passed as parameter `transform` (cf. Listing 2). NED language expressions have a C-like syntax and may make use of various mathematical functions. A list with possible functions can be found in [26]. For example non-linear transformations of the type

$$Y_t = c_1^{\frac{1-c_2}{c_2+c_3} Z_t}$$

for given $c_1 > 1$, $0 < c_2 < 1$ and some small constant c_3 can be realized by the function `pow(c_1, $ARRIVAL * (1-c_2)/(c_2+c_3))`. Note, that `$ARRIVAL` is a placeholder and the Arrival Process Module will replace every instance of `$ARRIVAL` in the expression with the current value obtained from simulating the stochastic process when the expression is evaluated. The transformation of the values Z_t from the ARMA process assures that the resulting values Y_t that are used in the simulation model are all non-negative. Observe that for the fitting of the ARMA process generating Z_t the original values τ_i in the trace have to be scaled to $\nu_i = (c_2 + c_3)/(1 - c_2) \cdot \log_{c_1} \tau_i$ such that the resulting values Y_t build a model for the original trace.

5. APPLICATION EXAMPLES

In the following we present two application examples to show how the `ArrivalProcess` module can be incorporated into OMNeT++ models. The results obtained from these simulation models support the observation that the negligence of autocorrelation may have serious impact on the simulation results.

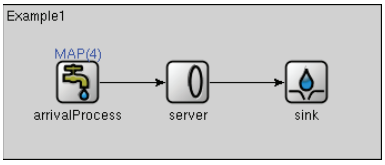

Figure 4: Simple OMNeT++ Model with ArrivalProcess module

Listing 3: Configurations for the Model from Fig. 4

```
1  [General]
2  network = Example1
3  **.server.serviceTime = exponential(0.5s)
4  **.server.buffer = 10
5
6  [Config MAP]
7  description = "Arrivals from MAP"
8  **.arrivalProcess.model = xmldoc("map.xml")
9
10 [Config ARTA]
11 description = "Arrivals from ARTA process"
12 **.arrivalProcess.model = xmldoc("arta.xml")
```

The first example model is a simple queueing model shown in Fig. 4. It consists of the `ArrivalProcess` module feeding a single server queue with a capacity of 10. We modeled different configurations of the model by generating interarrival times according to a MAP of order 4 and according to an ARTA model with exponential marginal distribution. Listing 3 shows the different configurations for the model.

Note, that the stochastic process used by the `ArrivalProcess` module can easily be exchanged by specifying another XML description using the parameter `**.arrivalProcess.model`.

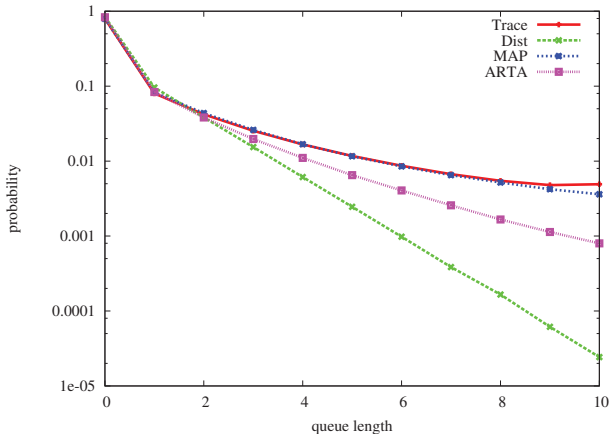

Figure 5: Queue Length Distribution for Utilization $\rho = 0.4$

The model is analyzed with several utilization levels for the server between $\rho = 0.4$ and $\rho = 0.8$ and the queue length distribution is taken as result measure. The two mentioned stochastic processes have been obtained by fitting MAPs and ARTA models to the trace *LBL-TCP-3* [27] from the Internet Traffic Archive [29]. For comparison we simulated the same setup with a slightly modified model that reads the interarrival times of the jobs directly from the trace to obtain reference values for the queue length according to a trace driven simulation. Finally, in another series of experiments we used an exponential distribution (i.e. the same distribution that the ARTA model uses as marginal distribution) fitted to the trace as a traffic generator with independent interarrival times resulting in a Poisson input process.

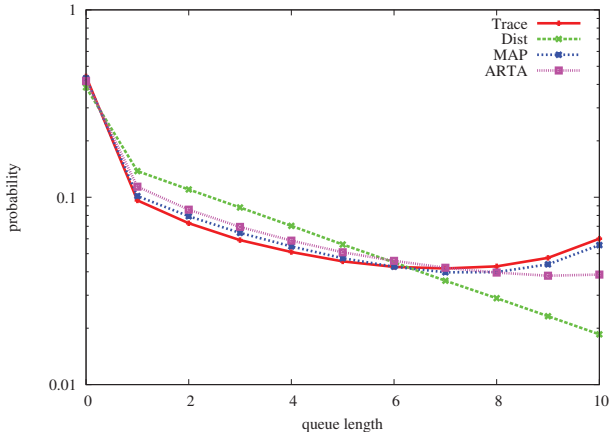

Figure 6: Queue Length Distribution for Utilization $\rho = 0.8$

Fig. 5 shows the queue length distribution for utilization $\rho = 0.4$. It is clearly visible that the probabilities of queue length up to 2 are similar for the trace, the two stochastic processes and the distribution. For larger values towards the tail of the queue length distribution the model with uncorrelated inputs significantly underestimates the probabilities. E.g., for queue length 10 the difference between the values of the trace driven simulation and the model with the Poisson input process is between 2 and 3 orders of magnitude. Among the two stochastic processes, the MAP provides much better results than the ARTA process.

For a utilization of $\rho = 0.8$ the stochastic processes perform better for almost all values of the queue length distribution as shown in Fig. 6. Again, the MAP gives better results than the ARTA process.

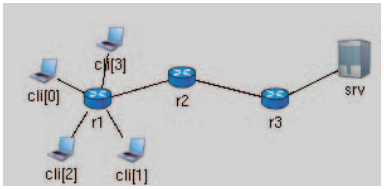

Figure 7: Example Model with simple Network

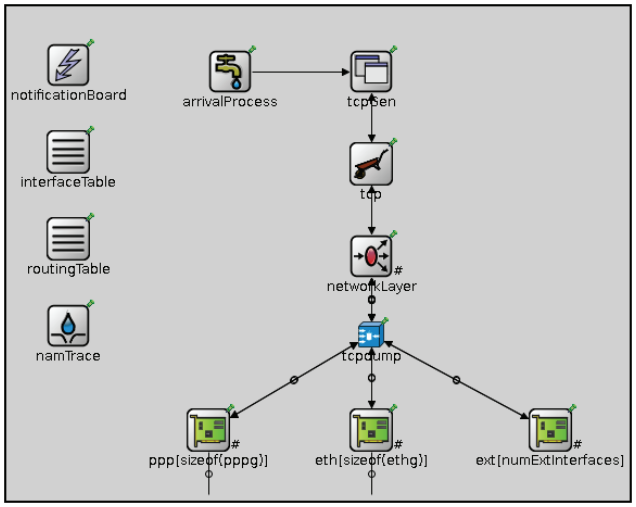

Figure 8: Host from Fig. 7

As a second example we modified the `NClients` model that is part of the INET Framework to use the `ArrivalProcess` module. Fig. 7 gives an overview of the network that consists of four client hosts connected to a server via different routers. Fig. 8 shows the inner view of one of those hosts.

The host consists of a module for the TCP protocol implementation, a module for the network layer and several interfaces, that are taken from the implementation of the `StandardHost` of the INET framework. TCP packets are generated according to the events created by the `ArrivalProcess` module. Since the `ArrivalProcess` module only generates correlated events, but is not tailored to a specific proto-

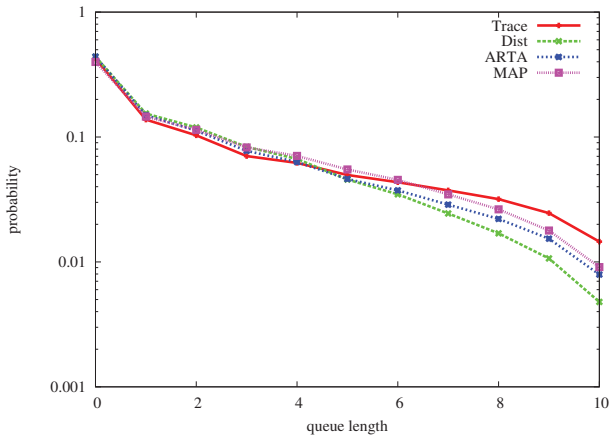

Figure 9: Queue Length Distribution for the Server from Fig. 7

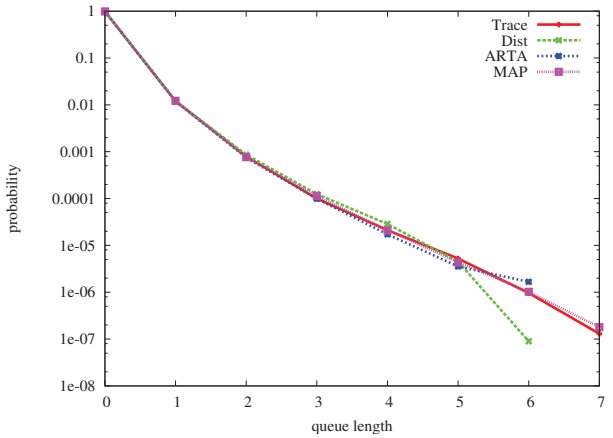

Figure 10: Queue Length Distribution for one Interface of a Router from Fig. 7

col implementation, this module is connected to the module `tcpGen` that generates a TCP connection and sends packets to the server whenever it is notified by the `ArrivalProcess` module. The server replies with a larger chunk of data to the request of `tcpGen`, i.e. the modules model typical web browsing behavior with a small request to the server and a larger reply by the server.

The four `ArrivalProcess` modules are initialized with ARTA models with different exponential marginal distributions and MAPs that have been fitted to different parts of the trace *BC-pAug89* [27]. For comparison we run the same setup with a slightly altered model where uncorrelated TCP packets are generated according to the exponential distributions that have also been used for the ARTA models. To obtain reference values a trace driven simulation with the original traces was run again.

Fig. 9 shows the queue length distribution of the server's network interface. Again, it is visible that the arrivals generated from stochastic process provide a better approximation of the tail of the queue length distribution than uncorrelated arrivals. For the first router the load is distributed between several different interfaces connected to the different clients and thus, the effect of the correlated packets is not as significant as for the server. Nevertheless, one can see from Fig. 10 that arrivals generated by MAPs result in a very close approximation of the queue length distribution, while independent and identically distributed arrivals underestimate the higher values of the distribution.

These results clearly demonstrate the importance of incorporating autocorrelation into input models. In particular, the results of the ARTA processes with exponential marginal distribution compared to the same independent exponential distribution show that adding a few lags of autocorrelation might help to improve the quality of simulation models significantly.

6. CONCLUSIONS

In this paper we present a new simple module, called `ArrivalProcess`, which can be used in OMNeT++ simulation models as a traffic source. In contrast to other available traffic sources, the new module supports the use of stochastic processes to generate correlated arrival streams with a wide variety of marginal distributions. In its current version the module supports the well known ARMA processes, as well as ARTA processes that have been developed more recently for the use in simulation models and it also supports Markovian Arrival Processes. For process description a simple XML interface has been defined. The arrival process module is accompanied by a software tool named ProFiDo which allows one to compute the parameters of arrival processes from trace data and generates automatically the XML description of the resulting stochastic process such that the whole approach is ready to be used for modeling real systems. Both, ProFiDo and the `ArrivalProcess` module for OMNeT++, are available on the ProFiDo homepage [28].

The approach can be easily extended by incorporating other process types in the module `ArrivalProcess` and additional fitting methods into ProFiDo. However, our current impression is that the major available techniques for modeling correlated arrival streams are adequately supported.

7. REFERENCES

[1] F. Bause, P. Buchholz, and J. Kriege. A Comparison of Markovian Arrival and ARMA/ARTA Processes for the Modeling of Correlated Input Processes. In M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, editors, *Proc. of the Winter Simulation Conference (WSC) 2009*, pages 634–645. Institute of Electrical and Electronics Engineers, Inc., 2009.

[2] F. Bause, P. Buchholz, and J. Kriege. ProFiDo - The Processes Fitting Toolkit Dortmund. In *Proc. of the 7th International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 87–96, 2010.

[3] F. Bause, P. Gerloff, and J. Kriege. ProFiDo - A Toolkit for Fitting Input Models. In B. Müller-Clostermann, K. Echtle, and E. P. Rathgeb, editors, *Proceedings of the 15th International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems and Dependability*

and Fault Tolerance (MMB & DFT 2010), volume 5987 of *LNCS*, pages 311–314. Springer, 2010.

[4] D. J. Best and D. E. Roberts. Algorithm AS 91: The Percentage Points of the χ^2 Distribution. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24(3):385–388, 1975.

[5] B. Biller and B. L. Nelson. Fitting time-series input processes for simulation. *Oper. Res.*, 53(3):549–559, 2005.

[6] M. Bohge and M. Renwanz. A realisitic VoIP traffic generation and evaluation tool for OMNeT++. In *OMNeT++ 2008: Proceedings of the 1st International Workshop on OMNeT++*, 2008.

[7] G. Box and G. Jenkins. *Time Series Analysis - forecasting and control*. Holden-Day, 1970.

[8] P. J. Brockwell and R. A. Davis. *Time Series: Theory and Methods*. Springer, 2nd edition, 1998.

[9] M. C. Cario and B. L. Nelson. Autoregressive to anything: Time-series input processes for simulation. *Operations Research Letters*, 19(2):51–58, 1996.

[10] M. C. Cario and B. L. Nelson. Numerical Methods for Fitting and Simulating Autoregressive-To-Anything Processes. *INFORMS J. on Computing*, 10(1):72–81, 1998.

[11] G. Fishman. *Concepts and Methods in Discrete Event Digital Simulation*. John Wiley, New York, 1973.

[12] I. D. Hill. Algorithm AS 66: The Normal Integral. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 22(3):424–427, 1973.

[13] R. Hornig and A. Varga. An Overview of the OMNeT++ Simulation Environment. In *Proc. of 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools)*, 2008.

[14] A. Horvath and M. Telek. Markovian modeling of real data traffic: Heuristic phase type and MAP fitting of heavy tailed and fractal like samples. In M. C. Calzarossa and S. Tucci, editors, *Performance 2002*, volume 2459 of *LNCS*, pages 405–434. Springer, 2002.

[15] K. V. Jonsson. HttpTools: a toolkit for simulation of web hosts in OMNeT++. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (Simutools 09)*, 2009.

[16] S. Kay. Recursive maximum likelihood estimation of autoregressive processes. *IEEE Trans. Acoust. Speech Signal Process.*, 31:56–65, 1983.

[17] W. D. Kelton, R. P. Sadowski, and D. A. Sadowski. *Simulation with Arena*. McGraw-Hill, 4 edition, 2007.

[18] J. Kriege and P. Buchholz. An Empirical Comparison of MAP Fitting Algorithms. In B. Müller-Clostermann, K. Echtle, and E. P. Rathgeb, editors, *Proceedings of the 15th International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB & DFT 2010)*, volume 5987 of *LNCS*, pages 259–273. Springer, 2010.

[19] A. M. Law and W. D. Kelton. *Simulation modeling and analysis*. Wiley, 2000.

[20] A. M. Law and M. G. McComas. Expertfit distribution-fitting software: how the expertfit distribution-fitting software can make your simulation models more valid. In *Winter Simulation Conference*, pages 169–174, 2003.

[21] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Trans. Netw.*, 2(1):1–15, 1994.

[22] D. M. Lucantoni, K. S. Meier-Hellstern, and M. F. Neuts. A single server queue with server vacations and a class of non-renewal arrival processes. *Advances in Applied Probability*, 22:676–705, 1990.

[23] H. Luetkepohl. *Introduction to Multiple Time Series Analysis*. Springer Verlag, New York, 1991.

[24] M. F. Neuts. A versatile Markovian point process. *Journal of Applied Probability*, 16:764–779, 1979.

[25] M. F. Neuts. *Matrix-geometric solutions in stochastic models*. Johns Hopkins University Press, 1981.

[26] *OMNeT++ - Discrete Event Simulation System Version 4.0 User Manual*.

[27] V. Paxson and S. Floyd. Wide-area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3:226–244, 1995.

[28] ProFiDo Homepage. `http://ls4-www.cs.tu-dortmund.de/profido/`.

[29] The Internet Traffic Archive. `http://ita.ee.lbl.gov/`.

[30] M. J. Wichura. Algorithm AS 241: The Percentage Points of the Normal Distribution. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 37(3):477–484, 1988.

[31] C. Williamson. Synthetic Traffic Generation Techniques for ATM Network Simulations. *Simulation*, 72(5):305–312, 1999.