In this survey, we review several approaches for predicting performance of NoC-based multicore systems, starting from the traffic models to the complex NoC models for latency evaluation. We first review typical traffic models to represent the application workloads in NoC. Specifically, we review Markovian and non-Markovian (e.g., self-similar or long range memory dependent) traffic models and discuss their applications on multicore platform design. Then, we review the analytical techniques to predict NoC performance under given input traffic. We investigate analytical models for average as well as maximum delay evaluation. We also review the developments and design challenges of NoC simulators. One interesting research direction in NoC performance evaluation consists of combining simulation and analytical models in order to exploit their advantages together. Towards this end, we discuss several newly proposed approaches that use hardware-based or learning-based techniques. Finally, we summarize several open problems and our perspective to address these challenges.

Categories and Subject Descriptors: A.1 [General and reference]: Introductory and survey; C.4 [Performance of systems]: Modeling techniques

Additional Key Words and Phrases: Performance evaluation, Network-on-Chips (NoCs), analytical model, average and maximum delay, simulation, traffic models

1. INTRODUCTION

With IC technology continuously shrinking down, state-of-the-art computing platforms widely use architectures such as Multi-Processor System-on-Chips (MPSoCs) and Chip Multi-Processors (CMPs); therefore, Network-on-Chips (NoCs) architectures are suggested in these designs as the future communication infrastructure to manage the information transfer among the Processing Elements (PEs) [Benini and De Micheli 2002; Dally and Towles 2001]. When designing NoC-based multi-core platforms, the latency metric usually creates a big challenge during the design space exploration [Ogras et al. 2010; Kiasari et al. 2013a]. In order to make a proper design choice, an accurate and fast evaluation of each design candidate which has different configurations is needed [Bogdan and Marculescu 2009; Ogras et al. 2010]. More specifically, to evaluate the...
performance of each feasible candidate, the designer should first characterize the application to extract a specific traffic model [Varatkar and Marculescu 2002; Bogdan and Marculescu 2011]. The traffic model needs to abstract two key features, i.e., the source and destination of every flow, as well as the packets inter-arrival time distribution. For instance, a simple representation of an application uses the application task graph [Hu and Marculescu 2003; 2005] to capture these two features (Fig 1-a shows an example of the Dual Video Object Plane Decoder (DVOPD) application from [Pullini et al. 2007; Bertozzi et al. 2005]). In this type of representation, a directed edge points from the source to the destination of each flow while the weight on the edge represents the mean traffic data rate with the Poisson inter-arrival time distribution. After pre-characterizing the application, designers then need to schedule and allocate the tasks on the proper PEs followed by the exploration of the PE core placement and routing algorithm design in the system. For each feasible design configuration, the performance analysis step needs to be performed to evaluate the quality of the design [Ogras et al. 2010] (shown in Fig. 1-b). Towards this end, both simulation-based and analytical models are used in the NoC design flow.

In general, NoC simulators attempt to model the architectural implementation details of NoC routers (i.e., both the data and control paths) and can provide estimations with very high accuracy. That is necessary to have detailed performance evaluation before prototyping. One limitation is that it usually takes a lot of time to simulate a system with large size. Moreover, it is not easy to figure out the performance bottlenecks (e.g., the choice of buffer size of a specific router) from the simulation results; which makes the simulations less powerful and efficient in design space explorations [Ogras et al. 2010; Kiasari et al. 2013a]. Because of these inefficiencies, analytical NoC models are also used in the early design stage to support fast exploration of a large design space. Compared to simulations, the accuracy in analytical models is compromised for the flexibility and speed enhancement [Ogras et al. 2010; Kiasari et al. 2013a].

In this survey, we review and summarize both the analytical and simulation approaches that have been used in evaluating NoC performance. We first notice that a specific NoC performance highly depends on the traffic/workload applied onto the system [Bogdan and Marculescu 2011]. Therefore, in section 2, we first review types of workload which are widely adopted in the literature. Then, we review the mathemat-
In this section, we first categorize the NoC workload models used in the performance evaluation. Then we summarize the methods to observe and characterize fractal and non-stationary behaviors. In particular, several representative techniques are reviewed, including the Hurst-parameter based models [Varatkar and Marculescu 2002], the phase type based models [Kuhn 2013], non-equilibrium statistical physics inspired models [Bogdan and Marculescu 2011] and multifractal models [Bogdan 2015].

2.1. NoC workload categorization

In general, there are two important aspects when describing an NoC workload [Dally and Towles 2003], i.e., the traffic patterns and the injection inter-arrival time (IAT) distributions [Bogdan and Marculescu 2011]. The former describes the distribution of the source and destination nodes of every flow in the topology. The latter dictates when to issue (inject) a packet from the source Processing Element (PE) into the network or the packets inter-arrival processes at the intermediate router buffer channels, so it represents the temporal characteristics of the traffic [Soteriou et al. 2006]. In Fig. 2, we categorize various NoC workloads based on these two features (i.e., spatial patterns and temporal injection processes). The terminology in the figure follows the conventions and descriptions in [Dally and Towles 2003]. Specifically, for the traffic patterns,
we classify them into synthetic and application-driven patterns. The synthetic traffic patterns use mathematical methods to create the source and sink PE addresses; while application-driven patterns determine the destination of each flow from the mapping results of the applications running on target CMP/MPSoC platforms. For the injection processes, mathematical models (e.g., Poisson or Fractal models) can be used to dictate the packets IAT distributions. Moreover, the packets IAT can also be extracted from real applications. For example, designers can use trace logs or fully execute the application to determine when to issue the next packet in a flow.

To evaluate an NoC design, the most accurate workloads should describe both the traffic pattern and injection method from real applications. In the table of Fig. 2, the crossed entry of realistic pattern (column) and injection method (row) is named as application-driven workload. It can be further classified into execution-driven and trace-driven workloads. For the execution-driven workload, the traffic is produced by modelling the PEs together with the NoC. Specifically, the processor executes the whole application program and decides when to issue a packet at run time. Because such injection method requires a detailed model of both the processors and NoC, this type of evaluation is also named as full-system simulation. In summary, the full-system simulation achieves the highest accuracy; however it takes a very long evaluation time. Moreover, this methodology is not very flexible because if a parameter (e.g., number of virtual channels or buffer size) changes, the whole evaluation needs to be performed again; this introduces additional timing overhead to explore the design space [Ogras et al. 2010]. To improve the speed, the trace-based workload has also been used. In trace-based workload, after performing the full-system simulation once, the traffic traces (e.g., a file recording the exact time/cycle when a PE injects a cache-coherent or request/response packet) are logged. Then, during subsequent simulations, the packets are issued into the network following the trace logs. The execution models of the processors are no longer required.

One limitation of using the application-driven workloads for MPSoC platforms is the users need to provide detailed application information at the very beginning. While for general purpose CMP platforms, there already exists a number of benchmarks, such as Parsec [PARSEC 2009], Splash-2 [SPLASH-2 1995] and Spec [Spradling 2007] suite, to provide representative applications in different domains (e.g., high performance computing, image/video processing). However, it is difficult to extrapolate the performance from these benchmarks to a new application class which may have very different characteristics (e.g., different levels of burstiness) [Soteriou et al. 2006; Gratz and Keckler 2010]. Another limitation of trace-based workloads is that the dependency among the packets may be changed after applying them on a new design [Hestness and Keckler 2010]. Therefore, researchers have found it is better to pre-analyze the collected traces before applying them in the new evaluations. The Netrace tool tries to address this dependency problem and improve the evaluation accuracy [Hestness and Keckler 2010]. This is achieved by performing a dependency analysis on the traces. The identified dependency relationship among packets is kept in the subsequent network-only simulation by preserving the order of packets that is injected in NoC. Similarly, in [Mahadevan et al. 2005], the dependency among the packets is kept by making a new transaction happen only after the PE receiving all responses from its dependent processors.

As shown in Fig. 2, in addition to application-driven workloads, mathematical (or statistical) models are also widely used to describe the traffic patterns or packet inter-arrival times. For the traffic patterns, they can either be abstracted from applications and represented in a graph or be synthetically created without using any application information. For the packet inter-arrival times, they can be described using different models (e.g., Poisson or self-similar models). One advantage of mathematical injection
methods is that by controlling some parameters (e.g., the Hurst parameter) in the model, a variety of traffic inputs with user-desired property (e.g., self-similarity) can be created. In the following, we review these traffic models. In particular, we elaborate on the memory dependent models (e.g., fractal or self-similar) in NoC traffic analysis.

2.2. Traffic pattern characterization
To describe the distributions of the traffic sources and destinations, the application task graphs and synthetic traffic patterns are two widely used methods [Marculescu and Bogdan 2009]. In Fig. 2, they belong to the application-driven and synthetic pattern, respectively. Specifically, the task graphs are structures used to represent some profiled applications. In a task graph $G = (C, A)$, the vertex set $C$ represents all computation tasks and each directed edge $a_{i,j}$ in $A$ represents the communication flow from vertex $c_i$ to $c_j$. The weights on the edge characterize the communication data volume (bits) between tasks [Hu and Marculescu 2003]. After mapping tasks onto proper PEs, the architecture characterization graph (ARCG) $ARCG = (T, P)$ further describes the communications among the PEs in NoC [Hu and Marculescu 2003]. There are several popular task-graph based benchmarks characterizing different multimedia and communication applications in NoC, including the MMS (Multimedia system) [Hu and Marculescu 2004b], PIP (Picture in picture), MWD (Multi-window detection), MPEG (MPEG decoder) [Bertozzi et al. 2005], DVOPD (Dual Video Object Plane Decoder) [Pullini et al. 2007], H264D (H264 decoder) [Liu et al. 2011] and LDPC (Low density parity check encoder/decoder) [Liu et al. 2011; Wang et al. 2014] applications.

In addition to using task graphs, synthetic traffic generators, which operate on the source and target PE addresses to produce a variety of artificial patterns, are also used. For example, in random traffic pattern, every PE in the system can choose a destination with same probability. This in general creates uniform traffic loads across different channels in the system. For uneven traffic, permutations on the node addresses are used [Gratz and Keckler 2010]. For instance, to obtain the destination address, the permutation can be performed by exchanging the X- and Y- coordinates of the source address (i.e., transpose traffic); it can also be performed in bit-level, such as reversing order of bits (i.e., bit-reversal), or complementing each bit (i.e., bit-complement) [Dally and Towles 2003]. Compared to task graphs and application-driven workloads, synthetic traffic patterns provide more artificial traffic scenarios to evaluate a design [Gratz and Keckler 2010].

As shown in Fig. 2, for both task-graph based and synthetic-based patterns, their injection processes can be memoryless or memory-dependent. To describe the packet injection process, let a random variable $x$ represent the packet inter-arrival time (IAT) of the target flow. The simplest model to describe the distribution of $x$ is based on Poisson injection process, which means $x$ follows a negative exponential distribution, i.e., $P_x(t) = \lambda e^{-\lambda t}$, where $P_x(t)$ is the probability density function and $\lambda$ is the mean packet rate (packets/cycle). In general, Poisson traffic models belong to the type of memoryless workload, which means the probability distribution does not depend on previous states. One limitation is that it cannot reflect the burstiness and packet dependencies [Wu et al. 2010; Kouvatsos et al. 2005]. Therefore, statistical techniques which considers non-Poisson IAT distributions are needed. Specifically, these models are built upon fractal/self-similar traffic models to capture the short-range and long-range memory dependencies among the IATs. In the following section, we survey these techniques.

2.3. Techniques to model and analyze mono- and multi-fractal traffic
In this section, we summarize the techniques to model fractal/self-similar NoC traffic. To unify the symbols of different approaches, based on the parameter conventions, the notations in Table I are used in the presentation of this section.
A random process which is either continuous time (with index variable $t$) or discrete time (with index variable $n$).  

The variance of the random process $X$ is defined as $\text{Var}(X) = E[(X - E(X))^2]$, which is a function of parameter $m$.  

The spectral density function of $X$, denoted as $S_X(w)$, is defined as $S_X(w) = \lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{T} \{X(t)\}^* \{X(t - \tau)\} d\tau$, where $\{X(t)\}$ denotes the complex conjugate of $X(t)$.

A random variable $(X(n))$ is the $n$th indexed random variable in the random process $X$.  

$X^n$ is the $n$th indexed random variable in process $X^m$.  

$E(X)$ is the expectation (mean) of the random process $X$.  

$\sigma^2(X)$ is the standard deviation of the random process $X$.  

$\sigma^2(X^m)/\text{Var}(m)$ is the variance of the random process $X$ averaged over $m$ consecutive data in $X$.

$X^n$ is the $n$th indexed random variable in process $X^m$.  

$E(X)$ is the expectation (mean) of the random process $X$.  

$\sigma^2(X)$ is the standard deviation of the random process $X$.  

$\sigma^2(X^m)/\text{Var}(m)$ is the variance of the random process $X$ averaged over $m$ consecutive data in $X$.

In section 2.3.1, we assume $X$ is second-order stationary (wide-sense stationary). Second-order stationary means the first two moments of $X$ (mean and co-variance) do not depend on specific index variable $t$ (for continuous time) or $n$ (for discrete time).

### 2.3.1. Review of LRD/fractal/self-similar traffic

Self-similarity, fractal and long-range dependence (LRD) are concepts that have been developed from observing the natural data sequences with memory dependencies between two different time instances or indexes [Bogdan et al. 2010; Yoshihara et al. 2001; Ryu and Lowen 2000]. There are several ways to formally characterize self-similar/fractal/LRD traffic flows. For example, suppose a set of random variables is used to represent the packet arrivals from two different time instances or indexes. This random variable sequence forms a discrete time random process $X$. The random variable $X(n)$ with an index $n$ represents the IAT between the $n$th and $(n + 1)$th packet. When analyzing such random process $X$, we usually start from wide-sense stationary (or namely second-order stationary) assumption [Yoshihara et al. 2001], which means the first (i.e., mean) and second moment (i.e., co-variance) of $X$ does not depend on specific index choices of the random variables. For example, for a discrete time random process $X$, by definition, the mean of $X$ should be a function of $n$ where the $n$th element corresponds to the expected value of $X(n)$. Under the wide-sense stationary condition, the mean at different indexes are the same. Therefore it can be represented by a single parameter $E(X)$.

Based on above assumption, the definition “long-range-dependency” comes from observing the correlations between two random variables in $X$, whose indexes are $l$ lags away. Formally, the autocorrelation function $r_X(l)$ of $X$ is defined as [Varatkar and Marculescu 2004; Park and Willinger 2000]:

$$r_X(l) = \frac{E[(X(n) - E(X))(X(n + l) - E(X))]}{\sigma^2(X)}$$

(1)

where $X(n)$ and $X(n + l)$ are two random variables at index $n$ and $n + l$, respectively. Under second-order stationary condition, the auto-correlation $r_X(l)$ only depends on $l$ and does not rely on the choices of instance $n$.

Using Eqn. 1, $r_X(l)$ can be computed with respect to different choices of value $l$. Intuitively, $r_X(l)$ reveals how current data is affected by its neighbors that are $l$ lags away. For short-range dependency (SRD) series, $r_X(l)$ decreases exponentially with $l$; while for long-range dependent (LRD) random process, $r_X(l)$ decreases much slowly.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>A random variable (e.g., represents the packet inter-arrival time (IAT))</td>
</tr>
<tr>
<td>$P_x(t)$</td>
<td>Probability density function (PDF) of $x$, i.e., probability density of $x = t$</td>
</tr>
<tr>
<td>$F_x(T)$</td>
<td>Cumulative distribution function (CDF) of $x$, i.e., probability of $x \leq T$</td>
</tr>
<tr>
<td>$X$</td>
<td>A random process which is either continuous time (with index variable $t$) or discrete time (with index variable $n$)</td>
</tr>
<tr>
<td>$X(n)$</td>
<td>The $n$th indexed random variable in the random process $X$</td>
</tr>
<tr>
<td>$X^m$</td>
<td>A discrete time random process, whose element is the random variable averaged over $m$ consecutive data in $X$</td>
</tr>
<tr>
<td>$X^n$</td>
<td>The $n$th indexed random variable in process $X^m$</td>
</tr>
<tr>
<td>$E(X)$</td>
<td>Expectation (mean) of the random process $X$</td>
</tr>
<tr>
<td>$\sigma^2(X)$</td>
<td>Variance of the random process $X$; $\sigma(X)$ is the standard derivation</td>
</tr>
<tr>
<td>$\sigma^2(X^m)/\text{Var}(m)$</td>
<td>Variance of the random process $X^m$, which is a function of parameter $m$</td>
</tr>
<tr>
<td>$r_X(l)$</td>
<td>Autocorrelation function of random process $X$ under the lag size $l$</td>
</tr>
<tr>
<td>$f_X(w)/f_X(w,H)$</td>
<td>Spectral density function of $X$; $w$ is the frequency parameter; $H$ is hurst parameter</td>
</tr>
<tr>
<td>$m(t)$</td>
<td>A random variable represents the number of packets generated during interval $[0, t]$</td>
</tr>
<tr>
<td>$N_i(t)$</td>
<td>Average number of router channels in NoC that have $i$ packets at time $t$</td>
</tr>
</tbody>
</table>

1 In section 2.3.1, we assume $X$ is second-order stationary (wide-sense stationary). Second-order stationary means the first two moments of $X$ (mean and co-variance) does not depend on specific index variable $t$ (for continuous time) or $n$ (for discrete time).
Theoretical derivations have shown a close to power law decreasing with respect to the leg length [Varatkar and Marculescu 2004; Park and Willinger 2000].

Besides examining the autocorrelation function $r_X(\ell)$, the dependency among the data samples can also be presented by the concepts of "self-similarity" or "fractal". These two definitions come from observing the random process $X^m$ under different scale levels [Varatkar and Marculescu 2004]. More precisely, in these definitions, for each scale level $m$, where $m$ is an integer value chosen by the observer, a new process $X^m$ can be produced by making each indexed random variable $X^m(n)$ equal to:

$$X^m(n) = \sum_{i=1}^{n} X((n-1)m + i)/m.$$  

The variance of $X^m$ is then calculated and is a function of the scale-level $m$. In Table I, this variance $\sigma^2(X^m)$ is also represented as $Var(m)$. Of note, when calculating $Var(m)$, for independent or SRD sequence $X$, by taking the average of every $m$ samples in $X$, it is very likely the data is smoothened. Consequently the data variance of $X^m$ should decrease very fast. Typically, $Var(m)$ decays with $m$ exponentially. On the other hand, for self-similar $X$, the increasing of scale level $m$ does not significantly reduce the variance of the new sequence $X^m$. This is reflected in $Var(m)$, which decreases much slowly as: $Var(m) \sim m^{-\beta}$, where $\beta$ typically locates in the range $(0, 1)$ [Varatkar and Marculescu 2002].

The third way to describe a LRD process $X$ is to transfer and observe the series in the frequency domain. Formally, let $f(w)$ represent the spectral density of the random process $X$; for fractal/self-similar $X$, it has been shown $f(w) \sim bw^{-\gamma}$ when $w \to 0$ [Varatkar and Marculescu 2004].

Based on above discussions, two methods have been widely used to examine whether a time series is long-range dependent or self-similar 1. The detail derivations are provided in [Varatkar and Marculescu 2002, 2004; Min and Ould-Khaoua 2004]. Here, we just highlight the ideas in those works as follows: The first method is the variance-time analysis which plots the curve of $\log(Var(m))$ against $\log(m)$. For LRD/self-similar process $X$, it has been shown $\log(Var(m))$ decreases linearly with $\log(m)$ and the slope $-\beta$ satisfies $0 < \beta < 1$. The second method is based on Hurst effect. Specifically, it calculates the "rescaled adjusted range statistics" (i.e., R/S statistics [Qian and Rasheed 2004]) of the random process $X$. Then, the Hurst parameter $H$ is calculated as the changing rate of the R/S statistics with respect to the data sequence size $n$. By examining the value of $H$, the dependencies among the time series can be described quantitatively. For memoryless time series, $H = 0.5$; while for LRD sequences, $0.5 < H < 1$. Moreover, it is observed the Hurst parameter $H$ in R/S method equals to $1 - \beta/2$ in the variance-time method.

2.3.2. Generative traffic models for self-similar traffic. The variance time and R/S statistic methods are useful in analyzing time sequences. However, they can not be used to generate self-similar traffic. To produce fractal traffic, two kinds of models are widely used. The first type utilizes only one parameter (e.g., the Hurst parameter $H$) to represent the dependency and self-similarity (e.g., [Varatkar and Marculescu 2004] and [Soteriou et al. 2006]). While the second type is based on "Phase method" which uses different phases to describe the packets generation process [Kuhn 2013]. Examples are the Generalized-exponential (GP) process [Wu et al. 2010], Markov-modulated Poisson process (MMPP) [Kiasari et al. 2013b; Fischer and Meier-Hellstern 1993] and a more generalized Markov arrival process (MAP) [Diamond and Alfa 2000; Klemm et al. 2002]. In the following, we summarize the principles of these two models, respectively.

---

1 For non-stationary random processes, the "detrended fluctuation analysis" (DFA) [Kantelhardt et al. 2002] should be applied, which uses a regression function to pre-fit the temporal trend over the data. The readers can find more details and a step-by-step tutorial of implementing DFA in matlab in [Ihlen 2012]
Hurst parameter-based modeling: In [Varatkar and Marculescu 2002; 2004], the authors for the first time propose to consider the dependencies in NoC traffic by using the Hurst parameter $H$. To produce a synthetic packet sequence with a user-needed self-similarity level $H$ ($0.5 < H < 1$), Fractional Gaussian Noise (FGN) model [J.Beran 1994] is used. Their procedures of applying FGN model to generate synthetic NoC traffic traces is shown in Fig. 3. We summarized their methods as follows: the inputs to the generative model include the user-specified $H$ value, the length of the total sequence $n$ and the desired packet inter-arrival time (IAT) distribution $F_x(t)$. The first step of the overall procedure is to produce a data series that has a self-similarity level $H$; this is done via sampling the FGN spectrum $f(w, H)$ ($w$ is the frequency component), which is given by [Paxson 1997]:

$$f(w, H) = A(w, H) \times [ |w|^{-2H-1} + B(w, H)] (H \in (0, 1); w \in (-\pi, \pi))$$  \hspace{1cm} (2)

In Eqn. 2, $A(w, H)$ and $B(w, H)$ are specific frequency functions whose closed-form are derived in [Paxson 1997]. In order to obtain $f(w, H)$, it requires to compute a summation of infinite terms due to the existence of function $B(w, H)$. Following the steps in [Paxson 1997], a simple approximation of $f(w, H)$ with finite terms of summation is obtained and is denoted as $g(w, H)$. The continuous power spectrum $g(w, H)$ is then used as the input of the subsequent sampling procedure (i.e., Step 2 in Fig. 3). After sampling, a Discrete Fourier Transform (DFT) spectrum with $n/2$ points is obtained. Based on the symmetric property of the spectrum, the DFT components can be mirrored and extended to the size of $n$. Transforming these $n$ points in the frequency domain back via ‘inverse Discrete Time Fourier Transform’ (IDTFT) creates an $n$-point time series $X^*$ (Step 4). This time series has a desired Hurst parameter $H$. However, it is noticed in the previous sampling step, since the FGN spectrum is used, the output data sequence actually has a Gaussian distribution shape whose mean equals to zero. Consequently, we need to match the distribution of $X^*$ with the desired cumulative distribution function $F_x(t)$. To achieve this, the following transformation is used to create the $n$-th data $X(n)$ from $X^*(n)$ [Varatkar and Marculescu 2004]:

$$X(n) = F_x^{-1}(F_{x^*}(x^*(n)))$$  \hspace{1cm} (3)

where $F_x(t)$ is the cumulative distribution function (CDF) of the sequence $X^*$ after step 4. $X$ is the created new time series with $X(n)$ denoting the $n$-th packet IAT. The generated traffic trace can then be fed into NoC simulators to evaluate the latency performance under the impacts of fractal traffic inputs.

2) Phase-type random process based traffic models: Phase-type models belong to the type of generative methods which are widely adopted in queuing analysis to reproduce the customer arrival or service process using multiple stages (phases) [Klein-
be modelled as Markovian Modulated Poisson Process (MMPP). In these works, an
in burst with zero intervals), different level of arrival burstiness can be modelled.

In MMPP traffic models, a function \( n(t) \) which represents the total number of pack-
et generated during time interval \((0, t]\) is used to determine \( \lambda_i \) of each state and \( q_{ij} \) in matrix \( Q \) [Yoshihara et al. 2001]. Specifically, to fit the traffic traces using a 2-state MMPP-based traffic model, the mean \( E(n(t)) \), the variance \( \sigma^2(n(t)) \) as well as the Index of Dispersion Count (IDC) \( IDC(n(t)) \) of the random function \( n(t) \) are measured for the application traffic first; then, the fitting procedure can be performed by matching the measured statistics with the closed-form formula of these metrics derived in a stochastic 2-state MMPP model [Shahram and Tho 1998]. To better capture the traffic self-similarity, multiple Markovian state-based modelling techniques were developed in which the traffic is modelled by the mixture of several interrupted Poisson processes (IPPs) [Yoshihara et al. 2001; Min and Ould-Khaoua 2004].

In [Yoshihara et al. 2001], a multiple state MMPP traffic model was developed which consists of \( d \) (\( d > 1 \)) IPPs. To be more specific, the \( d \)-state MMPP traffic model is ob-
tained by aggregating several \( 2 \)-state MMPP processes described previously. Compared
to the case of a \( 2 \)-state MMPP, there are more parameters need to be derived in the
\( d \)-state model. The traffic trace is pre-processed under \( d \) different time scales. For each scale level \( m \) (e.g., \( m = 1, 2, ..., d \)), a new data series is generated by averaging the original sequence with a time window size \( m \). Then, the parameter fitting procedure is performed on the obtained \( d \) sequences separately.

Besides MMPP models, a more generalized Markov Arrival Process (MAP) has also be
used in traffic analysis [Klemm et al. 2002]. In summary, the fitting algorithm of
such traffic model is still based on matching the statistic moments which are obtained analytically with the measurements in given traffic traces. For example, in [Casale et al. 2008], a MAP traffic model is obtained by matching the first three moments of IAT as well as the correlation between \( X(n) \) and \( X(n + 1) \) (i.e., \( E[X(n) \times X(n + 1)] \)) in the analytical model with the real traffic measurements.

Compared to the Hurst-parameter-based traffic modeling, phase-type based traffic
models are compatible with many existing queuing models (such as the MMPP/G/1
Fig. 4. An example illustrating the non-stationary traffic characteristics in a $4 \times 4$ mesh NoC (From [Bogdan and Marculescu 2010; 2011]): a) for a channel port, the packet header flit inter-arrival time distribution is shown; the IAT is compared with Poisson process b) the plot of third order moment versus two time lags derived from the FGN model c) the actual third order moment calculated from the application traffic and MAP/G/1 queues [Min and Ould-Khaoua 2004]). They can be used in a performance model to evaluate NoC latency under specific traffic characteristics. However, for highly LRD traffic models, it is usually very difficult to obtain a closed-form solution in queuing theory. Moreover, most phase-type traffic models are based on stationary or wide sense stationary IAT assumption. Therefore, they do not well model the non-stationary inter-arrival processes. In the next sections, we survey the techniques proposed for multi-fractal NoC traffic analysis.

2.3.3 Motivations for multi-fractal traffic modeling. Many existing NoC traffic models assume that the random process $X$ is wide-sense stationary (WSS). Consequently, the traffic models built upon techniques such as FGN (shown in Fig. 3) can be classified as mono-fractal analysis, where the traffic correlations between time instance $t$ and $t + \delta t$ can be represented using a single and unified parameter $H$ [Bogdan and Marculescu 2010]: $|X(t + \delta t) - X(t)|^q \sim (\delta t)^q H$. Recently, some profiling results of real NoC applications show a different non-stationary behavior due to the dependencies among different PEs as well as the request-response correlations of the packets [Bogdan and Marculescu 2011; 2010]. As a result, many traffic traces are non-stationary and can not be fully characterized by a single Hurst exponent. Instead, they are better characterized by a set of scaling exponents [Bogdan and Marculescu 2010]: $|X(t + \delta t) - X(t)|^q \sim (\delta t)^{g(q)}$, where $q$ is the moment index described in [Bogdan and Marculescu 2011; 2010] and $g(q)$ is a function of $q$ which is non-linear in general.

Based on the above observations, one interesting and on-going research direction on NoC traffic modeling is to extend beyond the existing single exponent (i.e., mono-fractal) traffic model to a more accurate multi-fractal model.

In Fig. 4-a, an example is shown which compares the Poisson, mono- and multi-fractal traffic models [Bogdan and Marculescu 2010; 2011]. In this example, a sequence of packet inter-arrival times (IAT) is used for analysis. From Fig. 4, it shows that when observing a specific router in NoC, the IAT may have a long tail form while conventional Poisson model cannot capture this kind of long-range dependency. To capture the burstiness, a Fractal Gaussian Noise (FGN) model described in Section 2.3.1 can be employed. That model is usually accurate in matching the first two moments of the input traffic trace. However, [Bogdan and Marculescu 2010] further compared the

\[The \ IAT \ distribution \ is \ plotted \ for \ the \ west \ input \ port \ of \ a \ node \ in \ a \ 4 \times 4 \ mesh \ NoC \ under \ the \ execution \ of \ a \ multi-threaded \ NoC \ application \ [Bogdan \ and \ Marculescu \ 2010; \ 2011].\]
third order cumulant $C_3$ which is defined as:

$$C_3 = \frac{X(t_1)X(t_2)X(t_3) - X(t_1)X(t_2)X(t_3) - X(t_2)X(t_1)X(t_3)}{X(t_3)X(t_2)X(t_1) + 2X(t_1)X(t_2)X(t_3)}$$

where $t_1$, $t_2$ and $t_3$ are three instances of the overall data sequence $X_t$. For an IAT distribution fitted by a Gaussian random process, the plot of $C_3$ with respect to two intervals $t_2 - t_1$ and $t_3 - t_1$ (i.e., time Lag 1 and Lag 2 in the figure) is shown in Fig. 4-b. In Fig. 4-b, $C_3$ is zero across different combinations of $t_2 - t_1$ and $t_3 - t_1$ in FGN model. However, the actual third order moment of the IAT is also plotted in Fig. 4-c. It can be observed $C_3$ of the actual trace collected from simulations has several peaks and is non-uniform in general, which are not captured by the mono-fractal Gaussian model. Because of this, it motivates to develop a new multi-fractal analysis framework for non-stationary traffic modelling.

2.3.4. Multifractal spectrum of real applications. One useful method to identify a multifractal data series from mono-fractal sequence is to build the multifractal spectrum (or namely singularity spectrum), which plots the singularity dimension $D(h)$ against the holder exponent $h$ [Lopes and Betrouni 2009]. Intuitively, the holder exponent $h$ dictates an additional $(t-t_i)^h$ term when using expansions to approximate the data series around point $t_i$; while $D(h)$ is the fractal dimension of the data set that is made up of the points having the same $h$ [Physionet 2004]. For mono- or memoryless series, the multifractal spectrum has a narrow width. For multifractal sequences, the multifractal spectrum spans a variety of $h$ values [Ihlen 2012]. In [Bogdan 2015], the author plotted the multifractal spectrum for two realistic NoC applications (shown in Fig. 5). Specifically, Fig. 5-a is the multifractal spectrum observed from three cores in Intel Single-Chip Cloud Computer (SCC) platform executing a SPEC MPI application. Fig. 5-b is the spectrum for a $8 \times 8$ mesh NoC running a PARSEC application. Similar observations have been reported for different PARSEC applications [Bogdan and Xue 2015]. From these spectra, we can conclude many realistic NoC applications indeed have multifractal characteristics. A set of exponents can better characterize the traffic behaviors instead of a single Hurst parameter.
2.3.5. Modeling the system dynamics using mean-field approach. To address the non-stationary property, a mean field (MF) traffic modelling technique is used in [Bogdan and Marculescu 2011; 2010]. In summary of their approach, the packet transmission in NoC is modelled using a random graph (RG). In a RG, the nodes represent the buffer channels in the router and the edges represent the application packets that are transferred between the RG nodes. At run time, the IN and OUT degree of each node in the RG correspond to the number of incoming and outgoing customers (packets), respectively. Of note, the randomness of the graph refers to the connections in the graph will change dynamically depending on the arrival and departure of packets at the channel [Bogdan and Marculescu 2011]. Based on these concepts, the authors derive that the IN degree of nodes in RG satisfies the following equation [Bogdan and Marculescu 2011; 2010]:

\[
\frac{\partial N_i}{\partial t} = \frac{p \eta_i(t)}{M_i(t)} f_1(N_{i-1}, N_i) - \frac{r \theta_i(t)}{Z_i(t)} f_2(N_i, N_{i+1})
\] (5)

where \(N_i(t)\) represents the average number of buffer channels at time \(t\) whose IN degree equals to \(i\) (i.e., having \(i\) arrival packets). In Eqn. 5, the first term considers the effects of packets arriving at a buffer channel. Specifically, by the definition of the RG, an edge is connected to a node if a packet reaches the corresponding buffer node. Therefore, at time \(t\), if a packet arrives at any one of the \(N_{i-1}\) nodes which have an IN degree \(i-1\), then \(N_i\) should increase by one at \(t + \delta t\). On the other hand, if a packet arrives at any one of \(N_i\) nodes that already have \(i\) packets, \(N_i\) will decrease by one and \(N_{i+1}\) is augmented by one as a result. Thus, the function \(f_1(N_{i-1}, N_i)\) in Eqn. 5 models the packet arrival effects by first adding the number of new nodes in RG that enter into \(N_i\) state from \(N_{i-1}\) and then subtracting the number of nodes that leave from \(N_i\) to \(N_{i+1}\). In Eqn. 5, \(p \eta_i(t)/M_i(t)\) is a specific time-dependent function which denotes the probability of a new packet arriving at one node in RG at time instance \(t\). Similar to the first part of Eqn. 5, the second term in Eqn. 5 considers the effects of packets departing from the current router channel. \(r \theta_i(t)/Z_i(t)\) in Eqn. 5 represents a time-dependent probability that a packet will leave the current node at time instance \(t\). The function \(f_2(N_i, N_{i+1})\) considers two cases: if a packet leaves node with IN degree \(N_{i+1}\), then \(N_i\) increases; on contrary, if a packet leaves node with IN degree \(N_{i}\), then \(N_i\) decreases. In \(p \eta_i(t)/M_i(t)\) and \(r \theta_i(t)/Z_i(t)\) of Eqn. 5, the factors \(p\) and \(r\) represent the probabilities of a packet arriving at or departing from a buffer channel in RG. They can be pre-characterized from the obtained packet injection traces and the routing algorithms; \(M_i(t)\) and \(Z_i(t)\) are two normalization parameters for \(p \eta_i(t)\) and \(r \theta_i(t)\) which are given in [Bogdan and Marculescu 2011; 2010]. Finally, the two time-dependent fitness functions \(\eta_i(t)\) and \(\theta_i(t)\) play the key roles on fitting the final system traffic. In [Bogdan and Marculescu 2009], these two fitness functions are developed in analogy to the energy fitness functions in statistical physics.

Based on Eqn. 5, given the pre-assumed functions \(\eta\) and \(\theta\), the time-dependent probability \(P(i, t|\eta, \theta)\) which describes the possibility of finding a node in RG that has an IN degree \(i\) at time instance \(t\), can be calculated as [Bogdan and Marculescu 2010]:

\[
\frac{\partial [t^\beta P(i, t|\eta, \theta)]}{\partial t} = \left[ \frac{\eta}{M} + \frac{\theta}{Z} \right] \frac{\partial^2 [iP(i, t|\eta, \theta)]}{\partial t^2} + \left[ \frac{\theta}{Z} - \frac{\eta}{M} \right] \frac{\partial [iP(i, t|\eta, \theta)]}{\partial t}
\] (6)

For \(\beta = 0\), Eqn. 6 is reduced to the form of a conventional Poisson traffic model. On the other hand, when \(\beta \neq 0\), Eqn. 6 is able to represent a statistical process which displays multifractal characteristics.

\(^{3}\)Assume \(\delta t\) is the infinitesimal time interval
Fig. 6. Summary of NoC traffic models. The short-range and long-range dependencies are identified by the autocorrelation function $r_X(l)$ defined in Table I. The mono- and multi-fractal traffic differ in the singularity spectrum $D(h)$, which is a function of singularity exponent $h$ [Ihlen 2012].

By solving the equations as Eqns 5 and 6, users can predict the system dynamics which evolve with time. This consequently enables a more accurate run-time control over the system. For example, in [Bogdan and Xue 2015] and [Bogdan 2015], the authors demonstrate a framework using model predictive control (MPC) for run-time power management. The multi-fractal workload model and system dynamic equations are developed first. Then, based on the predictions on the system dynamics, the controller aims to adjust the voltage/frequency of each tile at run time to reduce the power consumption. The authors compared the multifractal control with Poisson and mono-fractal approaches. It is observed the multifractal control framework accurately captures the system dynamics and therefore significantly saves system energy.

2.3.6. Summary of the traffic analysis techniques. In Fig. 6, we summarize different NoC traffic models discussed in previous sub-sections. In general, we can first compute the autocorrelation function $r_X(l)$ of a time sequence. For memoryless or short range dependent traffic, $r_X(l)$ rapidly reduces as $l$ increases. On the other hand, for long range dependent traffic, $r_X(l)$ has a long tail. The synthetic workload with Poisson injection process is an example of memoryless traffic. For such traffic, conventional $M/M/1$ queuing model or diffusion approximation approaches [Kobayashi 1974] can be used in the performance analysis. For the long-range dependent (LRD) traffic model, we need to further consider the mono- and multi-fractal property by comparing the multi-fractal spectrum $D(h)$. In summary, the Hurst-parameter-based and phase-type-based models discussed in Section 2 use a single exponent to characterize the workload. They belong to the mono-fractal model. On the other hand, many real applications such as those in PARSEC and SPEC benchmarks have a more complex multi-fractal behavior as shown in Fig. 5. One limitation of current multi-fractal traffic model is that most analytical performance models cannot directly take it as input. Therefore, a new performance analysis framework supporting multi-fractal traffic and can be embedded in the design space exploration is required.

3. ANALYTICAL MODELS FOR NOC
In this section, we first investigate the queuing-theory-based models for mean end-to-end delay prediction in NoC. Then, we review the approaches to derive the maximum delay bounds. For a clear presentation, the parameters used in the models and the cor-
Table II. Parameters in the analytical performance models [Qian et al. 2015; Ben-Itzhak et al. 2011]

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description and Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{s,d}$</td>
<td>Delay of a flow from the Processing Element (PE) $s$ to PE $d$</td>
</tr>
<tr>
<td>$\tau_{s,d}$</td>
<td>Waiting times at the injection buffer of PE $s$</td>
</tr>
<tr>
<td>$\eta_{s,d}$</td>
<td>Total time spent to transfer a flit from PE $s$ to PE $d$</td>
</tr>
<tr>
<td>$h_{s,d}$</td>
<td>Accumulated contention delays for the flow which is from source $s$ to sink $d$</td>
</tr>
<tr>
<td>$l^i_f$</td>
<td>The $i^{th}$ channel that flow $f$ traverses sequentially during the routing</td>
</tr>
<tr>
<td>$\lambda_f$</td>
<td>The aggregated flow arrival rate (packets/cycle)</td>
</tr>
<tr>
<td>$F_l$</td>
<td>The set that contains all the flows routing over the link $l$</td>
</tr>
<tr>
<td>$d(f,l,i)$</td>
<td>The $i^{th}$ downstream channel which $f$ traverses starting from the link $l$</td>
</tr>
<tr>
<td>$s/s_l$</td>
<td>Average service time of all packets traversing the channel $l$</td>
</tr>
<tr>
<td>$s_l^f$</td>
<td>Average flit service time in the buffer of channel $l$</td>
</tr>
<tr>
<td>$R^2_l$</td>
<td>Second moment (SCV) of the packet service time for link $l$</td>
</tr>
<tr>
<td>$u_{l,i}$</td>
<td>Maximum duration of packets in flow $f$ leaves link $l^i$</td>
</tr>
</tbody>
</table>

Fig. 7. Extracting an equivalent queuing system for each channel in NoC routers a) a typical single channel wormhole router architecture [Dally and Towles 2003] b) each input port channel can be treated as a queuing system by weighted averaging the service time of flows routing towards different output directions [Ogras et al. 2010; Hu and Kleinrock 1997; Kiasari et al. 2013b] 

3.1. Average-case performance evaluation

For most multi-core systems which do not have a deadline requirement (e.g., a general-purpose computing platform that provides best-effort services), the average-case metric is used for design space exploration [Ogras et al. 2010]. To evaluate the mean latency, the queuing theory-based models [Lysne 1998; Ogras et al. 2010; Hu and Kleinrock 1997; Fischer and Fettweis 2013] are most widely used. In the following, we summarize the basic ideas of queuing models.

3.1.1. The taxonomy of the queuing models. In Fig. 7, we first illustrate how to extract a queuing system from an NoC router during the queueing analysis. Specifically, each input buffer channel (e.g., the north input buffer highlighted in Fig. 7-a) is abstracted as a queuing system. The customers of this queuing system are packets or flits being pro-
cessed in the router. Because every customer at the current link may be routed to different output directions, the service process is usually approximated by the weighted average of the service times towards each output direction, where the weights are the amount of traffic (shown in Fig. 7-b) [Ogras et al. 2010; Kiasari et al. 2013b]. For such queuing system, the Kendall’s representation [Kiasari et al. 2013a; G. Bolch and Trivedi 2006] uses the ”A/B/m/K” abbreviation to characterize the system as follows:

— ”A” represents the arrival process: For example, the abbreviation ”M” of ”A” used in the queuing model stands for the Markovian arrival (i.e., Poisson arrival) process. ”Er” stands for Erlange arrival while ”G” corresponds to a general independent IAT distribution. Of note, some bursty arrival processes reviewed in previous sections have also been considered in NoC queuing models. Examples are the MMPP arrival process in [Min and Ould-Khaoua 2004] and the ”GE” (generalized exponential distribution) arrival process in [Qian et al. 2014; Wu et al. 2010].

— ”B” denotes the system service process. Similar to the arrival process, ”B” can be markovian with the abbreviation ”M”, deterministic with the abbreviation ”D” or a more general process with the abbreviation ”G”.

— ”m” is the total number of available servers in the queuing system. For wormhole NoCs with a single channel per port 4, since there is only one physical channel in the downstream router to serve the packets, the queuing system has only one server and m = 1 [Hu and Kleinrock 1997]. On the other hand, for NoCs with multiple virtual channels, the packets actually contend for one of the available VCs in the routing. Therefore, the effective number of servers will be larger than one depending on the available VCs [Ben-Itzhak et al. 2011].

— ”K” represents the number of customers that can be held in the queuing system. Existing NoC analytical models define the customer in the queuing system as either a single packet or a flit. Consequently, ”K” is calculated from the router buffer size based on the customer granularity.

In queuing models, when computing the waiting time of a queue, the analysis procedure usually computes the state probability P_i first, which represents the probability of having i customers (packets or flits) in the queuing system [Kiasari et al. 2013a]. Then, the average number of customers N in the system with capacity K can be calculated as: [Kleinrock 1975; Donald and Harris 2008]: \[ N = \sum_{i=0}^{K} i \times P_i. \] Finally, according to the Little’s Law [Kleinrock 1975], the average waiting time in the queue W_s equals to [Kleinrock 1975; Donald and Harris 2008]: \[ W_s = \frac{N}{\lambda}, \] where \( \lambda \) is the average customer arrival rate at the channel.

In this survey, we summarize several representative NoC analytical models from the following aspects:

— The arrival process: Most NoC analytical models consider the router models under the assumption of Poisson arrival process. For example, in [Guz et al. 2007], an M/M/1 based channel model is designed to analyze impact of link capacity on flow latency. Similarly, in [Lai et al. 2009; Nikitin and Cortadella 2009; Ben-Itzhak et al. 2011], the IAT distribution of the header flits in each flow is assumed to be Poisson. Recently, much research efforts have been spent on generalizing the arrival process model. In [Wu et al. 2010], the traffic burstiness as well as the short-range dependencies (SRD) are characterized using a generalized exponential (GE) distribution [Wu et al. 2010]. In [Kiasari et al. 2013b], instead of using a GE arrival model, a 2-state MMPP model

---

4In this section, we use the notation "Single Channel" to represent the wormhole NoC routers with a single channel at each port, while the notation "Multiple VC" represents the NoC routers with multiple virtual channels.
Table III. Comparison of NoC analytical models [Qian et al. 2015; Qian et al. 2014]

<table>
<thead>
<tr>
<th>Queuing theory based analytical models</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Guz et al. 2007]</td>
</tr>
<tr>
<td>[Ogras et al. 2010; Lai et al. 2009]</td>
</tr>
<tr>
<td>[Kiasari et al. 2013b]</td>
</tr>
<tr>
<td>[Ben-Itzhak et al. 2011]</td>
</tr>
<tr>
<td>[Wu et al. 2010]</td>
</tr>
<tr>
<td>[Min and Ould-Khaoua 2004]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Queue model</th>
<th>M/M/1</th>
<th>M/G/1/K</th>
<th>G/G/1/∞</th>
<th>M/M/m/K</th>
<th>Ge/G/1/∞</th>
<th>MMPP/G/1/∞</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Queuing model summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrival process</td>
</tr>
<tr>
<td>Service process</td>
</tr>
<tr>
<td>Traffic pattern</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported router architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router architecture</td>
</tr>
<tr>
<td>Buffer size</td>
</tr>
<tr>
<td>Arbitration scheme</td>
</tr>
</tbody>
</table>

is used to capture the bursty arrivals at the injection sources. Specifically, in that work, a second moment term, namely the "squared coefficient of variance" (SCV) is computed to characterize the bursty traffic. In [Min and Ould-Khaoua 2004], a multiple-state MMPP traffic model is employed to better represent the self-similarity in the application traffic. Then the MMPP/G/1 queuing model is used to analyze the latency performance for a specific topology (hyper-cubes) in supercomputers.

- The service process: Under the assumption that packet sizes follow the negative exponential distribution, the service time of the packets can also be approximated as exponentially distributed [Kleinrock 1975]. Based on this assumption, several works use the memoryless service time models to predict the flow delays under different link capacities or buffer sizes [Guz et al. 2007; Hu and Marculescu 2004a]. On the other hand, there are also many multi-core applications which use constant packet size [Nikitin and Cortadella 2009] or a more general packet length distribution [Kiasari et al. 2013b]. In order to work for a constant packet length distribution, in [Nikitin and Cortadella 2009], a modified M/D/1 queuing model is proposed to address the non-memoryless service time distribution. In order to model a general independent service time distribution, in [Ogras et al. 2010] and [Kiasari et al. 2013b], the M/G/1 and G/G/1 queue-based models are used respectively. More specifically, the second moments of the service times are calculated in these models before applying the M/G/1 and G/G/1 queuing formula.

- The number of servers: Most work assumes a single server at each input port. In multiple-VC router architectures, since the upstream packets can request the usage of any one of the available VCs at the downstream node. Therefore, instead of using a single server queuing system, [Ben-Itzhak et al. 2011] proposed to use a multi-server model.

- The system capacity calculation: Many existing analytical models assume a single flit buffer [Ben-Itzhak et al. 2011] or the buffer size is negligible [Fischer and Fet-tweis 2013; Guz et al. 2007]. Under these assumptions, a packet actually occupies the whole routing path during its transfer [Ben-Itzhak et al. 2011]. Therefore, for each intermediate link channel in the routing path, the contending packets at the link channel are actually accumulated at the injection sources. Since the source queue is usually assumed to have infinite capacity [Ben-Itzhak et al. 2011], the queueing for-
mula with infinite capacity (e.g., M/M/1/∞ or M/G/1/∞ queue) are used to analyze the link channels [Ben-Itzhak et al. 2011]. On the other hand, there are also analytical models such as [Lai et al. 2009; Ogras et al. 2010] which are developed assuming a single channel buffer can hold \( N \) packets at one time. \( N \) is an integer larger than one. Then, the queueing system capacity is modelled as \( N \). The derivation of the queueing capacity becomes more complicated if the NoC router has a finite size buffer (e.g., several flits) and can only hold a portion of packet in its buffer. In [Hu and Kleinrock 1997; Kouvatsos et al. 2005], the authors develop techniques to calculate the queueing capacity under this situation. Specifically, in their approaches, for any link channel, its capacity is the summation of two parts. The first part represents the number of contention flows sending towards this link; the second part computes the average packets that are stored at the input port, which is further calculated from the packet length distribution.

In Table III, we summarize several representative queuing-theory-based models in NoC. As discussed above, each model relies on its specific assumptions and is most accurate when the assumptions capture the target application and router architecture characteristics.

Based on the models proposed in [Qian et al. 2015; Ogras et al. 2010; Hu and Kleinrock 1997; Kiasari et al. 2013b], we summarize a typical procedure of applying queuing theory to predict NoC average latency. For the sake of simplicity, as in [Nikitin and Cortadella 2009; Qian et al. 2014], we assume each packet has a fixed size of \( L \) flits. Interested readers can refer to [Hu and Kleinrock 1997; Kouvatsos et al. 2005; Arjomand and Sarbazi-Azad 2010] for additional modifications that are needed to extend the model from fixed-length assumption to a general packet size distribution.

3.1.2. Summary of latency calculation for an NoC flow. In typical NoC analytical models, the delay \( L_{s,d} \) (shown in Fig. 8-a and -b) is used to represent the overall packet transfer time from source \( s \) to destination \( d \). It is broken down into three parts [Ben-Itzhak et al. 2011; Qian et al. 2014]: 1) the waiting time at the injection processor \( s \) (i.e., \( v_s \)), 2) the packet transfer time (i.e., \( \eta_{s,d} \)) after being allocated the channel and 3) the path contention delay (i.e., \( h_{s,d} \)). More specifically, it is expressed in these works as:

\[
L_{s,d} = v_s + \eta_{s,d} + h_{s,d} \tag{7}
\]

To compute the path contention delay \( h_{s,d} \), it is required to aggregate the delays of each specific channel along the path of \( f \) (Fig 8-b illustrates the channels that should be considered for the flow \( f_{6,2} \)). Therefore, in [Ben-Itzhak et al. 2011], \( h_{s,d} = \sum_{i=1}^{d_f} h_{l_i} \), where \( l_i \) is the \( i \)-th link of flow \( f \) and \( d_f \) is the path length.

Similarly, the packet transfer time \( \eta_{s,d} \) can be calculated as [Ben-Itzhak et al. 2011; Ogras et al. 2010]: \( \eta_{s,d} = \sum_{i=1}^{d_f} \eta_{l_i} + (L - 1) \), where the transfer time of the header flit is added across every link, then the second term represents the absorption of the remaining \( (L - 1) \) flits at the sink PE.

In summary, to derive \( L_{s,d} \), the packet competition times \( h \) and the packet transfer times \( \eta \) at every link channel should be computed. The queuing models should address two issues: i) First, we need to analyze the dependency among the channels and decide the orders of links for applying queueing formula [Hu and Kleinrock 1997; Qian et al. 2014]. ii) Second, it is important to extract equivalent queueing systems for each link channel and clearly identify the arrival and service processes of the queueing system. Particularly, when modeling the arrival and service processes, we need to differentiate between the Single Channel and Multiple VC routers, which are shown in Fig. 8-c and -d, respectively. In section 3.1.3, we summarize the analytical procedures for Single
Fig. 8. An illustration of the queueing delay calculation [Qian et al. 2014]: a) an application represented as an architecture characterization graph (ARCG), b) the link channels which form the routing path of flow $f_{6,2}$, c) the illustration of path competition time $h$, the flit transfer delay $\eta$ and d) flow sharing effects in Multiple VC router architectures [Ben-Itzhak et al. 2011].

Fig. 9. An illustration of the channel service scenario for a packet in the east link channel of a Single Channel wormhole router architecture (c.f. [Hu and Kleinrock 1997; Kouvatsos et al. 2005; Qian et al. 2014]).

To resolve the first issue of channel dependencies, it is realized the enforced dependency is due to the flow control between NoC routers. For example, as shown in Fig. 8, both links $l_{6,7}$ and $l_{7,8}$ are part of the routing path of flow $f_{6,2}$. The waiting times of $l_{7,8}$ affect the packets transmission at channel $l_{6,7}$. There are several methods to address this link dependency issue. For example, in [Kiasari et al. 2013b], a channel indexing method is proposed based on the distances to the destinations. In [Lai et al. 2009], a channel sorting method is proposed based on the routing algorithm used. A more commonly used method is to build the link dependency graph (LDG) of the application. A proper link order ensures when analyzing a vertex (i.e., a link channel) in LDG, all its predecessors have been computed. For more details, the readers are suggested to refer to [Hu and Kleinrock 1997; Foroutan et al. 2013; Arjomand and Sarbazi-Azad 2010; Qian et al. 2014], where the procedures of building LDG and deriving the link orders from LDG are provided.

3.1.3. Single Channel Wormhole Router models. In the following, we summarize a typical procedure to calculate the competition delay $h$, the transfer delay $\eta$ and the source queue waiting delay $\nu_s$ for Single Channel wormhole router architectures.

1) Flit transmission delay queueing model. In [Ben-Itzhak et al. 2011; Qian et al. 2015], the flit transfer time $\eta$ over link $l$ corresponds to the time cost to send the flit from the input port buffer to the head of the buffer at the next router, assuming the flit has been allocated to use that link. To be more specific, in Fig. 9, the flit transfer time
over the link connecting router $R_1$ and $R_2$ is depicted as the time duration from Point A to D. In [Qian et al. 2014], it was further broken it into two parts for calculation. The first part consists of the time needed to traverse the current router. In a pipelined NoC router, this time can be approximated as the depth of router pipelines. After leaving the router, the second component in $\eta$ calculates the transfer delay of a flit to move to the next channel (i.e., moving from Point C to Point D in Fig. 9).

In order to calculate the transfer time from Point C to D, [Qian et al. 2015] proposed a queuing model whose capacity equals to $B + 1$. Specifically, $B$ is the router buffer size defined in Table II. The additional one in the capacity $B + 1$ considers the flit location at the upstream buffer where the request to the current link is generated and granted. In order to apply the queuing formula, the arrival and service process of this equivalent queueing system are identified. The mean customer (i.e., flit) arrival rate is computed by aggregating that of every flow in set $F_i$ [Ogras et al. 2010]:

$$\lambda_f^{\text{flit}} = L \times \sum_{f \in F_i} \lambda_f,$$

where $\lambda_f$ is the mean rate of flow $f$, $L$ is the packet size. The mean service time is approximated as the weighted average of different flows passing through the link $l$ [Hu and Kleinrock 1997]. Specifically, it is calculated as [Qian et al. 2015]:

$$s_f^{\text{flit}} = \frac{\sum_{f \in F_i} [\lambda_f \times \left( \frac{h_{f,l;l;1}}{L} + \frac{1-P_{b,f,l;l;1}}{1-P_{b,f,l;l;1}} \right) \lambda_f]}{\sum_{f \in F_i} \lambda_f} \quad (8)$$

where $d(f, l; 1)$ represents the downstream neighboring channel of link $l$ regarding to flow $f$. To understand Eqn. 8, we can consider the service process of a specific packet during the routing. This service process always starts with the downstream channel arbitration from the header flit in packet. In Eqn.8, it assumes the header flit takes $h_{d(f,l;1)}$ cycles to be allocated the downstream link. Then, for the body and tail flits, due to the Single Channel router architecture, they can use the link channel for transmission exclusively. Therefore, they are sent out smoothly without any further delay. The only exceptional case which interrupts the smooth transmission is due to the flow control. Specifically, if the downstream channel indicates its buffer is full, the body/tail flits have to be stalled. Let $P_{b,d(f,l;1)}$ represent the probability of having a full buffer at link $d(f, l, 1)$. Then, similar to derivation in [Lai et al. 2009], the mean time to route a flit under $P_{b,d(f,l;1)}$ can be calculated as $1 - P_{b,d(f,l;1)}$. Hence, the overall service time for a flit belonging to flow $f$ is the average of different flits in the same packet and can be computed as $\frac{h_{d(f,l;1)}}{L} + \frac{1-P_{b,d(f,l;1)}}{1-P_{b,d(f,l;1)}}$ in Eqn. 8.

After modelling the queuing system in the flit transfer process, the corresponding queuing formula such as M/M/1/K in [Ben-Itzhak et al. 2011] or M/G/1/K in [Hu and Kleinrock 1997; Lai et al. 2009] can be used to evaluate the system waiting time. The derived waiting time then provides an estimation of flit transmission time $\eta$.

2) Path competition delay queuing model: The arbitration or flow competition delay $h$ is defined in [Ben-Itzhak et al. 2011] as the time for a packet header to be successfully granted its next channel over other competitors. In general, we can also abstract a queuing system for this arbitration process [Ben-Itzhak et al. 2011; Hu and Kleinrock 1997]. Various analytical models have been proposed. Examples are the G/G/1 [Kiasari et al. 2013b], M/G/1/K [Lai et al. 2009; Arjomand and Sarchizi-Asad 2009], GE/G/1 [Wu et al. 2010] or MMPP/G/1 [Min and Ould-Khaoua 2004] queuing models. In these queuing models, the capacity $K$ of the queuing system is usually modelled by the total number of flows contending for the same output direction [Ben-Itzhak et al. 2011; Hu and Kleinrock 1997].

Similar to the procedure of deriving $\eta$, the mean customer arrival rate of the queue extracted for calculating $h$ is obtained by accumulating the traffic rate of all flows passing the channel. In addition to the first moment, other higher moments of the arrival
process can also be computed depending on the specific traffic model employed in the analysis. For example, if the G/G/1 [Kiasari et al. 2013b] or GE/G/1 [Qian et al. 2014] queuing model is used to derive the contention delay \( h \), the second moments (i.e., SCVs) of the packet arriving process over the channel \( l \) is also required. The derivations of the input traffic SCV can be calculated following the approximations in [Kiasari et al. 2013b; Qian et al. 2015].

To represent the service process of the queuing model, a widely used service time model is illustrated in Fig. 9 (c.f. [Hu and Kleinrock 1997; Qian et al. 2014]). In Fig. 9, assuming we need to calculate the contention delay of a packet in the east input channel of router \( R_1 \), the service of that packet begins right after its header flit in Point A being granted the next channel and finishes at the instance that the tail flit leaves the buffer. This time duration is denoted as the service time because after that, other flows requesting the usage of \( l \) can arbitrate again for the physical link. In this way, the waiting time of the extracted queuing system has the meaning of the contention delay to be allocated a channel. For a network under light/medium load, this service duration just equals to the packet size because each flit can pass through the channel directly; on the other extreme, under heavy traffic load (e.g., the downstream channels are under severe congestion), this time is calculated as the duration of the header flit to move to the location (i.e., Point B in Fig. 9) where the entire packet can be stored across that location and the current buffer slot (i.e., Point A) [Hu and Kleinrock 1997; Kouvatsos et al. 2005; Arjomand and Sarbazi-Azad 2010]. Let \( x_f \) represent this worst-case service time, then the mean queue service time \( s_f \) with respect to flow \( f \) is a value dictated by the length of packet \( L \) (minimum value) and \( x_f \) (maximum value) [Hu and Kleinrock 1997]. In practice, \( x_f \) is calculated first by accumulating the waiting times along the channels from Point A to B; then, \( s_f \) is approximated based on \( L \) and \( x_f \) as in [Hu and Kleinrock 1997; Qian et al. 2015]. After computing every \( s_f \), the mean service time for packets traversing \( l \) can be computed by averaging over \( s_f \) [Hu and Kleinrock 1997], i.e.:\[
\bar{s}_l = \sum_{f \in F_l} (\lambda_f \times s_f^f) / \sum_{f \in F_l} \lambda_f.
\]

Besides the mean value of the service time, for queuing models with non-memoryless service time distribution, such as the M/G/1, G/G/1 and MMPP/G/1 models, the second moment term, i.e., SCV of the service process is also required. One way to approximate SCV is given in [Kiasari et al. 2013b; Qian et al. 2015]:

\[
R_f^2 = \frac{(s_f^f)^2}{(x_f^f)^2} - 1 = \frac{\left( \sum_{f \in F_l} \lambda_f \times (s_f^f)^2 \right) / \sum_{f \in F_l} \lambda_f}{(s_l)^2} - 1 \quad (9)
\]

Based on above discussions, after mathematically building the models for the arrival and service process in the queuing system, the M/G/1/K [Ben-Itzhak et al. 2011], G/G/1 [Kiasari et al. 2013b] or GE/G/1 [Qian et al. 2015] queuing formula can then be applied to evaluate the waiting times of the queueing system, i.e., the flow contention delay \( h \) for the target link channel.

3) Waiting times at the source node: Typically, the traffic injection queues at the NoC network interfaces (NIs) is modelled as a system whose capacity is infinite; this is because the inner memory bandwidth of the PEs are usually much higher than that of the router channels, which therefore supports accumulating much more packets at the sources before injecting them into the network [Dally and Towles 2003]. To model the source waiting time, different queuing models can be applied. For example, [Ben-Itzhak et al. 2011] proposes to use an M/M/1/\( \infty \) queue and [Wu et al. 2010; Qian et al. 2014] uses the GE/G/1/\( \infty \) queuing model. Of note, for the source queues, the
traffic arrival and service processes are calculated in a way similar to that discussed in characterizing queuing systems for contention delay \( h \).

### 3.1.4. Extensions to router architectures with multiple VCs

For Multiple VC router architectures, the physical channel bandwidth is time-division multiplexed (TDM) among several flows at the same port [Ben-Itzhak et al. 2011]. To model the VC router architectures, two methods are widely used. The first way is to employ a multiple servers queue to replace the single-server model in Single Channel routers. Specifically, in [Ben-Itzhak et al. 2011], when calculating the path contention delay \( h \), an M/M/m/K queuing is used. The number of servers \( m \) in the model equals to the number of effective VCs at the downstream channel. Moreover, the authors also propose to consider the facts that when there are multiple flows being granted downstream VCs of the same port, the flits actually share the same physical bandwidth together. Therefore, the flits appeared on the same link can come from several flows which is different from the case in Single Channel wormhole routers. To address this, the derivation of the flit transfer time \( \tau \) should also be modified. Towards this end, the concept of "effective channel bandwidth" of a particular flow is introduced to represent the equivalent bandwidth seen by a VC during the switch transmission. With these two modifications, the path contention delay \( h \) and flit transfer time \( \tau \) should be able to incorporate the effects of VC sharing. They can then be used to predict the mean flow latency as in the Single Channel models.

The second way to model Multiple VC router architectures is to refine the obtained mean packet waiting time from Single Channel model by a factor \( V \) [Ould-Khaoua 1999; Dally 1992]. The parameter \( V \) represents the mean "degree of VC multiplexing" at a specific link channel [Ould-Khaoua 1999]. With the scaling factor \( V \), the mean flow delay from source tile \( s \) to destination tile \( d \) is rewritten as [Ould-Khaoua 1999]:

\[
L_{s,d} = (v_s + \eta_{s,d} + h_{s,d}) \times V.
\]

To calculate \( V \), a typical way is to employ the VC state transition diagram (STD) [Kiasari et al. 2008; Wu et al. 2010]. In STD, each vertex \( V_i \) represents a state that there are \( i \) VCs at the input port already being allocated to some packets. Specifically, the weight on the edge of STD pointing from state \( V_i \) to \( V_{i+1} \) denotes the probability that a new VC is allocated to other packets under the existence of \( i \) busy VCs. Similarly, the weight on the edge pointing from \( V_i \) to \( V_{i-1} \) reflects the process of finishing serving a packet and releasing one VC for further routing [Kiasari et al. 2008; Wu et al. 2010].

The transition rate between \( V_i \) and \( V_{i+1} \) depends on the current channel state (i.e., number of VCs being used) as well as the characterized arrival/service processes. In [Wu et al. 2010] and [Min and Ould-Khaoua 2004], the transition probability from \( V_i \) to \( V_{i+1} \) is derived according to an optimization algorithm that maximizes the overall entropy for the GE and MMPP type traffics; the transition rate from \( V_i \) to \( V_{i-1} \) is dictated by the packet service rate, which is \( 1/s \) (\( s \) is the average time for serving flits at the current channel). After calculating the transition rates on STD, the state probability, \( P_{V_i} \), \( (0 \leq i \leq V) \), which represents the probability that there are \( i \) VCs being allocated, can be computed by solving the extracted Markov chain as in [Min and Ould-Khaoua 2004; Wu et al. 2010; Ould-Khaoua 1999]. Finally, the degree of VC multiplexing \( V \) is given by [Dally 1992]:

\[
V = \frac{\sum_{i=0}^{V} v^i P_{V_i}}{\sum_{i=0}^{V} v^i P_{V_i}}
\]

### 3.2. Worst-case latency prediction for NoC

For many real-time systems such as the healthcare and embedded control related applications, it is more important to guarantee the packets are received and the subsequent actions are taken in time. For NoC systems, since most of the resources (e.g., links and buffer channels) are shared by flows, the contentions introduce a large delay...
variation, especially under heavy workload. Therefore, for these systems, one challenge is to predict the worst-case delay accurately to avoid over-design and ensure a configuration can meet the hard/soft deadline requirement. In [Kiasari et al. 2013a], the authors have surveyed three worst-case performance analysis approaches, that are network calculus based method, the dataflow analysis framework and the schedulability analysis. In this section, we first present another approach used in NoC community, i.e., real-time bound (RTB) analysis [Rahmati et al. 2013; Lee 2003; Rahmati et al. 2009].

For the sake of completeness, we also highlight the concepts and ideas of the other three approaches.

### 3.2.1. Real-time bound analysis for maximum delay prediction.

The rationale behind real-time bound (RTB) analysis is to predict the target flow’s latency under a scenario that provides least support for current packet transmission [Rahmati et al. 2009]. Specifically, this worst-case transmission of a flow $f$ occurs when all the router channels locating in $f$’s routing path are full and flow $f$ has the least priority to be allocated the routing resources when competing with other flows. Under this assumption, the packets of $f$ have to wait until all other flows have been served. Therefore, in [Rahmati et al. 2009; 2013], the upper bound latency $D_f$ of flow $f$ is re-written as:

$$D_f = ts_1 + ts_2 + \sum_{j=0}^{d_f} u_{i_{f}} l_{i_{f}} \in P_f$$

where $ts_1$ ($ts_2$) represents the delay of issuing (collecting) a packet with multiple flits into (out of) the NoC and $u_{i_{f}}$ is the maximum delay required for the packets in flow $f$ to leave the current channel $l_{i_{f}}$. Of note, to be consistent with the previous sections, also following the conventions in [Hu and Kleinrock 1997], here we use $l_{i_{f}}$ to represent the $i$ – $th$ hop channel in the routing path of flow $f$. $l_{i_{f}+1}$ represents the next link channel in flow $f$‘s routing path.

$u_{i_{f}}$ is made up of two parts [Rahmati et al. 2009; 2013]. First, it includes the residual (remaining) time to route the packets which are already being served at the channel buffer. These packets belong to either flow $f$ or its competitor $f^*$. Due to backpressure or flow-control, this residual time corresponds to the maximum waiting times of the downstream links:

$$\max(u_{i_{f}+1}^*, u_{i_{f}+1}^{f^*}), \forall f^* \in I_{i_{f}}(f),$$

where $I_{i_{f}}(f)$ denotes the set of flows that contends with $f$ at the current link $l_{i_{f}}$. $l_{i_{f}+1}$ and $l_{i_{f}+1}^{f^*}$ represent the downstream channel of flow $f$ and $f^*$, respectively.

The second part in $u_{i_{f}}$ considers the additional delay due to the loss of arbitration under the least priority assumption. More specifically, after the remaining packets leave the channel, all the flows compete to use that channel again. For the worst-case transfer situation, the packet of current flow $f$ has to wait for packets of all other flows finishing transmission once (i.e., $\sum_{f, f^*} u_{i_{f}+1}^{f^*}$) since it always loses the arbitration.

Finally, the overall delay of $u_{i_{f}}$ is given by adding these two parts together [Rahmati et al. 2009]:

$$u_{i_{f}} = \max(u_{i_{f}+1}^*, u_{i_{f}+1}^{f^*}) + \sum_{f^* \in I_{i_{f}}(f)} u_{i_{f}+1}^{f^*} \forall f^* \in I_{i_{f}}(f)$$

At the destination nodes, since they consume the packets immediately; both $u_{i_{f}+1}^{f^*}$ and $u_{i_{f}+1}^{f^*}$ in Eqn. 11 equal to the packet length. Therefore, in practice, $u_{i_{f}}$ is calculated from the flow destinations backwards to the sources [Rahmati et al. 2009]. After obtaining $u_{i_{f}}$, the worst-case delay bound $D_f$ can be calculated accordingly.
3.2.2. Network calculus based worst-case delay prediction. Following the discussions in [Boudec and Thiran 2004; Kiasari et al. 2013a; Chang 2000], we can summarize the rationale of network calculus as follows: for a queuing system, let \( I(t) \) and \( O(t) \) represent its input and output characteristic function, respectively. More precisely, \( I(t) \) denotes the total number of packets received at a specific router port until time \( t \). \( O(t) \) represents the number of packets sending out of that port until the same time instance. Under these notations, \( b(t) = I(t) - O(t) \) actually represents the number of customers that have to wait in the buffer at time \( t \); while \( d(t) = \inf_{\tau \geq 0} \{ I(t) \leq O(t + \tau) \} \) describes the time duration that a customer has to stay in the system before it is routed to the next node under the assumption this customer arrives at time \( t \) [Qian et al. 2009a; Kiasari et al. 2013a].

For the worst-case delay derivation, the network calculus approaches use two curves to bound the input and output characteristic functions, namely the arrival curve \( \alpha(t) \) and the service curve \( \beta(t) \) [Kiasari et al. 2013a]. \( \alpha(t) \) and \( \beta(t) \) bound the arrival function \( I(t) \) and output function \( O(t) \) as follows [Chang 2000; Qian et al. 2009a]:

\[
I(t) - I(s) \leq \alpha \times (t-s), \ \forall s \leq t \tag{12}
\]

\[
O(t) \geq I(s) + \beta \times (t-s), \ \forall s \leq t \tag{13}
\]

Giving the characterized \( \alpha(t) \) and \( \beta(t) \) curves, the maximum delay of forwarding a customer is obtained by computing the maximum horizontal distance between \( \beta(t) \) and \( \alpha(t) \) curves over all the time instances \( t \) [Kiasari et al. 2013a; Chang 2000].

To calculate the maximum routing delay of a specific flow \( f \) in NoC, all the routers through which flow \( f \) passes are considered first. Then, an equivalent service curve \( \beta_f \) of this flow is derived by convolving each router's service process [Qian et al. 2009a]. To resolve different contention scenarios that a flow may encounter, in [Qian et al. 2010b], the authors propose a systematic way to build a contention tree. The flows competing with the current flow \( f \) at each router are recognized. Then, the contention scenarios are classified into three typical cases. The equivalent service curve is derived respectively for these three cases. The model is further optimized in [Qian et al. 2009b] by considering the feedback between the neighboring routers using credits for flow control.

Recently, the network-calculus based models have evolved in the following directions:

— Analyzing self-similar traffic: In [Qian et al. 2009c], the network calculus based techniques are applied to estimate the delay under self-similar traffic. Specifically, instead of using a deterministic arrival curve, the self-similar arrival curve is derived by including a few additional parameters (namely the excess probability in that work) to capture the LRD in the traffic. By using the new arrival curve model, the authors demonstrate a more practical delay bound can be obtained for fractal applications.

— Modeling for more complicated router architectures: in [Qian et al. 2010a], the network calculus approach is applied for the VC router architectures by further considering the service process of VC allocation. In [Zhao and Lu 2013], the NoC elements are modelled using a formal description approach called eXecutable Micro-Architectural Specification (xMAS) [Chatterjee et al. 2012]. Then the xMAS NoC models are analyzed using network calculus techniques to compute the delay bound of each flow. Compared to the previous models, the xMAS model takes the advantage of the xMAS framework to integrate more control details in the NoC routers.

— Applying real-time calculus techniques: Real time calculus [Thiele et al. 2000] is proposed to extend the network calculus idea on the real time systems. Compared to the conventional network calculus approaches, the upper and lower bound curves which are the functions of the time interval, are used to characterize the arrival
and service processes, respectively [Kiasari et al. 2013a]. For instance, the upper bound arrival curve with a parameter \( \delta \) indicates the maximum number of arriving customers during the time interval \( \delta \). Based on the upper and lower bound curves, [Chakraborty et al. 2003] develops a framework to analyze the maximum delay for the real-time embedded systems.

— Predicting statistical delay distribution: Recently, the stochastic network calculus [Jiang and Liu 2008] technique has been applied in NoC to predict the statistical delay bound for NoC systems [Lu et al. 2014]. Specifically, the original deterministic arrival and service curves have been extended with an additional function to describe the probability of bounding the actual arrivals and services [Kiasari et al. 2013a]. In this way, the delay distribution of a target flow can be predicted using the statistical arrival and service curves; the result is more useful for systems with soft-deadline requirements.

3.2.3. Schedulability and data flow analysis techniques. Besides above two approaches, the schedulability analysis and data flow analysis techniques have also been applied in NoC-based systems (e.g., [Shi and Burns 2008; Bekooij et al. 2004]). In general, these two approaches are more often used in evaluating whether a scheduling/priority assignment policy can satisfy the deadline requirement of real-time systems. For the schedulability analysis in [Shi and Burns 2008], the authors identified two types of interferences of a target flow \( f \). Specifically, the direct interfering flows share the links with \( f \) and cause resource contentions. The indirect interferences only implicitly affect \( f \) via influencing the issuing time of packets in \( f \)'s direct interferences. Based on the priority assignment and the preemption assumption, a worst case delay can be derived accordingly. Such analysis framework is used to explore an optimal (i.e., lowest hardware cost) task mapping and priority assignment for NoC-systems with deadline constraints [Shi and Burns 2010]. For the data flow analysis, the data flow graph of the system is built first. Then, it can be used to evaluate whether a given buffer sizing or scheduling approach meet all the flow deadline requirements [Bekooij et al. 2004; Hansson et al. 2008].

4. SIMULATION BASED EVALUATION METHODS

Besides mathematical models, NoC simulations are also important in performance evaluations due to their higher accuracy. In this section, we review the development of simulation-based evaluation methods in NoC.

4.1. NoC simulators design

When using an NoC simulator to evaluate a design, the user needs to choose one with a proper abstraction level. Generally, the most accurate simulator implements every component (e.g., crossbar switch, switch allocators etc.) in the router exactly following the guidelines to be laid out. Netmaker [Netmaker 2009] is such a simulation platform which is implemented in SystemVerilog and can be synthesized in ASIC or FPGA. It supports a variety of router and network configurations (e.g., routers and links with variable numbers of pipeline stages). It also supports some advanced router structures such as the speculative router [Peh and Dally 2001]. Another example is CONNECT [Connect 2011], which is a flexible Bluespec SystemVerilog based NoC emulator supporting different configurations in architectures (e.g., input queued, virtual output queued or virtual channel based routers). Moreover, the CONNECT users can also configure the network into different topologies such as Mesh, Torus, Fat Tree and Butterfly. In general, the hardware description language (HDL) based platforms are most suitable for the final stage system prototyping. When such simulators are used in the design space exploration, it takes a significant time to re-synthesize the designs.
and re-run the simulations especially when the data- or control-path components still need to be largely explored.

To speedup the evaluation, the other type of NoC simulators abstracts the router components in a higher level language (e.g., SystemC or C++) and provides cycle- and flit-accurate evaluations of the target design. Compared to hardware description languages, more flexible data types and structures can be used in these models. Therefore, many router components such as buffers and allocators can be implemented more efficiently. The simulations are then performed either cycle-by-cycle (i.e., cycle-driven) or use an event queue to maintain the next operation (i.e., event-driven) [Dally and Towles 2003]. Another feature of these simulators is the parametrized design, which allows the users to explore over a large design space such as the buffer sizes, link bandwidth, routing schemes without re-compiling the simulators.

In Table IV and V, several widely used NoC simulation tools in the research community are summarized and compared. They are Netmaker [Netmaker 2009], CONNECT [Papamichael and Hoe 2012], Booksim [Jiang et al. 2013], Noxim [Noxim 2011], Wormsim [Wormsim 2008], HNoC [Ben-Itzhak et al. 2012; HNoC 2013], NNSE [Lu et al. 2005], gpNoCsim [Hossain et al. 2007], TOPAZ [Abad et al. 2012], Garnet [Agarwal et al. 2009], gMemNoCsim [gMemNoCsim 2011; Lodde and Flich 2012], Xmulator [Nayebi et al. 2007], DARSIM [Lis et al. 2010], NoCtweak [Tran and Baas 2012], Xpipes [Bertozzi and Benini 2004; Stergiou et al. 2005], OCIN_TSIM [Prabhu 2010] and HORNET [Lis et al. 2011] 5. Specifically, in Table IV, we summarized the topologies, the router architectures, the backpressure signal characteristics and the traffic

5 Actually, there are more open-source simulators that are available and have been used in NoC evaluations, examples are the SICOSYS [Puente et al. 2002], OCCN [OCCN 2003], Nigram [NIRGAM 2007] and Atlas [Atlas 2011]. They share many similar features as those in Table IV and V. Interested readers may refer to [Ababei et al. 2012], which summarized the information of simulators used by different groups.


<table>
<thead>
<tr>
<th>Table IV. Summary of current NoC simulators</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Topology</strong></td>
</tr>
<tr>
<td>Netmaker</td>
</tr>
<tr>
<td>CONNECT</td>
</tr>
<tr>
<td>Booksim</td>
</tr>
<tr>
<td>Noxim</td>
</tr>
<tr>
<td>WormSim</td>
</tr>
<tr>
<td>HNoC</td>
</tr>
<tr>
<td>NNSE</td>
</tr>
<tr>
<td>gpNoCsim</td>
</tr>
<tr>
<td>TOPAZ</td>
</tr>
<tr>
<td>Garnet</td>
</tr>
<tr>
<td>gMemNoCsim</td>
</tr>
<tr>
<td>Xmulator</td>
</tr>
<tr>
<td>DARSIM</td>
</tr>
<tr>
<td>NoCtweak</td>
</tr>
<tr>
<td>Xpipes</td>
</tr>
<tr>
<td>OCIN_TSIM</td>
</tr>
<tr>
<td>HORNET</td>
</tr>
</tbody>
</table>

---

\( ^a \) SPEC: the Speculative router architecture [Peh and Dally 2001]
\( ^b \) IQ: Input queued router; VOQ: Virtual output queued router
\( ^c \) WH: the wormhole router without using virtual channels
\( ^d \) ABP: Alternating Bit Protocol for flow control [Noxim 2011]
\( ^e \) HVC: Heterogeneous Virtual channel architecture. The VCs of each port may be different.
\( ^f \) VCT: Virtual Cut Through router architecture
\( ^g \) BFL: Bufferless router architecture
\( ^h \) Xpipes Lite [Stergiou et al. 2005] supports the stall and go protocol [Flich and Bertozzi 2010].
\( ^i \) BDL: NoC architecture with Bi-directional Links
### Table V. Summary of current NoC simulators (Cont.)

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Modeling Language</th>
<th>Parallel sim</th>
<th>Embedded in FSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netmaker</td>
<td>Packet/VC^a/Allocator^b etc.</td>
<td>SystemVerilog</td>
<td>-</td>
</tr>
<tr>
<td>CONNECT</td>
<td>VC/Allocator etc.</td>
<td>Bluespec SystemVerilog</td>
<td>-</td>
</tr>
<tr>
<td>Booksim</td>
<td>Pipeline^c/VC/Allocator/Packet etc.</td>
<td>C++</td>
<td>-</td>
</tr>
<tr>
<td>WormSim</td>
<td>Buffer/Packet etc.</td>
<td>SystemC</td>
<td>-</td>
</tr>
<tr>
<td>HNoC</td>
<td>VC/Link etc.</td>
<td>OMNET++</td>
<td>√</td>
</tr>
<tr>
<td>NNSE</td>
<td>VC/Packet etc.</td>
<td>SystemC</td>
<td>-</td>
</tr>
<tr>
<td>gpNoCsim</td>
<td>Packet/VC etc.</td>
<td>Java</td>
<td>-</td>
</tr>
<tr>
<td>TOPAZ</td>
<td>Pipeline/Packet/VC etc.</td>
<td>C++</td>
<td>√</td>
</tr>
<tr>
<td>Garnet</td>
<td>Pipeline/VC/Bandwidth etc.</td>
<td>C++</td>
<td>-</td>
</tr>
<tr>
<td>gMemNoCsim</td>
<td>VC/Packet etc.</td>
<td>C/C++</td>
<td>√</td>
</tr>
<tr>
<td>Xsimulator</td>
<td>VC/Message etc.</td>
<td>C#</td>
<td>-</td>
</tr>
<tr>
<td>DARSim</td>
<td>VC/Packet/Bandwidth etc.</td>
<td>C++</td>
<td>√</td>
</tr>
<tr>
<td>NoCTweak</td>
<td>VC/Pipeline/Allocator etc.</td>
<td>SystemC</td>
<td>-</td>
</tr>
<tr>
<td>Xpipes</td>
<td>VC/Switches^e etc.</td>
<td>SystemC/Verilog</td>
<td>-</td>
</tr>
<tr>
<td>OCIN^f_TSIM</td>
<td>VC/Pipeline/Allocator etc.</td>
<td>C++</td>
<td>-</td>
</tr>
<tr>
<td>HORNET</td>
<td>VC/Packet/Bandwidth etc.</td>
<td>C/C++</td>
<td>√</td>
</tr>
</tbody>
</table>

\^a FSS: Full-system simulator. The entries with √ represent the simulator has been embedded in some full-system simulator, such as Simics [Simics 2012], Gem5 [Gem5 2009], and Graphite [Graphite 2010] for execution-driven simulation.
\^b The virtual channel number, buffer depth can be specified by the user.
\^c The allocator and arbiter type can be configured.
\^d The pipeline depth can be specified.
\^e The switch (crossbar) size can be configured.

Supported by these simulators. As shown in Table IV, most simulators support the exploration of regular or semi-regular topologies (e.g., mesh, torus and hyper-cubes). These topologies are more commonly used in Chip Multiprocessor (CMP) architectures, where the processing elements are homogeneous. On the other hand, there are other simulation frameworks such as Xpipes and HNoC, which support arbitrary topologies and heterogeneous configuration of different routers. Therefore, they are suitable to explore the design of Multiprocessor System-on-Chip (MPSoC) architectures. In Table V, we summarize the features of different simulation engines, including their configurable parameters, the modeling language, whether they provide the native support of parallel simulation and can be directly embedded in a full-system simulator. For the configurable parameters, most simulators allow the user to specify the VC (e.g., number of VCs per port and VC buffer depth) and the packet (e.g., packet size, fixed- or variable length distributions) configurations. Some simulators also provide the options for users to explore the router architectures (e.g., allocator type and pipeline depth). For the power evaluation, two methods are widely used. The first is to use bit-level energy model which estimates the energy consumption of transferring a single bit over the switch and the link [Hu and Marculescu 2005]. Noxim and Wormsim simulators are two examples. The second method is to include the Orion [Kahng et al. 2012] library, which is a detailed NoC power model, in the simulators. For example, Booksim and Garnet simulators have integrated Orion. Wormsim also provides an option to evaluate power using Orion model.

4.2. Addressing scalability challenge in NoC simulations

Previously, for flit-level NoC simulators, there are built on either cycle-based or event-driven models [Dally and Towles 2003]. Cycle-based simulators check and update the network status (e.g., the buffer occupancy, the allocation request/response) every cycle. On the other hand, event-driven simulators update the network status only at the specific instances when an event/transaction was previously scheduled in the event queue. Both these two simulators accurately predict the design performance in the system...
level. However, with the advancement in IC integration, there may be hundreds or thousands of cores on future NoC systems [Borkar 2009]. An emerging challenge for both cycle-based and event-driven evaluations is to provide fast and efficient simulation of a large scale NoC system. In the following, we summarize several representative approaches:

— High level abstraction for simulation: INSEE [Ridruejo Perez and Miguel-Alonso 2005] is such an simulation framework which contains a functional simulator named FSIN. Different from cycle-accurate simulators, FSIN omits some modeling details such as cycle-accurate behaviors of the routers. Therefore, the simulator is simple and fast for early phase design space exploration. The users can use FSIN to simulate a much larger network compared to the SICOSYS [Puente et al. 2002] simulator. However, the accuracy is compromised in such functional simulators.

— Reduce simulation points by statistical sampling techniques: To compensate the accuracy loss in functional simulation, one way is to combine detailed and functional simulations. In general, the statistical sampling techniques help to choose only a portion of the application that needs to be fully simulated [Dai and Jerger 2014]. For the other parts, either only functional simulation is conducted or they are simply skipped. In [Dai and Jerger 2014], the idea of previous sampling based full-system simulation (e.g., [Carlson et al. 2013],[Ardestani and Renau 2013] and [Wenisch et al. 2006]) is extended to NoC-based systems. Two schemes are proposed. The first scheme uses sampling theory to explore the size of samples based on the metric (e.g., latency) variation and confident interval requirements. The second scheme identifies different phases in the application by using the clustering/classification method. Then, the sampling is done by choosing samples from each class. It has been demonstrated that both approaches can achieve an order of magnitude speedup with a small error (less than 10%).

— Exploring capabilities for parallel simulation: To enable fast and large-scale simulation, instead of executing the simulator program sequentially on a host PC/workstation, parallel computing resources (e.g., multithreads or multicore resources on a CPU or GPU card) are exploited to accelerate the overall evaluation process [Eggenberger and Radetzki 2013]. In [Lis et al. 2011], a highly flexible and multi-threaded-based NoC simulator named HORNET is developed. Specifically, parallel evaluation is achieved by dividing the whole NoC system into several different parts; each part is mapped and scheduled onto an individual thread. During the simulation, different threads are synchronized periodically to exchange the network status of different regions. The synchronizing frequency determines the tradeoff between the simulation speedup and the evaluation accuracy. On one hand, frequent synchronizations ensure each region to keep pace with the state changes that happen in the neighboring region. On the other hand, frequent synchronizations create barriers and significantly affect the overall speedup [Eggenberger and Radetzki 2013]. In [Zolghadr et al. 2011], an NoC simulator is developed on Nvidia GPU platform based on CUDA library. In this simulator, each GPU thread is used for the evaluation of a router or a link. Every cycle, the shared or global memory of GPU is used to synchronize/update the transactions. Therefore, it requires a significant effort on synchronization among threads; the overall speedup performance under very large network size is limited [Eggenberger and Radetzki 2013]. In [Pinto et al. 2011], the General Purpose GPU (GPGPU) based simulator for large scale NoC system is designed. Each GPU thread is used to evaluate a tile (the core and router). The GPU global memory is used to store the packets to speedup the data access across different threads. To simulate a large-scale network, the instruction instead of cycle level accuracy is provided. Recently, in [Eggenberger and Radetzki 2013], the authors pro-
pose a simulation method with high scalability. Specifically, the simulation tasks are ordered first based on their dependencies. The maximum number of tasks that can be simulated simultaneously without causing interferences with each other is identified. An efficient synchronization scheme is then used together with a load balancing scheme to dynamically distribute the simulation workload among different threads within the host PC. Based on above optimizations, when executing the simulations on a multi-core host computer, a significant speedup can be achieved even for an NoC with more than one thousand processors.

4.3. NoC benchmarking efforts

Standardization efforts have been taken to provide methodologies for NoC benchmarking. The Open Core Protocol International Partnership Association (OCP-IP) formed a working group on NoC benchmarking [Mackintosh 2008] to propose standard benchmarks and methods for evaluating and comparing NoC designs with different topology, traffic and router implementation characteristics [Grecu et al. 2007]. With a similar target, the NoCBench [NoCbench 2011] project has also released its models which contains benchmark traffic and router models for performance evaluation.

5. COMBINING ADVANTAGES OF ANALYTICAL MODELS AND SIMULATIONS

The NoC analytical models discussed in Section 3 provides a fast way for evaluating NoC performance in the early design stage. On the other hand, the NoC simulators closely model the real system behaviors at run time but require significantly longer time. Therefore, one interesting research direction in NoC performance evaluation is to combine the advantages in these two approaches and explore a new way for estimating NoC performance. In this section, we review several interesting progresses towards this end. We first discuss using FPGAs to accelerate the simulations and then introduce the machine learning based techniques to improve the analytical model accuracy from simulation training samples.

5.1. Hardware based NoC simulator

Instead of implementing the software-based NoC simulators, one recent direction in NoC emulator design is to use real hardware platform for performance evaluation. Of note, this is different from previous work [Wolkotte et al. 2007; Genko et al. 2005] which implements the NoC exactly as an ASIC chip or a FPGA embedded system for prototyping. The concept of hardware based NoC simulator still tries to explore a higher level abstraction of the router model (for example, using the FPGA to implement the mathematical models as in [Papamichael et al. 2011] or the behavioral models as in [Wang et al. 2011]). The motivation of implementing such a high level NoC model on hardware is from two aspects [Wang et al. 2011]: first, the abstraction in a higher level takes the advantage of larger flexibility. Therefore, they are suitable to explore and compare different design choices before the data- and control- path being fixed. Second, compared to executing the simulation as a software program, the hardware based simulator emulates different PEs and routers in parallel every cycle. Therefore, the simulator is more scalable for evaluating a large network size and reduces the overall evaluation time.

For example, in [Wang et al. 2011], a flexible and efficient FPGA-based simulation architecture was developed. The authors claimed instead of implementing the system exactly on hardware, it is more flexible to map the major data- and control- path components as generic library components in FPGA. For different NoC architectures to be evaluated, the users just need to make a collection of corresponding basic elements from their library (e.g., the specific arbiters and buffers). In this way, the FPGA simulation engine can be viewed as a virtualization of the architecture to be explored.
Moreover, by using hardware rather than a software program to conduct simulations, an over 100× speedup compared to the Booksim simulator can be achieved without degradation in the final simulation accuracy [Wang et al. 2011].

In [Papamichael et al. 2011], a fast and simple FPGA based simulator (FIST) for evaluating NoC latency is developed. The objective of FIST simulator is to implement an analytical router model in hardware so as to replace the detailed full-system evaluations. The main idea of FIST design is to model each router as a lookup table in FPGA which corresponds to a load-delay curve identified from the queuing-theory based models. At run time, during every time window, by checking the current load at the intermediate router, a delay value can be obtained which determines the waiting time of a packet in the current router buffer before being routed. By following the packet routing path and adding together the load-dependent delays at each intermediate router, the user can obtain an end-to-end delay estimation for a specific packet/flow. Compared to the conventional queuing models, the FIST simulator adjusts the delays based on the router’s load dynamically. Therefore, it better reflects the temporal behaviors (e.g., latency variations over different time period) of a traffic flow. Compared to the cycle-based full-system simulation, a 43× speedup and less than 6% error are reported for the FIST simulator [Papamichael et al. 2011].

5.2. Learning based NoC performance evaluation

As discussed in Section 3, most NoC analytical models are based on queuing theory, which use the M/M/1, M/D/1, M/G/1/K or G/G/1 formula to estimate the channel waiting times. These models can achieve high prediction precision if some assumptions are met, such as the injection process of the packet headers is Poisson [Ogras et al. 2010].
However, for the cases where the application or the NoC configuration does not satisfy these assumptions, the accuracy of the analytical model degrades. To address such problem, in [Qian et al. 2013; 2015], SVR-NoC, a support vector regression based latency model is proposed for NoC latency prediction. The rationale of the SVR-NoC is to refine the analytical models by applying learning techniques on the collected simulation samples. To be more specific, the workflow of the SVR-NoC can be summarized as follows (shown in Fig. 10) [Qian et al. 2013; 2015]: First, based on the NoC architecture on which the application is mapped and executed, several synthetic traffic patterns are fed into a corresponding simulator to collect training data samples. The non-parametric support vector regression [Vapnik 1998] is then applied on the collected training data to estimate the relationship between the waiting times and their features. In the learning process, the users need to collect two sets of features, i.e., $X_{CQ}$ and $X_{SQ}$ which correspond to the features of the waiting times at the buffer channels (i.e., the channel waiting delay $CQ$) and source queues (i.e., the source waiting delay $SQ$). Of note, the features in the set $X_{CQ}$ and $X_{SQ}$ include the factors that directly affect the channel and source waiting times. For example, the arrival rate elements in $X_{CQ}$ and $X_{SQ}$ reflect the traffic load injected onto the channel, while the forwarding probability elements in $X_{CQ}$ aim to recognize the potential contentions among the flows within the same router. After applying support vector regression (SVR) for learning, two model functions, namely the channel queuing and source queuing model $(f_{CQ}$ and $f_{SQ})$ are obtained from the training data. During the training process, in order to avoid data-overfitting, cross-validation needs to be used to determine the best hyper-parameter combinations [Vapnik 1998]. Finally, to apply the learning model to evaluate the NoC latency of a new application, the input traffic is first characterized as an architecture characterization graph (ARCG) discussed in Section 2.1. Then, for each source channel and intermediate link channels, the corresponding feature vectors $X_{CQ}$ and $X_{SQ}$ are calculated based on the extracted ARCG. By applying the regression functions $(f_{CQ}$ and $f_{SQ})$ on the new feature sets, the waiting times at each link channels as well as the overall flow latency can be predicted accordingly.

6. OPEN PROBLEMS AND RESEARCH PERSPECTIVES

Based on discussions in the previous sections, several open problems that need to be addressed are identified and some potential research directions are presented in the following discussions:

— Modeling non-stationary and time-dependent NoC traffic: The major problem of memoryless (i.e., Poisson or Markovian) and short-range dependent traffic models (i.e., GE, 2-state MMPP) is that they cannot capture the time-dependent arrival processes. For NoCs under light or medium load, the temporal correlations among the packets are not obvious. Therefore, these models may still provide good approximations. However, under heavy workload (i.e., high packet injection rates [Bogdan and Marculescu 2011]), the contentions for the shared resources introduce a long tail of the arrival and service time distribution, which indicates the earlier packets in the network may affect their subsequent ones. The accuracy of delay prediction applying such models is greatly reduced. To overcome this limitation, there are two possible directions. The first direction is to generalize over the current Poisson or 2-state MMPP traffic models. Specifically, in the subject of Internet traffic modeling and operational research, the more general processes such as Batch Markovian Arrival Processes (BMAP) have been used to model the long-range dependency relations [Klemm et al. 2002]. The self-similarity and traffic burstiness can be more efficiently characterized. The second direction is to use the multi-fractal traffic model. Currently, the model is complicated and requires a set of master equations to be solved. An efficient
— Developing corresponding performance models for multi-fractal traffic: The multi-fractal analysis improves the accuracy over previous models, especially when the system works near the saturation point. Moreover, when fixing some of the exponent parameters, it has been observed that multi-fractal model reduces to the previous memoryless or short-range dependent models. In conclusion, the multi-fractal analysis framework provides a more general solution in both the traffic analysis and performance prediction. However, the complexity of solving master equations is much higher than other models. Also, current research efforts only focused on the workload characterization. The performance evaluation model (e.g., models for average-case, worst-case or statistical metrics) based on multi-fractal input is still rarely explored. One research direction is to consider solving the multi-fractal formalism presented in [Bogdan 2015] and derive the probability of large (rare) events in the formalism. Then, these probabilities could be used to compute the real-time performance metrics such as the delay bounds.

— Relaxing assumptions in current queueing models: The common limitation of analytical queueing models is they rely on different assumptions that are made before the derivation. The accuracy and usage is constrained by how close the real cases match with the assumptions. For the average-case prediction, as summarized in Table III, each queueing model is built based on its specific traffic input and router architecture (e.g., buffer size, arbitration policy). The accuracy degrades if the model is applied to a very different situation. Recently, some generalization efforts have been made, such as assuming general independent arrival and service processes, removing the assumptions on the buffer depths and packet sizes. However, these models still do not capture the time-dependency among the inter-arrival and service processes. Therefore, one future direction is to extend the queueing model with long-range dependent arrival and service processes while relaxing the assumptions on the topology, buffer size and arbitration scheme at the same time.

— Transient analysis for Network-on-Chips: Most analytical queueing models only focus on the aggregated average performance (e.g., mean delay). However, the authors in [Ohmann et al. 2014] pointed out for non-stationary applications, since the system is unstable, transient behavioral analysis is more useful. In [Ohmann et al. 2014], they also propose a transient queueing model for a single router element. They discussed, for this emerging direction, additional efforts are required to provide a working model for a router network. Moreover, the methods of applying transient analysis for better flow control or buffer sizing are also desired [Bogdan 2015].

— Efficient resources management strategies: The NoC performance models have been widely used in buffer sizing [Du et al. 2014] and flow regulation [Jafari et al. 2010]. However, if the workload is time-dependent, the configuration decision made may be optimistic and not satisfy the QoS requirement [Varatkar and Marculescu 2004]. This limits the usage of such models for some real-time systems. On the other hand, if a multi-fractal traffic model is used, it requires to develop an efficient hardware implementation which can support collecting the information and re-acting to the network congestions in time.

— Developing efficient approaches for modeling NoC resources contentions in real-time systems: In NoC, the resources (e.g., link channels) are shared among different flows. Therefore, it has been realized in research that the contentions are the major causes of delay variations. How to efficiently model the effects of resources sharing is one key problem. Especially for real time systems with QoS requirement, over-estimating contention delays introduces additional resources overhead while under-estimating the delays causes a potential system failure. In the worst-case analytical models,
the real bound analysis makes a conservative assumption that the target flow always loses arbitration. The bound tightness is compromised. The network calculus bounds depend on whether the arrival and service curves can capture the real-time or statistical behaviors. How to model a network with more complicated contention scenarios still needs to be explored. For the schedulability and data flow analysis, they make assumptions on the priority assignment of the traffic flows. Different VCs in the same port may be allocated to different priority levels. Therefore, they are mostly suitable for priority-based architectures. For best-effort NoC or real time NoC with soft-deadline requirements, these two approaches cannot be directly applied.

Addressing scalability challenge in NoC simulators design: For a large-scale system with more than hundreds of cores, the full-system simulation is unaffordable. Even with current parallel simulation techniques, how to efficiently partition the simulations among multiple threads and address the synchronization barrier is still difficult. Statistical sampling techniques help to reduce the simulation points. However, the determination of sampling strategy is application-specific which relies on the pre-analysis of the application characteristics. Another direction is to use hardware based modeling or learning based techniques to accelerate the evaluation. For the learning based modeling approach, the current model only works for the average latency prediction. How to reduce the training data size and extend the learning method for more performance metrics (e.g., worst-case or statistical delay) remain open to the whole community.

7. CONCLUSIONS
In this survey, we have reviewed several techniques for NoC performance evaluation. These techniques range from building the traffic models to developing the analytical and simulation-based latency models. We first summarized the typical workloads that are employed in NoC-based system evaluation. Then, we have reviewed the approaches in traffic analysis for capturing both the short- and long-range dependence behaviors. In addition, a multi-fractal based approach to model the non-stationary traffic is introduced. We concluded the traffic analysis techniques by summarizing the features of each traffic model. For the NoC latency performance evaluation, we have presented the techniques to predict the average-case and worst-case delay. The simulation-based evaluation models are also reviewed. We have also reviewed the state-of-the-art progresses that combines the advantages in analytical models and simulations to provide rapid and scalable performance evaluations. The performance evaluation for NoC-based multicore system still faces many interesting challenges. We have discussed several open problems and potential research directions towards this purpose.

ACKNOWLEDGEMENT
The authors would like to thank the reviewers for their detailed and valuable comments and suggestions.

REFERENCES


Performance Evaluation of NoC-based Multicore Systems: From Traffic Analysis to NoC Latency Modelling


