Computing (2019) 101:1–2
https://doi.org/10.1007/s00607-018-0672-z

EDITORIAL

Special Issue on NETYS'2016

Parosh Aziz Abdulla[1] · **Carole Delporte**[2]

Published online: 10 November 2018

The aim of the International Conference on Networked Systems is to bring together researchers and engineers from both the theory and practice of distributed and networked systems. The scope of the conference covers all aspects related to the design and the development of these systems, including multi-core architectures, concurrent and distributed algorithms, parallel/concurrent/distributed programming, distributed databases, big data applications and systems, cloud systems, networks, security, and formal verification.

This special issue, comprised of a selection of best papers presented at NETYS'2016, features papers addressing some of the most important issues in networked systems.

The paper "The Out-of-core KNN Awakens: The light side of computation force on large datasets" considers the K-Nearest Neighbors (KNN) algorithm that is widely used in applications such as recommendation systems, information retrieval, and image classification. In general, KNN is used for finding similar entities in a large set of candidates, by computing similarity between the profiles of different entities. The paper addresses a particular shortcoming of the algorithm, namely that it is resource greedy for large datasets. It proposes a novel approach to apply KNN on large datasets by leveraging both disk and main memory efficiently. The main idea is to minimize random accesses to disk, maximize sequential accesses, and use the available memory efficiently.

The paper "A Mechanized Refinement Proof of the Chase–Lev Deque using a Proof System" considers the verification of a concurrent data structure, namely the Chase–Lev work-stealing queue (WSQ). It shows that WSQ is *linearizable* using the CIVL proof system. The latter is a system that employs automated and modular refinement reasoning for concurrent programs. The lowest level description of WSQ is the data structure code described in terms of fine-grained actions whose atomicity is guaranteed by hardware. Higher level descriptions consist of increasingly coarser action blocks obtained using a combination of Owicki–Gries annotations, reduction,

✉ Parosh Aziz Abdulla
parosh.abdulla@it.uu.se

1 Uppsala University, Uppsala, Sweden

2 University of Paris Diderot, Paris, France

and abstraction. The top-level description for WSQ consists of a single atomic action for each operation, where the specification of the action is sufficiently precise to show that WSQ is indeed linearizable.

The paper "Waiting in Concurrent Algorithms" considers different notions of *waiting* between processes in asynchronous concurrent programs. It introduces a spectrum of different levels of waiting ranging from one extreme, i.e., *lock-based* algorithms, which involve "a lot of waiting", to another extreme, i.e., *wait-free* algorithms, which are "free of locking and waiting".

The paper defines this spectrum formally and investigates its properties. It defines a hierarchy of progress conditions, called *k-waiting*, for $k \geq 0$, intended to capture the "amount of waiting" of processes. To illustrate the utility of the new conditions, they are used to derive new lower and upper bounds, and impossibility results for well-known basic problems such as consensus, election, renaming and mutual exclusion.

The paper "Time-Efficient Read/Write Register in Crash-prone Asynchronous Message-Passing Systems" considers implementing the *atomic register*. The atomic register is one of the most basic and useful objects of in distributed systems, and hence its implementation has received a lot of attention. It is well-known that having a strict minority of processes that may crash is a necessary and sufficient requirement to build an atomic register on top of a crash-prone asynchronous message-passing system.

The paper builds on this and provides a new time-efficient asynchronous algorithm of an atomic register that reduces latency in many cases: a write operation always costs a round-trip delay, while a read operation costs a round-trip delay in certain circumstances (when it is not concurrent with a write). When designing this algorithm, the authors have tried to remain as close as possible to the original algorithm proposed by Attiya, Bar-Noy, and Dolev.

Acknowledgements We would like to thank all the authors of NETYS'2016 as well as the reviewers of the selected papers of this special issue who thoroughly evaluated their quality through a rigorous refereeing process.