Optimizing Job Reliability Through Contention-Free, Distributed Checkpoint Scheduling

Yu Xiang, Hang Liu, *Member, IEEE*, Tian Lan, *Member, IEEE*, Howie Huang, *Senior Member, IEEE*, and Suresh Subramaniam, *Fellow, IEEE*

***Abstract*—A datacenter that consists of hundreds or thousands of servers can provide virtualized environments to a large number of cloud applications and jobs that value the requirement of reliability very differently. Checkpointing a virtual machine (VM) is a proven technique to improve reliability. However, existing checkpoint scheduling techniques for enhancing reliability of distributed systems fails to achieve satisfactory results, either because they tend to offer the same, fixed reliability to all jobs, or because their solutions are tied up to specific applications and rely on centralized checkpoint control mechanisms. In this work, we first show that reliability can be significantly improved through contention-free scheduling of checkpoints. Then, inspired by the Carrier Sense Multiple Access (CSMA) protocol in wireless congestion control, we propose a novel framework for distributed and contention-free scheduling of VM checkpointing to provide reliability as a transparent, elastic service. We quantify reliability in closed form by studying system stationary behaviours, and maximize job reliability through utility optimization. Our design is validated via a proof-of-concept prototype that leverages readily available implementations in Xen hypervisors. The proposed checkpoint scheduling is shown to significantly reduce checkpointing interference and improve reliability by as much as one order of magnitude over contention-oblivious checkpoint schemes.**

***Index Terms*—Reliability optimization, cloud computing, checkpointing.**

I. Introduction

RELIABILITY is a critical requirement for modern datacenters. High reliability is desirable for many jobs and applications, because even a small service downtime may potentially lead to business interruption with hefty financial penalties. In a public cloud, reliability is provided as a fixed service parameter, e.g., all Amazon EC2 users are expected to receive 99.95% reliability [1]. In other words, the best reliability that cloud customers can get right now is also the worst. Thus it is up to the cloud customers to harden the jobs running within their VM instances in order to enhance reliability for critical applications. Now, applications that provide increased reliability *do* exist, e.g., Oracle's payroll and general ledger programs. However, these approaches are specific to applications or require specific problem structures, and providing elastic reliability for the masses remains an elusive goal in cloud computing today.

This article introduces an approach for assigning elastic reliability to heterogeneous datacenter jobs via distributed checkpoint scheduling and reliability optimization. Virtual Machine (VM) checkpointing is a widely-employed, application-transparent solution to improve reliability in public clouds [4], [5]. To optimize reliability of *a single job*, prior work has proposed a number of models for calculating the optimal checkpoint schedule [6], [7], [8], [9], [10], [11], and several algorithms for balancing checkpoint workload and performance overhead have also been proposed in [15], [16], [17]. Unfortunately, these solutions fall short in optimizing checkpoints of *multiple jobs* whose reliability requirements may vary significantly, due to their inadequacy of taking into account resource contention among different jobs' checkpoints.

In a multi-job scenario, uncoordinated VM checkpoints taken independently run the risk of interfering with each other [12], [13] and may cause significant resource contention and reliability degradation [2]. In particular, the time to save local checkpoint images is determined largely by how I/O resources are shared, while the overhead to transfer locally saved images to networked storage relies on how network resources are shared. In a large datacenter, chances are that VM checkpointing, if unmanaged and uncoordinated, would encounter severe network and I/O congestion, resulting in high VM checkpointing overhead and reliability loss. For a large datacenter, a centralized checkpoint scheduling scheme that micro-manages each job's checkpoints is impractical for handling tens of thousands of jobs. Distributed checkpoint scheduling is needed for achieving our goal of providing elastic reliability as a service.

To this end, we propose a novel job-level self-management approach that not only enables distributed checkpoint scheduling but also optimizes reliability assignments to individual jobs. Our contention-free scheduling solution is inspired by the Carrier Sense Multiple Access (CSMA) method, a distributed protocol for accessing a shared transmission medium, wherein a node verifies the absence of other traffic before transmitting on the medium [25], [26], [27]. If a job senses any ongoing checkpoint actions at its serving hosts, it waits (or backs-off) for an indefinite amount of time and keeps silent if any of its hosts is busy or becomes busy during its backoff. The proposed

Manuscript received March 18, 2020; revised July 26, 2020 and September 29, 2020; accepted October 2, 2020. Date of publication October 14, 2020; date of current version June 10, 2021. The work of Howie Huang was supported by NSF under Grant 1350766, Grant 1618706, and Grant 1717774. The associate editor coordinating the review of this article and approving it for publication was Y. Diao. *(Corresponding author: Tian Lan.)*

Yu Xiang is with the AT&T Labs Research, Middletown Township, NJ 07748 USA.

Hang Liu, Tian Lan, Howie Huang, and Suresh Subramaniam are with the Department of ECE, George Washington University, Washington, DC 20052 USA (e-mail: tlan@gwu.edu).

Digital Object Identifier 10.1109/TNSM.2020.3030937

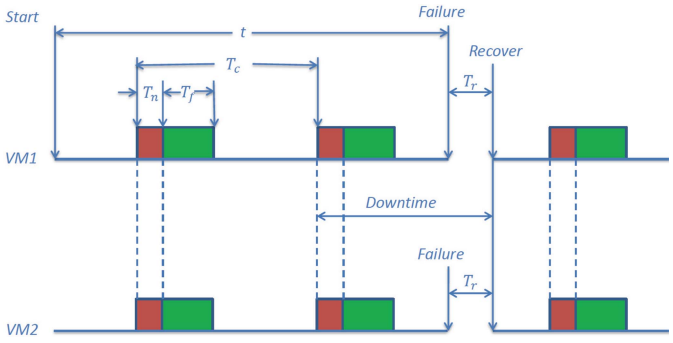


Fig. 1. Multiple VMs belonging to the same job must be checkpointed simultaneously to avoid cascaded rollbacks. It increases the chance of checkpoint contention.

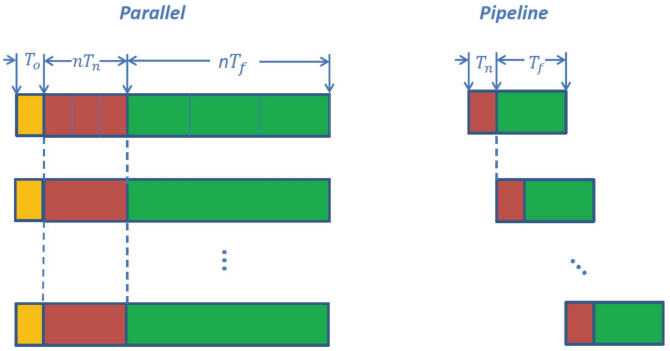


Fig. 2. Fully coordinated checkpoint scheduling in a pipeline mode significantly reduces resource contention over parallel checkpoints.

framework allows jobs to jointly optimize their reliabilities via a fully distributed protocol for checkpoint scheduling. In particular, we model the system evolution as an embedded Discrete Time Markov Chain, solve the stationary distribution, derive the mean checkpoint interval under the proposed scheduling protocol, and finally characterize the resulting reliability of all jobs in the system. The analysis enables us to develop a novel reliability optimization algorithm for the proposed framework, which is able to re-balance reliability assignment of different users in an online environment with dynamic job arrivals and departures. Further, we show that when sensing rates meet a certain set of system equations, the optimal reliability assignment can be found in closed form.

We conduct simulations and experiments to compare our method with contention-oblivious checkpoint scheduling, wherein each job simply checkpoints its VMs at a predetermined rate regardless of any contention from other jobs' checkpoints. The numerical results show that our proposed contention-free framework can achieve a reliability of two nines (i.e., 99%), which is one nine improvement over contention-oblivious scheme (i.e., 90%), while reducing application downtime by up to 18.3%. To the best of our knowledge, this is the first work using a CSMA-based scheme for distributed datacenter resource scheduling and reliability optimization. The main contributions of this article are summarized as follows:

- We harness CSMA-based interference management to provide a distributed and contention-free checkpoint scheduling protocol. Our solution is well suited for implementation in large-scale datacenters, as its job-level distributed checkpoint scheduling mechanism can effectively mitigate resource contentions caused by concurrent job checkpoints.
- Reliability received by each individual job is characterized in closed form by studying the stationary behavior of our proposed protocol. It enables a joint reliability optimization where flexible service-level agreements (SLAs) are negotiated through a joint assessment of all jobs' reliability requirements and total datacenter resources available.
- Results are validated via a proof-of-concept prototype that leverages readily available implementations in Xen and Linux. The proposed CSMA-based checkpoint scheduling is shown to significantly reduce checkpoint interference and improve reliability by one order of magnitude over contention-oblivious checkpoint schemes.

The rest of this article is organized as follows. Section II discusses the related work. Section III introduces the system model and illustrates the necessity for distributed, contention-free checkpoint scheduling. Our theoretical analysis of CSMA-based checkpoint scheduling and reliability optimization are presented in Section IV. Section V contains experimental results via a proof-of-concept prototype, and Section VI concludes this article.

II. Related Work

Checkpointing and restoring the state of a running virtual machine has been crucial for fault tolerance in virtualized cloud environments. While sparse checkpoints can result in loss of reliability, checkpointing too frequently may lead to a large system overhead. References [9], [11] study the impact of checkpointing on execution time of an application under different failure and checkpoint models. Existing work has explored a number of approaches for optimizing the checkpointing scheme. An age-dependent checkpointing model is presented in [8], wherein the authors proposed several kinds of statistical approximation schemes taking account of queuing effects, to find the optimal checkpoint interval that maximizes system reliability. References [6], [10] propose online learning algorithms (Bayesian learning and Q-learning) that provide an optimal interval for statistical checkpointing models, minimizing the operating cost incurred from checkpoint overhead. To further reduce the network traffic and local system overhead from checkpointing, while maintaining necessary checkpoints for reliability, [15] proposes a smart checkpoint infrastructure where read-only content is differentiated from read-write parts in VM images. Thus, the read-only parts need to be checkpointed only once, while the rest of checkpoints only save the modifications in read-write parts, and the checkpointing time is reduced. However, these solutions only address checkpointing for a single job scenario, as none of these considered the checkpointing resource interference where multiple jobs are running across VMs.

Recent works also endeavor to solve the resource contention between checkpoints in a multi-job scenario. Reference [14]

develops a cooperative checkpoint scheduling strategy for concurrent jobs. The strategy involves quantifying the impact of interference on the I/O bandwidth, and scheduling checkpoints only when I/O bandwidth utilization is below a certain threshold. However, this does take VM image transfer into account, so resource contention at network level still exists. As a distributed protocol for transmission sharing schemes, CSMA has been widely adopted for solving both I/O and network resource contentions. Reference [25] introduced an adaptive CSMA scheduling algorithm, to resolve link collisions in a multi-hop wireless network, where transmitter of a link would periodically sense link transmissions and back-off if traffic is intense, which improves network resource contention and maximizes link throughput. While CSMA-inspired algorithms for checkpoint scheduling were introduced in [33], they rely on local search heuristics to optimize reliability in a dynamic setting. In contrast, in this article we provide a complete analytical framework for CSMA-inspired checkpoint scheduling protocols, enabling a joint reliability optimization via a fully distributed protocol for checkpoint scheduling.

Cloud computing is a primary choice for a wide range of users from both academia and industry. However, computers come with failures, and data center is no exception. Common cloud failures include software error, hardware failure, scheduling error, service error, power outage, human operational errors, etc., [42]. Reliability is hence of paramount value to the well-being of data centers. We find checkpointing, replication, logging and VM migration are viable options to improve the data center reliability [41]. However, performing the aforementioned operations to enhance reliability is likely to experience contentions at various stages. Using checkpoint as an example, one might suffer from contentions when multiple VMs are capturing the VM states on the same physical machine, or transferring the captured states out from the same physical machine, or writing the received checkpoint files to the same storage node [43]. In this article, we use CSMA method to alleviate the potential contentions at all phases of checkpoint. We anticipate that this method will also benefit other methods such as replication, logging and VM migrations.

III. System Model and Motivations

A. Reliability Model Using Checkpoints

In the simplest form, a Virtual Machine Monitor (VMM) can periodically record a clear state of the running VMs, including a full image of the VM's memory, CPU, and all the device states, and flush resulting VM images to a central storage server to establish recovery points [12], [18], [19]. For a single job consisting of multiple VMs, uncoordinated checkpoints taken independently of each other [12], [13] run the risk of cascaded rollbacks if causality is not respected. This can be avoided by taking synchronous checkpoints of all the VMs that a job comprises; however, the probability of checkpoint contention increases as jobs often consist of multiple VMs.

As shown in Figure 1, a single job periodically checkpoints all its VMs every T_c seconds. Each checkpoint requires time $T_n + T_f$ to complete, which includes time T_n to suspend all its VM executions in order to save a consistent local checkpoint image, as well as time T_f to transfer the saved VM images to a remote destination. After a failure occurs, the job can be restored from an available checkpoint and rolled back to the last saved state with recovery time T_r. We define reliability by 1 minus the fraction of expected service downtime. More precisely, let a failure occur at time *t* after the *n*th checkpoint is fully completed. Then,

$$R = 1 - \mathbb{E}\left[\frac{\text{Service Downtime}}{\text{Total Service Time}}\right] = 1 - \mathbb{E}\left[\frac{t - (n-1)T_c - T_n - T_f + nT_n + T_r}{t + T_r}\right], \quad (1)$$

where nT_n is the total service downtime due to taking checkpoints, $t-(n-1)T_c - T_n - T_f$ is the lost service time due to rollback, and $\mathbb{E}$ is an expectation over all random factors, e.g., checkpoint overhead and failure time. We note that the transfer time does not count directly as downtime, since VMs can resume execution once a local copy of checkpoint image is created. However, if a failure occurs before the transfer completes, then the execution has to be rolled back because the local checkpoint image has not yet been properly stored on a remote server. Thus, a longer transfer time T_f would lead to higher downtime only if a failure occurs. For clarification, we summarize main notations in this article in Table I.

TABLE I
Main Notation

Symbol	Meaning
$\mathcal{N}$	N job indexed by $i = 1, \ldots, N$
$\mathcal{S}$	S hosts indexed by $h = 1, \ldots, S$
λ_i	Sensing rate of job i
μ_i	Service rate to checkpoint job i
τ_i^c	Mean checkpoint time of job i
τ_i^r	Mean rollback and recovery time of job i
f_i	Mean failure rate of job i
R_i	Reliability of job i
$\mathcal{X}_k$	A state in our Markov Chain model
$P_{\mathcal{X}_k,\mathcal{X}_l}$	Transition rate between states $\mathcal{X}_k$ and $\mathcal{X}_l$
π_k	Stationary distribution in state $\mathcal{X}_k$
$\mathcal{A}_i$	A set of all states containing job i
$\mathbb{E}[Y]$	Expectation of random a variable Y

For a single job, a number of proposals for determining the optimal temporal scheduling of checkpoints have been provided in [15], [16], [17], [20], [21]. Further, protocols for taking consistent snapshots of distributed services in virtualized environments using OpenFlow hardware to improve fault tolerance are presented in [22]. However, these results do not take into account the contention among different checkpoints in a multi-job scenario. It is shown that as the size of datacenter grows large, uncoordinated VM checkpoints from different jobs may cause significant resource contention and result in high checkpoint overhead T_n (due to I/O resource contention) and T_f (due to network resource contention) [2], which directly translates to severe reliability degradation according to (1). A utility-based framework for joint reliability maximization under data center resource constraints is proposed in [2]. The solution is shown to reduce expected service downtime by as much as an order of magnitude,

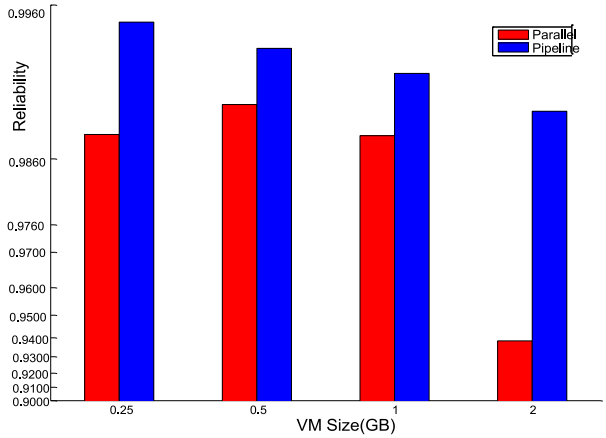


Fig. 3. Fully coordinated pipeline checkpoint schedule significantly reduces contention and improves reliability over parallel checkpoint schedule. Reliability calculated with 8 failures/year.

even though it requires centralized coordination/scheduling and does not allow a distributed implementation at a large scale.

B. Need for Contention-Free Checkpoint Scheduling

Job reliability benefits from mitigating checkpoint overhead, while checkpoint frequency also has to be determined to amortize not only service downtime due to taking checkpoints but also potential service loss due to failure and rollback. Due to resource sharing in datacenters, all of these require a joint checkpoint scheduling over all jobs that share a common pool of resources. Consider two extreme cases for multi-job checkpoint scheduling: parallel and pipeline scheduling, as illustrated in Figure 2. In the parallel mode, the checkpoints of all N jobs are done at the same time and the total I/O and network bandwidth are shared among them. In theory, the time to save a local checkpoint T_n and to transfer VM images T_f will be at least N times higher than when checkpoints are taken one at a time, and there can also be an overhead T_o for switching between simultaneous VM checkpointing processes in the parallel mode. On the other hand, if fine-grained checkpoint control is possible, checkpoints of jobs can be taken one immediately after another in a pipelined fashion by overlapping image-saving time of one job's checkpoint with the image transfer time of another job. With such completely coordinated checkpoints, jobs can take full advantage of all I/O and network bandwidth resource available, causing minimal interference to others.

To demonstrate the advantage of checkpoint coordination, we set up a simple experiment involving two hosts and four VMs on each host to quantify how much reliability is achieved under each scheme. We implement both parallel and pipeline scheduling, and measure the checkpoint overhead and VM image transfer time. Figure 3 shows that pipeline scheduling outperforms parallel scheduling by nearly an order of magnitude for various VM sizes. Significant I/O contentions are observed in the parallel mode since multiple VMs take checkpoints simultaneously, while the pipeline mode avoids such contention and achieves much higher reliability. Further, the

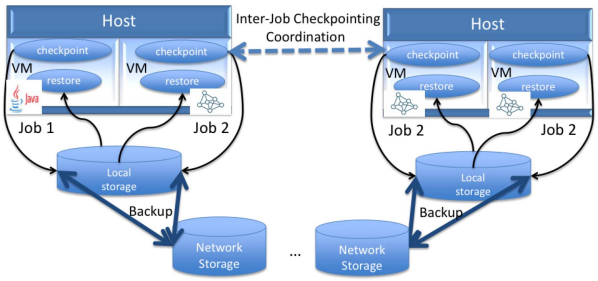


Fig. 4. Our proposed architecture for checkpoint scheduling, with an illustration of two jobs, consisting of 1 and 3 VMs respectively and placed on 2 hosts.

reliability improvement increases as VM size grows because the larger the VM size, the longer time it needs to save and transfer checkpoint images, and the more likely checkpoint contentions occur.

We conclude that checkpoint scheduling is crucial to provide high reliability in a multi-job scenario. Even though pipeline scheduling completely avoids checkpoint interference and is able to efficiently utilize all I/O and network bandwidth available, such a centralized coordination and micro-management approach is prohibitive in large-scale datacenters hosting tens of thousands of jobs. Therefore, a practical checkpoint scheduling scheme should (i) allow a distributed implementation without relying on any centralized, fine-grained checkpoint control, (ii) be able to schedule contention-free checkpoints for a large number of jobs that may have heterogeneous parameters and demands, and (iii) enable a joint reliability maximization to assign the optimal reliability level to each job that suits its demand. To this end, this article makes novel use of the CSMA protocol in wireless inference control to derive a distributed, contention-free checkpoint scheduling protocol with joint reliability optimization.

IV. Our Proposed Protocol and Reliability Optimization

In this section, we propose a CSMA-based checkpoint scheduling protocol and quantify the resulting reliability received by each job in closed-form. Unlike existing work [25], [26], [27] that apply CSMA to wireless interference management and are often concerned with data throughput, our reliability analysis quantifies the reliability received by each job in closed-form and enables joint reliability optimization of all jobs via a utility framework. The protocol is inspired by CSMA but is applied to datacenter resource sharing to achieve distributed, contention-free checkpoint scheduling in large-scale datacenters. Each job may consist of one or more VMs, which are distributed to different physical machines (or servers). Figure 4 shows an overview of the proposed system architecture. It illustrates 2 jobs consisting of 1 and 3 VMs respectively and placed on 2 hosts. Our checkpoints are organized at the job level - if a checkpoint of a job is triggered, all VMs that belong to the job first save their checkpoint images to the local storage (in order to minimize VM downtime) and then transfer them to the networked storage to avoid host failure.

In our design, *each job achieves reliability optimization via self-management in two ways*: first, each job autonomously determines its own checkpointing scheduling based on locally available information, e.g., the co-location of other jobs and occurrence of checkpoint contention. Second, each job autonomously updates its checkpoint rate based on locally available optimal solutions, which is done at runtime with no dependence on any centralized management decisions.

In this section, we will first introduce the CSMA-based checkpoint scheduling protocol, characterize the resulting reliability via a Markov Chain analysis of system stationary distributions, and then present a joint reliability optimization. Theoretical results obtained in this section will be validated through a prototype implementation in Section V.

A. CSMA-Based Checkpoint Scheduling

We consider a datacenter serving N jobs denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$ and using S servers denoted by $\mathcal{S} = \{1, 2, \ldots, S\}$. Each job i is comprised of h_i VMs that are hosted on a subset of servers, denoted by $\mathcal{H}_i$.

As discussed in Section III, mitigating checkpoint contention can significantly reduce service downtime and improve reliability. To develop a checkpoint scheduling protocol that is not only contention-free but also fully distributed, we extend an idealized model of CSMA as in [25], [26], [27]. CSMA is a probabilistic medium access control protocol in which a node verifies the absence of other traffic before transmitting on a shared transmission medium. Our proposed checkpoint scheduling works as follows: Each job *i* makes the decision to create a remote checkpoint image based only on its local parameters and observation of contention. If job *i* senses ongoing checkpoints[1] at any of its serving hosts (i.e., any host *s* such that $s \in \mathcal{H}_i$), then it keeps silent. If none of its serving hosts is busy, then job *i* waits (or backs-off) for a random period of time which is exponentially distributed with the mean $1/\lambda_i$ and then starts its checkpointing.[2] During the back-off, if some contending job starts taking checkpoints, then job *i* suspends its back-off till the contending checkpoint is complete. We note that waiting a random back-off time with different mean allows us to adjust different jobs' checkpointing probabilities. It will not cause excessive idle time because only the relative values of $1/\lambda_i$ matter and the mean waiting time can be set small enough in this CSMA model.

For analytical tractability, we assume that the total time of saving a local checkpoint and transferring it to a remote destination is exponentially distributed with the mean of $1/\mu_i = \mathbb{E}(T_n + T_f)$. This assumption of exponential checkpoint time can be further removed using results in [27]. In such an idealized CSMA model, if sensing time is negligible and back-off time follows a continuous distribution, then the probability for two contending checkpoints to start at the same time is 0 [25]. Therefore, the CSMA-based protocol, summarized in Figure 5 achieves contention-free, distributed scheduling of job checkpoints. We hasten to point out that the back-off time parameters λ_i need to be optimized to ensure fairness between different jobs, e.g., to prevent checkpoint-intensive jobs from blocking others. We will quantify the reliability received by different jobs for a given set of λ_i in Section IV-C and leverage the result to jointly optimize job reliability in Section IV-D.

[1]This can be achieved by assigning a timer to each job and having a controller in each host's hypervisor to monitor the status.

[2]The random backoff time is to ensure that two potentially-contending jobs that sense no contention from other jobs do not start checkpointing at the same time and trigger a contention.

Assign positive sensing rates $\lambda_i > 0\ \forall i$

Each job independently performs:
Initialize backoff timer B_i
while job i is running
  while $B_i > 0$
    if any server in $\mathcal{H}_i$ is busy
      Job i keeps silent
      Generate new backoff: B_i = exponential with mean $\frac{1}{\lambda_i}$
    end if
    Update $B_i = B_i - 1$
  end while
  Checkpoint all VMs of job i
  Generate new backoff: B_i = exponential with mean $\frac{1}{\lambda_i}$
end while

Fig. 5. Our contention-free, distributed checkpoint scheduling protocol inspired by CSMA.

In contrast to existing CSMA analysis that focuses on data throughput, this article aims to quantify individual-job reliability resulting from such contention-free, distributed checkpoint scheduling protocol in large-scale datacenters. It requires us to investigate the distributions of checkpoint intervals T_i, which are random variables due to the CSMA-based, probabilistic checkpoint scheduling protocol. Given a set of sensing rates $\lambda_1, \ldots, \lambda_N$, we use R_i to denote the reliability received by job *i* in the proposed checkpoint scheduling protocol. Notice that reliability also depends on service rates $\mu_1, \ldots, \mu_N$ and job failure rate f_i, which may further depend on server failure model and VM placement. In this article, we focus on the checkpoint scheduling protocol and reliability maximization by optimizing parameters $\lambda_1, \ldots, \lambda_N$. Using our model, we are able to find the reliability received by each job in closed form and then perform reliability optimization jointly over all jobs with respect to their utilities.

B. Markov Chain Model

In order to optimize reliability, we first need to obtain the reliability each job receives in the CSMA-based checkpoint scheduling protocol for given sensing rates $\lambda_1, \ldots, \lambda_N$. We make use of a Markov Chain model, which is commonly employed for CSMA analysis in wireless interference management. The Markov Chain for analyzing the protocol depends on the sensing rate λ_i and checkpoint overhead μ_i, as well as the datacenter VM placement that determines the pattern of job interference. We will first describe the model in this subsection and then use it to derive job reliability in closed form to enable reliability optimization.

For any time *t*, we define a system state as the set of jobs actively taking checkpoints at *t*. Since our CSMA-based protocol achieves contention-free checkpoint scheduling, in each state, a set of non-conflicting jobs (known as an Independent Set) are scheduled. We assume that there exist $K \leq 2^N$ possible states, represented by $k = 1, \ldots, K$. Let $\mathcal{X}_k \subseteq \mathcal{N}$ denote

the subset of jobs actively taking checkpoints in state k. If job i is not taking checkpoints in state k and all of its conflicting jobs are not taking checkpoints, the state k with checkpointing jobs $\mathcal{X}_k$ can transit to state k' with checkpointing jobs $\mathcal{X}_k \cup \{i\}$, with a rate λ_i (i.e., job i starts its checkpoint). Similarly, state k' with $\mathcal{X}_k \cup \{i\}$ can transit to state k with $\mathcal{X}_k$, with a rate μ_i (i.e., job i completes its checkpoints). It is easy to see that the system state at any time is a Continuous Time Markov Chain (CTMC).

Unlike existing CSMA analyses for wireless systems that focus on throughput, our goal is to quantify job reliability using the Markov Chain model. According to Equation (1), this requires the characterization of the distribution of checkpoint overhead T_n, T_f, and checkpoint interval T_i, which are related to sojourn time and returning time of the CTMC. We first transform the CTMC into an embedded Discrete Time Markov Chain (DTMC) that is easier to analyze. Since the embedded chain also has different holding times for its states, we further apply the uniformization technique to obtain a randomized DTMC. It is sufficient to consider transitions between states that differ by one job because there is no contention in our idealized CSMA model. Let v be a uniformization constant that is sufficiently large. Then, the DTMC has the following transition probabilities:

$$P_{\mathcal{X}_k,\mathcal{X}_k\cup\{i\}} = \frac{\lambda_i}{v} \text{ and } P_{\mathcal{X}_k\cup\{i\},\mathcal{X}_k} = \frac{\mu_i}{v}, \tag{2}$$

where $P_{\mathcal{X}_k,\mathcal{X}_l}$ denotes the transition probabilities from state $\mathcal{X}_k$ to state $\mathcal{X}_l$. Due to uniformization, we define $v_k = \sum_{l\neq k} v \cdot P_{\mathcal{X}_k,\mathcal{X}_l}$ to be the sum of transition probabilities out of state $\mathcal{X}_k$ and add a self-transition rate $1 - v_k/v$ so that the transition probabilities form a stochastic matrix. This means we have

$$P_{\mathcal{X}_k,\mathcal{X}_k} = 1 - \frac{v_k}{v}. \tag{3}$$

Now we can study the properties of the original CTMC through the DTMC whose state transitions occur according to the jump times of an independent Poisson Process with rate v, which is typically used in analyzing CSMA algorithms [25], [26], [27]. Fig. 6 (a) gives an example datacenter with 3 jobs and 2 hosts. If each host is able to checkpoint 1 VM at a time without incurring any performance loss, then checkpoints of job 3 conflict with those of jobs 1 and 2, whereas jobs 1 and 2 can take parallel checkpoints without any resource contention. Therefore, this system has $K = 5$ feasible states (or Independent Sets): $\{\cdot\}$, $\{1\}$, $\{2\}$, $\{3\}$, $\{1,2\}$. State $\{\cdot\}$ means no job is taking checkpoints, $\{i\}$ means a single job i takes checkpoints for $i = 1, 2, 3$, and $\{1, 2\}$ means jobs 1 and 2 take checkpoints at the same time.

Given the above DTMC model, we are interested in analyzing its stationary behavior, which reveals the distributions of checkpoint overhead T_n, T_f, and checkpoint interval T_i. The transition probability matrix P of the DTMC has size K by K and its stationary distribution is denoted by $\pi_1, \ldots, \pi_K$, satisfying

$$(\pi_1, \ldots, \pi_K) = (\pi_1, \ldots, \pi_K) \cdot P, \tag{4}$$

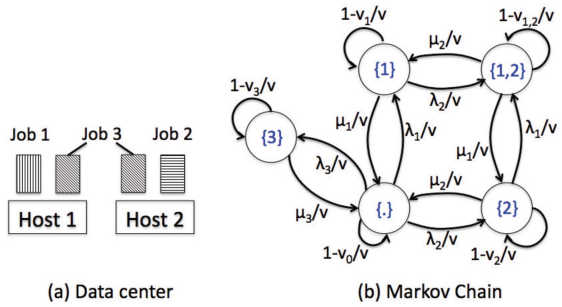


Fig. 6. Example: 3 jobs and corresponding Markov Chain.

where π_k is the stationary probability that the DTMC stays in state $\mathcal{X}_k$. In the following lemma, we show that the stationary distribution can be obtained in closed form for our DTMC model.

Lemma 1: When no checkpoint interference (i.e., contention) is permitted, the DTMC has stationary distribution:

$$\pi_k = \frac{\prod_{i\in\mathcal{X}_k} \lambda_i \cdot \prod_{j\notin\mathcal{X}_k} \mu_j}{C_\lambda}, \tag{5}$$

where C_λ is a normalization factor such that $\sum_k \pi_k = 1$.

Proof: This lemma can be directly proved by showing that the stationary distribution in Equation (5) satisfies the detailed balance equation $\pi_k P_{\mathcal{X}_k,\mathcal{X}_l} = \pi_l P_{\mathcal{X}_l,\mathcal{X}_k}$, $\forall k, l$. Therefore, the DTMC is time-reversible and its stationary distribution depends on rates λ_i, μ_i of all jobs. ■

C. Reliability Analysis

Now we can analyze the stationary behavior of the CTMC through the DTMC and an independent Poisson Process with rate v. From Equation (1), reliability is defined by 1 minus the fraction of service downtime. It means that we need to obtain the distributions of checkpoint overhead T_n, T_f, and checkpoint interval T_i from the Markov Chain model. We assume that each job has a known Mean Time to Failure (MTTF) $1/f_i$ and its failure time is modeled by an exponential distribution. In practice, the MTTF can be estimated from existing failure models or large-scale datacenter event logs [28], [29]. For example, if each server has independent failures according to a Poisson Process with rate f_0 and job i is hosted by m_i different servers, then we have $f_i = m_i \cdot f_0$.

Consider checkpoint overhead T_n, T_f, and checkpoint interval T_i in our CSMA-based protocol for a single job i. Let $\mathcal{A}_i = \{\mathcal{X}_k : i \in \mathcal{X}_k\}$ be the set of all states containing job i. It is not hard to see that total checkpoint overhead $T_n + T_f$ is the sojourn time that the CTMC stays within $\mathcal{A}_i$, i.e., the time to checkpoint job i's VMs. Similarly, checkpoint interval T_i is the first returning time of the CTMC to $\mathcal{A}_i$. It is easy to see that both sojourn time and first returning time are random variables whose distributions depend on the Markov Chain model. Using the definition in Equation (1), we first rewrite reliability R_i with respect to random checkpoint overhead and checkpoint interval.

Lemma 2: Let T_i be the random checkpoint interval of job i. If job i has Poisson failures with rate f_i, then its reliability

is given by

$$R_i = 1 - \tau_i^c \mu_i \pi_{\mathcal{A}_i} - f_i \pi_{\mathcal{A}_i} \mathbb{E}\, T_i - f_i \tau_i^r - \frac{f_i \mathbb{E}\big(T_i^2\big)}{2\mathbb{E}\, T_i} \quad (6)$$

where τ_i^c is the mean time to save a local checkpoint image, τ_i^r is the mean repair time, and $\pi_{\mathcal{A}_i} = \sum_{k \in \mathcal{A}_i} \pi_k$ is the sum of stationary distribution of all states in $\mathcal{A}_i$.

Next, we provide some interpretations on the result. First, $\pi_{\mathcal{A}_i}$ is the fraction of time that the Markov Chain spends in states $\mathcal{A}_i$ (i.e., checkpointing job *i* VMs). Since our protocol is contention-free, out of $\mathbb{E}(T_n + T_c) = 1/\mu_i$ seconds on average for each checkpoint, job *i* VMs have to be suspended for $\mathbb{E}(T_n) = \tau_i^c$ seconds to save consistent, local checkpoint images during the process. Therefore, the service downtime due to checkpointing is given by $\tau_i^c \mu_i \pi_{\mathcal{A}_i}$. Second, $f_i \tau_i^r$ is the expected downtime due to failure recovery and repair. Further, because of our assumption of Poisson failures, lost service time due to VM rollback after each failure can be derived using the Poisson Arrival Sees Time Average (PASTA) property, i.e., $f_i \mathbb{E}(T_i^2)/2\mathbb{E}\, T_i$. Finally, when a failure arrives before a checkpoint is completed (which again has probability $\pi_{\mathcal{A}_i}$), all VMs must be recovered from the last available checkpoint images. It implies that an additional rollback time of $\pi_{\mathcal{A}_i} \mathbb{E}\, T_i$ is incurred on average. A formal proof for this lemma can be found in our online technical report [34].

Therefore, the reliability of job *i* can be obtained if $\mathbb{E}\, T_i$ and $\mathbb{E}\, T_i^2$ are known. Next we derive them via their counterparts in the embedded DTMC. Since job *i* takes a checkpoint if the DTMC is in a state belonging to $\mathcal{A}_i$, its checkpoint interval T_i can be measured by the first returning time to $\mathcal{A}_i$, denoted by $t_{\mathcal{A}_i}$. Let $Y_1, Y_2 \ldots$ be a sequence of i.i.d. exponentially-distributed variables with mean $1/v$. We have $T_i = \sum_{l=1}^{t_{\mathcal{A}_i}} Y_l$, which results in

$$\mathbb{E}\, T_i = \mathbb{E} t_{\mathcal{A}_i} \cdot \mathbb{E}\, Y_l = \frac{1}{v} \mathbb{E} t_{\mathcal{A}_i} \quad (7)$$

and

$$\begin{aligned} \mathbb{E}\, T_i^2 &= \mathrm{var}(Y_l) \cdot \mathbb{E} t_{\mathcal{A}_i} + (\mathbb{E}\, Y_l)^2 \cdot \mathbb{E} t_{\mathcal{A}_i}^2 \\ &= \frac{1}{v^2} \Big(\mathbb{E} t_{\mathcal{A}_i} + \mathbb{E} t_{\mathcal{A}_i}^2 \Big). \end{aligned} \quad (8)$$

Here the derivations are straightforward from the i.i.d. property of Y_l, as well as the independence between the DTMC and the underlying Poisson Process (i.e., the independence between Y_l and $t_{\mathcal{A}_i}$). We refer readers to [40] for the details.

Now it remains to find $\mathbb{E} t_{\mathcal{A}_i}$ and $\mathbb{E} t_{\mathcal{A}_i}^2$ in the embedded DTMC. When the number of jobs is large, we can approximate the first returning time $t_{\mathcal{A}_i}$ by an exponential distribution [30]. Then, its second order moment should be $\mathbb{E} t_{\mathcal{A}_i}^2 = 2[\mathbb{E} t_{\mathcal{A}_i}]^2$. To find $\mathbb{E} t_{\mathcal{A}_i}$, we apply Kac's Formula in [30] and obtain the following result.

Lemma 3: The expectation of first returning time $t_{\mathcal{A}_i}$ for the DTMC is given by

$$\mathbb{E} t_{\mathcal{A}_i} = 1 + \frac{v}{\mu_i} \left(\frac{1}{\pi_{\mathcal{A}_i}} - 1 \right). \quad (9)$$

Proof: Checkpoint interval $t_{\mathcal{A}_i}$ is the time that the DTMC first returns to any state in $\mathcal{A}_i$ since the previous time. Let $\mathcal{X}_{(n)}$ be the DTMC state at time *n* under stationary distribution. It is easy to see that

$$t_{\mathcal{A}_i} = \min \Big\{ T | \mathcal{X}_{(0)} \in \mathcal{A}_i, \mathcal{X}_{(1)} \notin \mathcal{A}_i, \mathcal{X}_{(T)} \in \mathcal{A}_i \Big\}, \quad (10)$$

that is the minimum (random) time the chain returns to $\mathcal{A}_i$ after it leaves at time $n = 0$. Applying Kac's Formula [30] to the DTMC, we have $1/\pi_{\mathcal{A}_i} = \mathbb{E}[\tau_{\mathcal{A}_i}^+]$, where $\tau_{\mathcal{A}_i}^+ = \min\{ T | \mathcal{X}_{(0)} \in \mathcal{A}_i, \mathcal{X}_{(T)} \in \mathcal{A}_i, T \geq 1 \}$ is the first hitting time from a stationary distribution. Using the law of total probability, we further have

$$\begin{aligned} \mathbb{E}\Big[\tau_{\mathcal{A}_i}^+\Big] &= \frac{v - \mu_i}{v} \mathbb{E}\Big[\tau_{\mathcal{A}_i}^+ | \mathcal{X}_{(1)} \in \mathcal{A}_i\Big] \\ &\quad + \frac{\mu_i}{v} \mathbb{E}\Big[\tau_{\mathcal{A}_i}^+ | \mathcal{X}_{(1)} \notin \mathcal{A}_i\Big], \\ &= \frac{v - \mu_i}{v} + \frac{\mu_i}{v} \mathbb{E}\big[t_{\mathcal{A}_i}\big], \end{aligned} \quad (11)$$

where the first step uses $\mathbb{P}_{\{\mathcal{X}_{(1)} \in \mathcal{A}_i\}} = 1 - \mu_i/v$ and $\mathbb{P}_{\{\mathcal{X}_{(1)} \notin \mathcal{A}_i\}} = \mu_i/v$ because departure probability from $\mathcal{A}_i$ is a constant μ_i/v from all states. The second step uses the definition of $t_{\mathcal{A}_i}$ in Equation (10), as well as the fact that $\mathbb{E}[\tau_{\mathcal{A}_i}^+ | \mathcal{X}_{(1)} \in \mathcal{A}_i] = 1$ due to the definition of hitting time. Combining Equation (11) and Kac's formula $1/\pi_{\mathcal{A}_i} = \mathbb{E}[\tau_{\mathcal{A}_i}^+]$, we derive the desired Equation (9). This completes the proof. ■

Plugging these results into Equation (6), we can quantify the reliability received by each job *i* in our contention-free, distributed checkpoint scheduling protocol. Again, we refer readers to our online technical report [34] for a complete proof.

Theorem 1: For given rates $\lambda_1, \ldots, \lambda_K$, each job *i* in our protocol receives the following reliability R_i:

$$R_i = 1 - f_i \tau_i^r - \tau_i^c \mu_i \pi_{\mathcal{A}_i} - \frac{f_i}{\mu_i} \left(\pi_{\mathcal{A}_i} + \frac{1}{\pi_{\mathcal{A}_i}} \right). \quad (12)$$

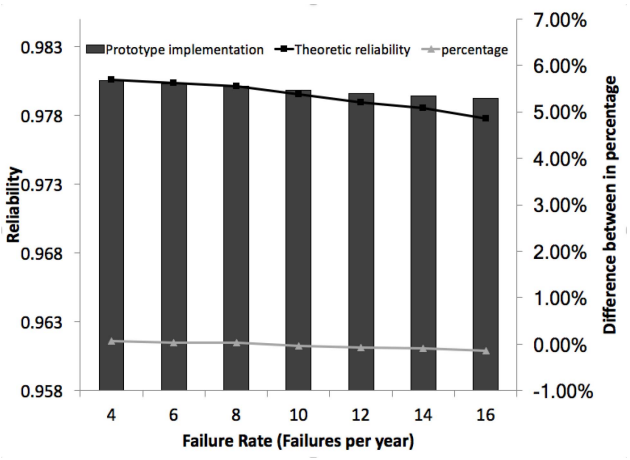


Fig. 7. Comparison of the reliability values from our theoretical analysis with a prototype experiment using 24 VMs in Xen. Our reliability analysis can accurately estimate reliability in the proposed contention-free checkpoint scheduling protocol within a margin of ± 0.2%.

D. Reliability Optimization

We can use Theorem 1 to numerically calculate the reliability of each job *i* for any given rates $\lambda_1, \ldots, \lambda_K$ and failure rate f_i. Let $U_i(R_i)$ be a utility function, representing the

value of assigning reliability level r_i to job i. Our goal is to derive an autonomous reliability optimization where flexible SLAs are negotiated through a joint assessment of users' utility and total datacenter resources available. Toward this end, we formulate a joint reliability optimization through a utility optimization framework [31], [32] that maximizes total utility $\sum_i U_i(R_i)$, i.e.,

$$\begin{aligned} \max \ & \sum_i U_i(R_i) \\ \text{s.t.} \ & R_i = 1 - f_i\tau_i^r - \tau_i^c\mu_i\pi_{\mathcal{A}_i} - \frac{f_i}{\mu_i}\left(\pi_{\mathcal{A}_i} + \frac{1}{\pi_{\mathcal{A}_i}}\right), \\ & \pi_{\mathcal{A}_i} = \frac{1}{C_\lambda} \cdot \sum_{\mathcal{X}_k \in \mathcal{A}_i} \prod_{j \in \mathcal{X}_k} \lambda_j \cdot \prod_{l \notin \mathcal{X}_k} \mu_l, \\ \text{var.} \ & \lambda_1, \ldots, \lambda_K \end{aligned} \tag{13}$$

where C_λ is a normalization factor such that $\sum_k \pi_k = 1$. Here we use the closed-form reliability characterization in Equation (12) and the stationary distribution in Equation (5). We note that fairness between different jobs can be achieved using certain utility functions. In particular, by choosing a family of α-fairness utility functions [38], [39], i.e., $\sum_i U(R_i) = \sum_i R_i^{1-\alpha}/(1-\alpha)$ for $\alpha > 0$, the resulting reliability assignment will achieve different notions of fairness, e.g., proportional fairness for $\alpha = 1$ and max-min fairness for $\alpha \to \infty$. Thus, our proposed optimization algorithm can capture different notions of fairness and balance the reliability achieved by different jobs.

The reliability optimization is computed by maximizing an aggregate utility $\sum_i U_i(R_i)$ over all feasible sensing rates $\lambda_1, \ldots, \lambda_K$. In a dynamic setting, such reliability optimizations must be solved for each job arrival and departure to balance reliability assignments autonomously. We note that many local search heuristics, such as Hill Climbing [35] and Simulated Annealing [36], can be employed to solve the reliability optimization in Equation (13) by incrementally improving the total utility over single search directions. Under certain conditions, we can also characterize the optimal solution in closed form.

Theorem 2: If there exists a set of rates $\lambda_1, \ldots, \lambda_K$ and a positive constant C_λ satisfying the following system of equations, then the rates maximize the aggregate utility in Equation (13) for arbitrary non-decreasing functions:

$$\begin{aligned} \sum_{\mathcal{X}_k \in \mathcal{A}_i} \prod_{j \in \mathcal{X}_k} \lambda_j \cdot \prod_{l \notin \mathcal{X}_k} \mu_l &= C_\lambda \cdot \sqrt{\frac{\tau_i^c \mu_i^2}{f_i} + 1}, \ \forall i \\ \sum_{k=1}^{K} \prod_{j \in \mathcal{X}_k} \lambda_j \cdot \prod_{l \notin \mathcal{X}_k} \mu_l &= C_\lambda. \end{aligned} \tag{14}$$

These rates simultaneously maximize the reliabilities received by all jobs, i.e.,

$$R_i = 1 - f_i\tau_i^r - 2\sqrt{\tau_i^c + \frac{f_i}{\mu_i}^2}, \ \forall i. \tag{15}$$

Proof: We apply the following inequality, $ax + b/x \geq 2\sqrt{ab}$ for all positive $a, b, x > 0$, to the reliability in Equation (12). It implies

$$\begin{aligned} R_i &= 1 - f_i\tau_i^r - \tau_i^c\mu_i\pi_{\mathcal{A}_i} - \frac{f_i}{\mu_i}\left(\pi_{\mathcal{A}_i} + \frac{1}{\pi_{\mathcal{A}_i}}\right) \\ &\leq 1 - f_i\tau_i^r - 2\sqrt{\tau_i^c + \frac{f_i}{\mu_i}^2}, \end{aligned} \tag{16}$$

where we use $x = \pi_{\mathcal{A}_i}$, $a = \tau_i^c\mu_i + \frac{f_i}{\mu_i}$ and $b = \frac{f_i}{\mu_i}$ in the inequality. Notice that the last step holds with equality only if $x = \sqrt{b/a}$. For arbitrary non-decreasing utility functions $U_i(R_i)$, it is easy to see that aggregate utility $\sum_i U_i(R_i)$ is maximized if Equation (16) holds with equality for all $i = 1, \ldots, N$, i.e., all reliability values are maximized simultaneously. This proves the maximum achievable reliability in Equation (15), which can be achieved only if $\pi_{\mathcal{A}_i} = \sqrt{\frac{\tau_i^c\mu_i^2}{f_i} + 1}, \ \forall i$. Plugging the stationary distribution in Equation (5), we get exactly the conditions in Equation (14). ■

Remark: Theorem 2 establishes the maximum utility that our checkpoint scheduling algorithm can achieve. If the conditions in Theorem 2 are satisfied, then solving Equation (14) gives us a set of rates $\lambda_1, \ldots, \lambda_K$, which maximize the aggregate utility in Equation (13) for arbitrary non-decreasing utility functions. As an example, if all jobs share a common resource bottleneck that allows only a single checkpoint at each time, then we have $\mathcal{A}_i = \{i\} \ \forall i$ because any pair of jobs conflict with each other. It is easy to verify that the following rates satisfy conditions in Equation (14), and therefore the reliability optimization can be solved in closed form for arbitrary non-decreasing utility functions:

$$\lambda_i = \frac{\sqrt{\frac{\tau_i^c\mu_i^2}{f_i} + 1}}{\sum_{j=1}^N \sqrt{\frac{\tau_j^c\mu_j^2}{f_j} + 1}} \cdot \frac{1 - \prod_{j=1}^N \mu_j}{\sum_{j=1}^N \prod_{l \neq j} \mu_l}, \ \forall i. \tag{17}$$

Once the optimal solutions are obtained (through either local search heuristics or the sufficient conditions above), each job only has to update its checkpoint rate according to the optimal solutions. Due to the distributed nature of CSMA-based scheduling, jobs can easily reconfigure their checkpoint rates on-the-fly without relying on any centralized checkpoint coordination.

Validation of Theoretical Analysis: To validate the reliability analysis in Theorem 1, we implement a prototype of the contention-free, distributed checkpoint scheduling protocol with 3 servers supporting 24 Xen VMs each with 1GB DRAM. The detailed implementation parameters are provided later in Section V. We first benchmark necessary parameters in our theoretical model using Markov Chain analysis, i.e., the mean checkpoint local-saving time $\tau_i^c = 30.2$ seconds, mean checkpoint overhead $1/\mu_i = 71.5$ seconds, and mean repair time $\tau_i^r = 80.2$ seconds for all jobs $i = 1, \ldots, 24$. So all jobs in the experiment receive an equal reliability value. For a sensing rate of $\lambda_i = 1/(2.5 \text{ days})$ and exponential failures with f_i ranging from 2 to 16 failures per year, we compare the reliability values from our theoretical analysis to the values obtained from the experiment. Figure 7 shows that our theoretical analysis can accurately estimate the reliability values received in

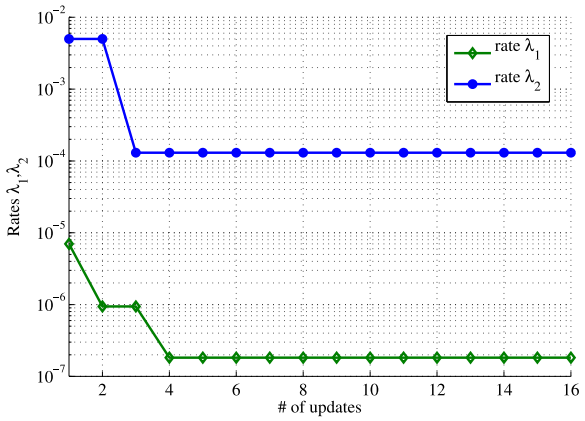


Fig. 8. Plot convergence of sensing rates λ_1, λ_2 when Hill Climbing local search [35] is employed to solve the reliability optimization in Equation (13) with 2 classes of jobs and a utility $2R_1 + R_2$. The algorithm converges within only a few local updates to the optimal sensing rates.

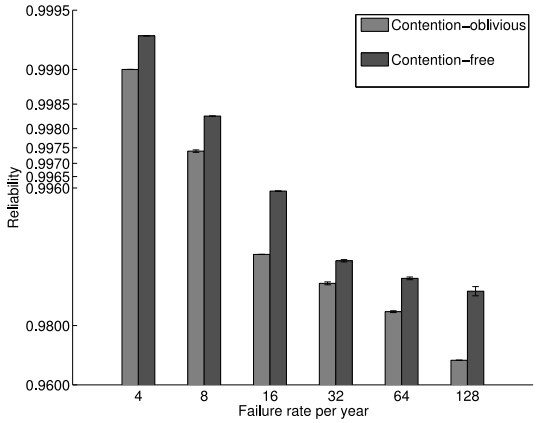


Fig. 9. Reliability for different failure rates.

the proposed protocol, with a small error margin of $\pm$ 1%. This implies that our theoretical reliability analysis provides a powerful tool for reliability estimation and optimization.

Example for Reliability Optimization: To give a numerical example of the proposed reliability optimization, consider a datacenter with 2 classes of jobs: 10 large jobs that contain 10 VMs each and 100 small jobs that contain 2 VMs each. Assume that at most 2 jobs can take non-contending checkpoints at each time, average checkpoint overhead is $\tau_1^c = 50$ seconds for large jobs and $\tau_2^c = 25$ for small jobs, and the recovery time is $\tau_1^r = 400$ seconds and $\tau_2^r = 200$ seconds. Assume that each host has independent failures with rate $f_0 = 2$/year. Then, large jobs have failure rates $f_1 = 10 \cdot f_0 = 6.43e-7$ and small jobs $f_2 = 2 \cdot f_0 = 1.29e-7$. Finally, the total checkpoint time is $1/\mu_{\text{large}} = 200$ seconds for a large job and $1/\mu_{\text{small}} = 100$ seconds for a small job. We implement Hill Climbing local search [35] to find the optimal sensing rates λ_1, λ_2 that maximize a utility $2R_1 + R_2$. As shown in Figure 8, the algorithm converges within a few local updates to the optimal sensing rates. At optimum, large jobs receive a higher reliability $R_1 = 0.99$ than small jobs $R = 0.90$ because the weight of large jobs is twice as that of small jobs in the optimization objective $2R_1 + R_2$.

V. Implementation and Evaluations

We have implemented a prototype of the contention-free checkpoint scheduling in C. In particular, each job employs an individual timer to perform required back-offs and determine when to perform the next checkpoint attempt. A controller is implemented in the hypervisor of each host to monitor the timers of different jobs/VMs and update the busy/idle state. We use a cluster of four machines with Intel Atom CPU D525, 4GB DRAM, 7200 RPM 1TB hard drives, and interconnected with a 1GB/s network. Note that I/O and network bandwidths rather than CPU and memory are the major limiting factors for our tests. To simulate the workload, each VM runs a CPU-intensive benchmark [37] with 1 VCPU, 512MB or 1GB DRAM, and 10GB VDisk. The host OS is Linux 2.6.32 and Xen 4.0. If not specified, the failure rate is eight times per year, and each reliability result is the average of three runs.

We perform a hybrid experiment. First, we run the proposed checkpoint scheduling algorithm on the testbed and obtain a number of measurements through an average of 20 runs: (i) the time to checkpoint each VM and save that checkpoint file locally; (ii) the time for transferring each checkpoint file to a remote server (located in the same rack) that is dedicated for storing the checkpoint images; (iii) the total time for transferring the required checkpoint images during a recovery from the remote server; and (iv) the total time for a job and all its VMs to restore from an available checkpoint. Then, based on these measurements and traces, we simulate datacenter failures with the desired failure rate f_i, e.g., in the range of 4 to 128 failures per year, and compute the corresponding reliability values. Since the failure rate is independent of the checkpointing and recovery overhead, this hybrid approach allows us to evaluate the proposed algorithm under different failure rates without having to cope with prohibitive experiment running time.

Figure 9 shows the reliability of a job when the annual failure rate varies from 4 times to 128 times per year. In this experiment, we run three jobs (two VMs per job, and six VMs in total) and present the average reliability. For small failure rates, the reliabilities for both contention-oblivious and contention-free scheduling are very high. But as more failures occur, the relative benefit of contention-free scheduling becomes more obvious. This is because as failure rate increases, the checkpointing frequency, and thus contentions in the system, increases accordingly. It becomes more important to mitigate contentions in the system. Thus, our proposed optimization algorithm for the contention-free strategy would be able to achieve higher relative improvement.

Reliability as a function of checkpoint interval is shown in Figure 10. Overall, our proposed contention-free scheduling with optimized solution can achieve a reliability of two nines (>99%), which is "one nine" improvement over contention-oblivious schemes (>90%). For contention-free scheduling, the reliability of the system keeps increasing as the checkpoint interval becomes larger. At the same time, the contention-oblivious mechanism increases at a slower pace, but it can also potentially reach as high reliability as contention-free scheduling. This happens because when the checkpoint interval becomes large enough, the probability of checkpoint

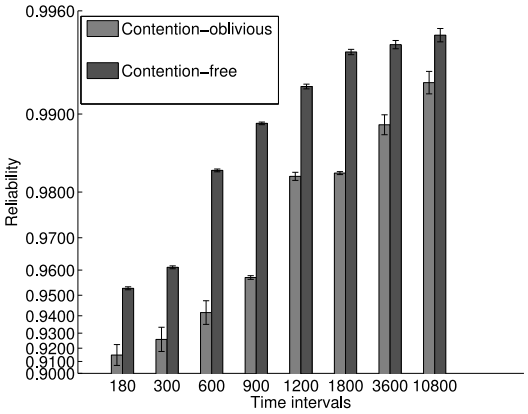


Fig. 10. Reliability for different checkpoint time intervals.

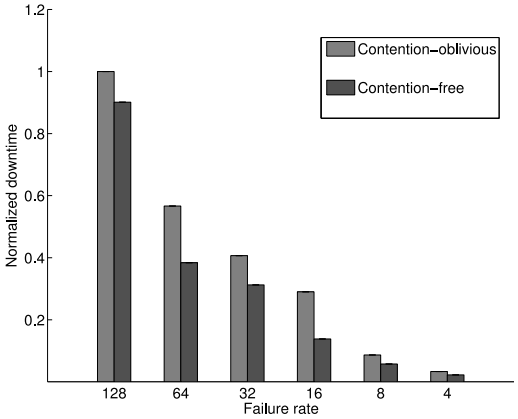


Fig. 11. Normalized downtime for different annual failure rates.

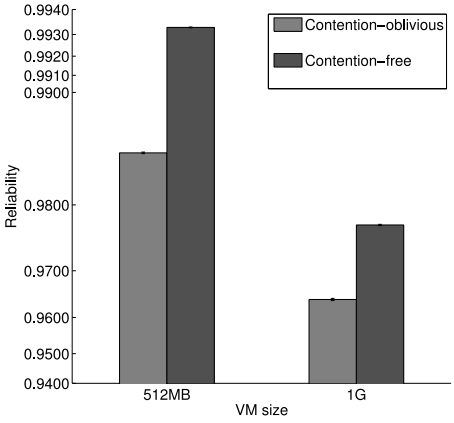


Fig. 12. Reliability for different VM sizes.

contention from different jobs becomes small. To demonstrate the scale of our approach, we also extend this test to simulate 128 jobs. In this experiment, we intentionally intensify the job checkpointing rate in our cluster. As shown in Figure 13, almost all contention-free configuration jobs can achieve a reliability of two nines but the major percentage of contention-oblivious jobs falls into one nine reliability range. In addition, we present the normalized downtime for different annual failure rate settings in Figure 11. Note that the downtime of a system includes the checkpoint time, and recovery time if the host is down. All times are normalized to the downtime

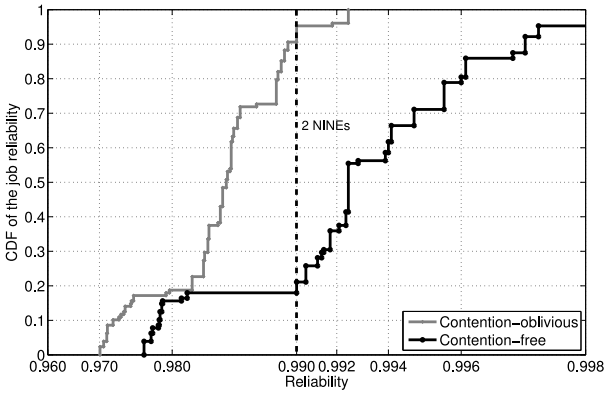


Fig. 13. Reliability of 128 jobs for both contention-free and contention-oblivious checkpoint scheduling.

for contention-oblivious scheduling with 128 failures per year. One can see that our contention-free checkpointing can achieve a reduction in service downtime of upto 18.3% compared to contention-oblivious schemes.

In Figure 12, we show the effect of VM memory size (and, therefore, checkpoint duration) on reliability. Under the same checkpoint interval and failure rate, a job with VM memory size of 512 MB achieves higher reliability due to its smaller memory footprint. The reason is that a larger DRAM size requires more time to suspend the VM and transfer the VM image from the host machine to the destination storage. But the overall trend still shows that our contention-free checkpointing mechanism significantly outperforms contention-oblivious scheduling.

VI. Conclusion

Inspired by the CSMA protocol for wireless interference management, we propose a new protocol for distributed and contention-free checkpoint scheduling in datacenters, where jobs' requirements for reliability vary significantly. The protocol enables datacenter operators to provide elastic reliability as a transparent service to their customers. Using Markov Chain analysis of system stationary behaviours, the reliability that each job receives in our protocol is characterized in closed form. We also present optimization algorithms to jointly maximize all reliability levels with respect to an aggregate utility. Our design is validated through prototype implementations in Xen and Linux, and significant reliability improvements over contention-oblivious scheduling checkpointing are demonstrated via experiments in realistic settings. We will consider checkpointing on a sub-job level and further refine our model, e.g., based on datacenter network topology and using better estimate of modeling parameters, in our future work.

References

[1] Amazon. (Oct. 2008). *We Promise Our EC2 Cloud Will Only Crash Once A Week*. [Online]. Available: https://www.businessinsider.com/2008/10/amazon-we-promise-our-ec2-cloud-will-only-crash-once-a-week-amzn-

[2] N. Limrungsi, J. Zhao, Y. Xiang, T. Lan, H. Huang, and S. Subramaniam, "Providing reliability as an elastic service in cloud computing," in *Proc. IEEE ICC*, Aug. 2012, pp. 2912–2917.

[3] M. Wiboonrat, *An Empirical Study on Data Center System Failure Diagnosis*, ICIMP, New Delhi, India, Jul. 2008.

[4] J. Hui, "Checkpointing orhestration: Toward a scalable HPC fault-tolerant environment," in *Proc. IEEE/ACM Int. Symp. CCGrid*, May 2012, pp. 1–6.

[5] B. Nicolae, "BlobCR: Efficient checkpoint-restart for HPC applications on IaaS clouds using virtual disk image snapshots," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, Nov. 2011, pp. 1–12.

[6] N. Kobayashi and T. Dohi, "Bayesian perspective of optimal checkpoint placement," in *Proc. 9th IEEE Int. Symp. High Assurance Syst. Eng. (HASE)*, 2005, pp. 143–152.

[7] K. M. Chandy, "A survey of analytic models of rollback and recovery strategies," *Computer*, vol. 8, no. 5, pp. 40–47, May 1975.

[8] T. Dohi, N. Kaio, and K. S. Trivedi, "Availability models with age-dependent checkpointing,," in *Proc. 21st IEEE Symp. Rel. Distrib. Syst.*, 2002, pp. 130–139.

[9] N. H. Vaidya, "Impact of checkpoint latency on overhead ratio of a checkpointing scheme," *IEEE Trans. Comput.*, vol. 46, no. 8, pp. 942–947, Dec. 1997.

[10] H. Okamura, Y. Nishimura, and T. Dohi, "A dynamic checkpointing scheme based on reinforcement learning," in *Proc. 10th IEEE Pac. Rim Int. Symp. Depend. Comput.*, Mar. 2004, pp. 151–158.

[11] A. Duda, "The effects of checkpointing on program execution time," *Inf. Process. Lett.*, vol. 16, no. 5, pp. 221–229, 1983.

[12] P. Ta-Shma, G. Laden, M. Ben-Yehuda, and M. Factor, "Virtual machine time travel using continuous data protection and checkpointing," *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 1, pp. 127–134, 2008.

[13] M. Sun and D. M. Blough, *Fast, Lightweight Virtual Machine Checkpointing*, Georgia Tech., Atlanta, GA, USA, May 2010. [Online]. Available: https://www.semanticscholar.org/paper/Fast%2C-Light weight-Virtual-Machine-Checkpointing-Sun-Blough/0f0654e0647e75 add0e2b2ca51dd16f05142e393

[14] T. Herault *et al.*, "Optimal cooperative checkpointing for shared high-performance computing platforms," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, Vancouver, BC, Canada, 2018, pp. 803–812, doi: 10.1109/IPDPSW.2018.00127.

[15] I. Goiri, F. Julià, J. Guitart, and J. Torres, "Checkpoint-based fault-tolerant infrastructure for virtualized service providers," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp.*, Aug. 2010, pp. 455–462.

[16] M. Zhang, H. Jin, X. Shi, and S. Wu, "VirtCFT: A transparent VM-level fault-tolerant system for virtual clusters," in *Proc. Parallel Distrib. Syst. (ICPADS)*, Dec. 2010, pp. 147–154.

[17] Y. Liu, R. Nassar, C. Leangsuksun, N. Naksinehaboon, M. Paun, and S. L. Scott, "An optimal checkpoint/restart model for a large scale high performance computing system," in *Proc. Parallel Distrib. Process. (IPDPS)*, Apr. 2008, pp. 1–9.

[18] A. Warfield, R. Ross, K. Fraser, C. Limpach, and S. Hand, "ParallaX: Managing storage for a million machines," in *Proc. HotOS*, Jun. 2005, pp. 1–9.

[19] R. Badrinath, R. Krishnakumar, and R. Rajan, "Virtualization aware job schedulers for checkpoint-restart," in *Proc. ICPADS*, Dec. 2007, pp. 1–7.

[20] T. Ozaki, T. Dohi, H. Okamura, and N. Kaio, "Distribution-free checkpoint placement algorithms based on min–max principle," *IEEE Trans. Depend. Secure Comput.*, vol. 3, no. 2, pp. 130–140, Mar. 2006.

[21] T. Dohi, T. Ozaki, and N. Kaio, "Optimal checkpoint placement with equality constraints," in *Proc. 2nd IEEE Int. Symp. Depend. Auton. Secure Comput.*, Oct. 2006, pp. 77–84.

[22] A. Kangarlou, D. Xu, U. Kozat, P. Padala, B. Lantz, and K. Igarash, "In-network live snapshot service for recovering virtual infrastructure," *IEEE Netw.*, vol. 25, no. 14, pp. 12–19, Jul. 2011.

[23] H. Liu, "Optimize performance of virtual machine checkpointing via memory exclusion," in *Proc. ChinaGrid Annu. Conf.*, Aug. 2009, pp. 199–204.

[24] B. Schroeder, E. Pinheiro, and W. Weber, "DRAM errors in the wild: A large-scale field study," in *Proc. SIGMETRICS 11th Int. Joint Conf. Meas. Model. Comput. Syst.*, 2009, pp. 193–204.

[25] L. Jiangand and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Trans. Network*, vol. 18, no. 3, pp. 960–972, Jun. 2010.

[26] X. Wang and K. Kar, "Throughput modelling and fairness issues in CSMA/CA based ad-hoc networks," in *Proc. IEEE INFOCOM*, Mar. 2005, pp. 23–34.

[27] S. C. Liew, C. Kai, J. Leung, and B. Wong. (2009). *Back-of-the-Envelope Computation of Throughput Distributions in CSMA Wireless Networks*. [Online]. Available: http://arxiv.org//pdf/0712.1854

[28] International Working Group on Cloud Computing Resiliency (IWGCR). (2012). *Downtime Statistics of Current Cloud Solution*. [Online]. Available: http://iwgcr.files.wordpress.com/2012/06/iwgcr-paris-ranking-001-en1.pdf

[29] H. Gunawi, T. Do, J. M. Hellerstein, I. Stoica, D. Borthakur, and J. Robbins. (2012). *Failure as a Service (FaaS): A Cloud Service for Large-Scale, Online Failure Drills*. [Online]. Available: http://techreports.lib.berkeley.edu/accessPages/EECS-2011-87.html

[30] F. Aldous, *Reversible Markov Chains and Random Walks on Graphs*, Univ. California at Berkeley, Berkeley, CA, USA, Jul. 2002. [Online]. Available: https://www.stat.berkeley.edu/ aldous/RWG/book.pdf

[31] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proc. IEEE*, vol. 95, no. 1, pp. 255–312, Mar. 2007.

[32] T. Lan, X. Lin, M. Chiang, and R. B. Lee, "Stability and benefits of sub-optimal utility maximization," *IEEE/ACM Trans. Netw.*, vol. 19, no. 4, pp. 1194–1207, Aug. 2011.

[33] Y. Xiang, H. Liu, T. Lan, H. Huang, and S. Subramaniam, "Optimizing job reliability through contention-free, distributed checkpoint scheduling," in *Proc. ACM SIGCOMM Workshop Distrib. Cloud Comput. (DCC)*, Aug. 2014, pp. 255–312.

[34] Y. Xiang, H. Liu, T. Lan, H. Huang, and S. Subramaniam. (Jul. 20, 2013). *Optimizing Job Reliability Through Contention-Free, Distributed Checkpoint Scheduling*. [Online]. Available: https://www.seas.gwu.edu/∼tlan/papers/ICAC.pdf

[35] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2020, pp. 111–114.

[36] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983, doi: 10.1126/science.220.4598.671.

[37] J. J. Dongarra, P. Luszczek, and A. Petitet, "The LINPACK benchmark: Past, present and future," *Concurrency Comput. Practice Exp.*, vol. 15, no. 9, pp. 803–820, 2003.

[38] T. Lan, D. Kao, M. Chiang, and A. Sabharwal, "An axiomatic theory of fairness for resource allocation," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 115–132.

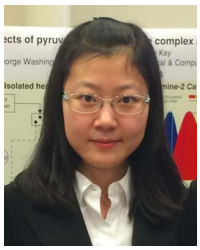
[39] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework," in *Proc. INFOCOM*, Mar. 2012, pp. 1206–1214.

[40] S. K. Ross, *Introduction to Probability Models*, 3rd ed. New York, NY, USA: Academic, 1985, pp. 83–103, ch. 3.

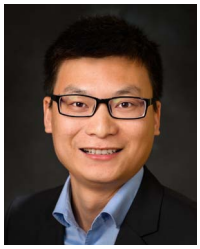
[41] Z. Amin, H. Singh, and N. Sethi, "Review on fault tolerance techniques in cloud computing," *Int. J. Comput. Appl.*, vol. 116, no. 18, pp. 11–17, 2015.

[42] Y. Sharma, B. Javadi, W. Si, and D. Sun, "Eliability and energy efficiency in cloud computing systems: Survey and taxonomy," *J. Netw. Comput. Appl.*, vol. 74, pp. 66–85, Aug. 2016.

[43] I. Zhang, T. Denniston, Y. Baskakov, and A. Garthwaite, "Optimizing VM checkpointing for restore performance in VMware ESXi" in *Proc. USENIX Conf. Annu. Techn. Conf.*, 2013, pp. 1–12.



Yu Xiang received the B.A.Sc. degree in electrical engineering from the Harbin Institute of Technology in 2010, and the Ph.D. degree in electrical engineering from George Washington University in 2015. She is currently a Principal Inventive Scientist with AT&T Labs-Research. Her current research interest includes cloud computing, software-defined storage, mobile edge computing, 5G RAN, and distributed systems.



Hang Liu (Member, IEEE) received the bachelor's degree from the Huazhong University of Science and Technology in 2011, and the Ph.D. degree from George Washington University (GW) in 2017. He is an Assistant Professor from the Department of Electrical and Computer Engineering, Stevens Institute of Technology. Prior to joining Stevens, he was an Assistant Professor with the University of Massachusetts Lowell. He was a Regular PC Member for SC, IPDPS, and HPDC. He received the NSF CRII Award, one of the best papers award in VLDB'20, and was the DARPA/MIT/AMAZON Graph Challenge Champion in 2018 and 2019. He earned the Best Dissertation Award from GWs Department of Electrical and Computer Engineering. In addition, his graph traversal systems are ranked highly in both Graph500 and Green Graph500 benchmarks, which measure the performance and energy efficiency of the worlds most powerful supercomputers.

Tian Lan (Member, IEEE) received the B.A.Sc. degree from Tsinghua University, China, in 2003, the M.A.Sc. degree from the University of Toronto, Canada, in 2005, and the Ph.D. degree from Princeton University in 2010. He is currently an Associate Professor of Electrical and Computer Engineering with George Washington University. His research interests include network optimization and algorithms, cyber security, and machine learning. He received the IEEE Signal Processing Society Best Paper Award in 2008, the IEEE GLOBECOM 2009 Best Paper Award, the INFOCOM 2012 Best Paper Award, and the Securecomm 2019 Best Paper Award.

Howie Huang (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Virginia. He is a Full Professor with the Department of Electrical and Computer Engineering and Director of the Graph Computing Lab (GraphLab), George Washington University. He also holds a courtesy appointment with the Department of Computer Science and is an affiliated Faculty Member with Institute for Data, Democracy and Politics, GWU. Motivated by the needs of big data and cybersecurity applications, he works at the intersection of algorithms, computer architecture and systems, with focus on developing high-performance computing and machine learning techniques tailored for large-scale graph datasets. His GraphLab explores novel applications of graph-based knowledge discovery in computer systems, cybersecurity, social networks, biology, and health.

Suresh Subramaniam (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of Washington, Seattle, in 1997.

He is a Professor and the Chair of Electrical and Computer Engineering with the George Washington University, Washington, DC, USA, where he directs the Lab for Intelligent Networking and Computing. He has published over 220 peer-reviewed papers in these areas. His research interests are in the architectural, algorithmic, and performance aspects of communication networks, with current emphasis on optical networks, cloud computing, data center networks, and IoT. He received the 2017 SEAS Distinguished Researcher Award from George Washington University, and he has been an IEEE Distinguished Lecturer since 2018. He has served in leadership positions for several top conferences, including IEEE ComSocs flagship conferences of ICC, Globecom, and INFOCOM. He serves on the editorial boards of the IEEE/ACM TRANSACTIONS ON NETWORKING and the IEEE/OSA JOURNAL OF OPTICAL COMMUNICATIONS AND NETWORKING. From 2012 to 2013, he served as the elected Chair of the IEEE Communications Society Optical Networking Technical Committee.